Implicit Sweep Surfaces

Ryan Schmidt
University of Calgary
Computer Science
rms@cpsc.ucalgary.ca

Brian Wyvill
University of Calgary
Computer Science
blob@cpsc.ucalgary.ca

Abstract

A technique is presented for generating implicit sweep objects that support direct specification and manipulation of the surface with no topological limitations on the 2D sweep template. The novelty of this method is that the underlying scalar field is bounded and C^1 continuous, apart from surface creases. Bounded scalar fields guarantee local influence when modeling with implicit surfaces, an important usbility requirement for interactive modeling. A discrete approximation is also described that supports fast evaluation for bounded scalar fields. The new sweep objects are implemented in an interactive BlobTree modeling tool, providing an intuitive and expressive free-form implicit modeling component. This sweep representation permits conversion of parametric sweep surfaces to implicit volumes. An application to volume reconstruction from parallel contours is also explored.

1. Introduction

Generating a three-dimensional surface by sweeping a 2D curve along a 3D trajectory has a long history in computer graphics [Req80]. This technique is nearly ubiquitous in 3D parametric modeling software. However, few sweep surface methods exist in the implicit domain [Blo97].

Several problems complicate the definition of implicit sweep surfaces. The standard technique for parametric surfaces involves forward mapping of a sweep template from 2D to 3D, where the sweep template is a set of 2D contours [Req80]. In the implicit domain this mapping must be inverted to sample the sweep template. This inverse mapping is often non-trivial and generally not unique.

Implicit sweep templates are functionally-defined 2D scalar fields that represent the desired contour [PSS96b]. The template function should have local support and at least C^1 continuity to produce implicit sweep objects suitable for use in interactive constructive modeling.

Preserving continuity in 3D is another challenge for implicit sweeps with arbitrary trajectories. Even if the surface is not self-intersecting, the bounded scalar field defining the sweep may be. In this case C^1 discontinuities can be introduced in the field away from the surface, producing unexpected and undesired creases when blending multiple surfaces [BWdG05].

We present several improvements to the implicit sweep objects of [CBS96]. First, our technique produces the same surface as a parametric sweep for a given closed contour and sweep trajectory, permitting direct surface specification and manipulation. Second, we eliminate the "star-shaped" sweep template restrictions of [CBS96]. Our bounded C^2 continous sweep template is defined by an arbitrary set of closed 2D contours which may include holes. We develop several endcap styles for open sweep trajectories, and use the operators of [BWdG05] to produce sweeps with C^1 continuous bounded scalar fields.

Our underlying scalar fields are bounded, guaranteeing local influence when modeling and preserveing an informal "principle of least surprise" that is important for interactive implicit modeling. We integrate our sweep objects with the *BlobTree* implicit modeling system [WGG99]. The BlobTree supports constructive modeling of complex hierarchical implicit models. Since our sweeps permit direct surface interaction, they are an intuitive and expressive free-form addition to the BlobTree. This sweep representation supports conversion of parametric sweep surfaces to implicit volumes, as well as implicit volume reconstruction from parallel contours.

We proceed by reviewing related work (Section 2), followed by background material on implicit surfaces (Section 3). Two algorithms for converting contours to scalar fields are described in Section 4, leading to the development of implicit sweep objects in Section 5. Applications of implicit sweep objects are presented in Section 6, followed by our conclusions in Section 7.

2. Related Work

Solid modeling by sweeping a 2D area along a 3D trajectory is a well known technique in computer graphics [Req80] [FvDF*93]. Most early CAD systems [RV82] supported sweep surfaces as boundary representations (b-reps). These B-rep sweep solids lacked a robust mathematical foundation, self-intersecting sweeps were simply invalid. Recent work on the more general problem of sweeping a 3D volume has produced several general sweep theories, [AMBJ00] provides a recent survey. In particular, [AMYB00] explicitly calculates the b-rep created by sweeping an implicitly-defined solid but requires symbolic representations of the solid and sweep trajectory. [SP96] functionally defines implicit swept volumes, however evaluating the resulting function requires slow non-linear optimization algorithms.

[SW97] and [WC02] produce sweep objects represented with volume datasets. The sweep dataset is initialized by sampling a sweep template specified as a 2D image. Volume datasets have a high memory cost and cannot represent arbitrary surface creases. The scalar fields produced are not suitable for constructive implicit modeling[WGG99].

[CBS96] describes implicit sweep primitives with a bounded scalar field. Profile curves defined in polar coordinates are used to create an anisotropic distance field. The sweep surface is an offset surface from the swept profile curve, prohibiting direct surface specification. Profile curves are limited to "star-shapes" by the polar definition. [Gri99] describes implicit generalized cylinders which have a similar limitation.

The key issue in converting parametric sweep surfaces to implicit form is the definition of a suitable sweep template. A 2D scalar field must be defined that represents a closed 2D contour. This scalar field is then swept along some trajectory. [PSS96b] approximates the 2D contour with a polygon and generates a smooth scalar field that represents this polygon. The scalar field can be smoothed at polygon vertices to avoid gradient discontinuities on the sweep surface. The method is extended to direct representation of cubic splines by [PSS96a], and field variation is improved by [BDS*03]. Variational interpolation [YT02] can also be applied to approximate a 2D contour with a scalar field. However, none of these techniques produce a 2D scalar field that is bounded.

To summarize, several implicit sweep models have been proposed that support direct surface specification and manipulation. However, none of these models produces the bounded scalar fields necessary for interactive modeling of complex hierarchical models [SWG05].

3. Implicit Modeling

Given a continuous scalar function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, we can define a surface $\mathcal{S}$:

$$\mathcal{S} = \left\{ \mathbf{p} \in \mathbb{R}^3 : f(\mathbf{p}) = v_{iso} \right\} \tag{1}$$

where v_{iso} is called the *iso-value*. We call this surface an *implicit surface* [Blo97]. Since f defines a scalar field, we frequently refer to it as a *field function* or *field*. This definition also holds in 2D, where $\mathcal{S}$ is a contour.

Equation 1 is misleading in it's mathematical conciseness. Directly specifying function f that generates a desired surface is rather challenging. A reasonable approach is to incrementally construct $\mathcal{S}$ by combining a set of simple implicit surfaces, called *primitives*. A useful class of primitives are *skeletal primitives*, defined by a geometric skeleton E and a one-dimensional function $g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$. For each skeleton E, such as a point or line, we define a *distance function* $d_{\mathsf{E}} : \mathbb{R}^3 \rightarrow \mathbb{R}_+$ that computes the minimum Euclidean distance from $\mathbf{p}$ to E. The field function is then:

$$f_{\mathsf{E},g}(\mathbf{p}) = g \circ d_{\mathsf{E}}(\mathbf{p}) \tag{2}$$

The resulting surface is primarily determined by E, which we require to be finite. While g can be any function, a monotonically decreasing function with local support is desirable. We use [Wyv]:

$$g_{wyvill}(x) = (1 - x^2)^3 \tag{3}$$

where x is clamped to the range $[0, 1]$. This polynomial smoothly decreases from 1 to 0 over the valid range, with zero tangents at each end. The iso-value should also be in the range $[0, 1]$, we choose 0.5.

An important property of this skeletal primitive definition is that the scalar field is *bounded*, meaning that $f = 0$ outside some sphere with finite radius. Bounded fields guarantee local influence, preventing changes made to a small part of a complex model from affecting distant portions of the surface. Local influence preserves a "principle of least surprise" that is critical for interactive modeling.

Another useful property of skeletal primitives is that the iso-value v defines both an implicit surface $\mathcal{S}$ and an *implicit volume* $\mathcal{V}$:

$$\mathcal{V} = \left\{ \mathbf{p} \in \mathbb{R}^3 : f(\mathbf{p}) \geq v_{iso} \right\} \tag{4}$$

Composition operators [Ric73] [WGG99] on implicit surfaces are defined as scalar functions that can be nested to incrementally construct complex models. Valid operators should at minimum produce a new scalar field that defines a closed surface and preserves the volume definition (Equation 4). Under these conditions it is impossible to create a scalar field that does not define a valid surface and volume. This is a desirable property for interactive modeling.

Finally we consider field continuity. Operators that preserve C^1 (gradient) continuity are necessary because the field gradient is used to calculate surface normals. C^1 discontinuities in the field can produce unexpected creases in blend surfaces (Figure 1). This is a significant problem for interactive modeling with implicit surfaces. Implicit sweep objects with self-intersecting scalar fields often contain C^1 discontinuities. Our development of implicit sweeps is largely driven by the need to avoid this situation.

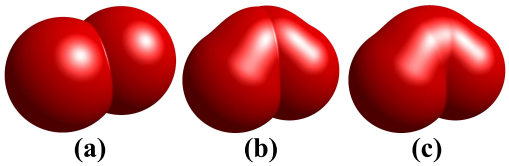

Figure 1. Various common implicit surface operators, such as the Ricci CSG Union, create C^0 discontinuities in the scalar field. While this is desirable on the surface (a) the discontinuity exists throughout the field. When a primitive is blended, (b), the surface will appear to have a crease where the discontinuity region (a plane in this case) intersects the blend surface. Using C^1 CSG operators avoids this issue (c).

3.1. Variational Implicit Curves

One useful technique for generating a 2D scalar field is by interpolating a set of 2D field value samples $(\mathbf{m}_i, v_i)$, where v_i is the desired field value at point $\mathbf{m}_i$. We use a variational interpolation scheme based on thin-plate splines which is globally C^2 continuous. Variational interpolation has been used in 3D to define implicit Surfaces [YT02]. We will apply similar techniques in section 4.2 to create an implicit curve that approximates a 2D curves $\mathcal{C}$.

To unify notation we will denote the variational field function as $f_{\mathcal{C}}$, although the following equation describes general variational interpolation. The function $f_{\mathcal{C}}(\mathbf{u})$ is defined in terms of points $(\mathbf{m}_i, v_i)$ weighted by coefficients w_i, and a polynomial $\mathcal{P}(\mathbf{u}) = c_1\mathbf{u}_x + c_2\mathbf{u}_y + c_3$:

$$f_{\mathcal{C}}(\mathbf{u}) = \sum w_i \|\mathbf{u} - \mathbf{m}_i\|^2 \ln(\|\mathbf{u} - \mathbf{m}_i\|) + \mathcal{P}(\mathbf{u}) \qquad (5)$$

The weights w_i and coefficients c_1, c_2, and c_3 are found by solving a linear system defined by evaluating Equation 5 at each known solution $f_{\mathcal{C}}(\mathbf{m}_i) = v_i$. These coefficients determine a variational solution which is guaranteed to interpolate all sample points $(\mathbf{v}_i, v_i)$ with C^2 continuity while minimizing global curvature [Duc77].

4. 2D Scalar Field Generation

In the parametric domain a 3D sweep surface is created using a 2D curve $\mathcal{C}$. However, in the implicit domain we are creating 3D volumes. Hence we must restrict $\mathcal{C}$ to closed contours. To create a sweep primitive suitable for use in the BlobTree, $\mathcal{C}$ should be represented by a bounded, continuous 2D scalar field. We describe two methods for scalar field generation in this section. Note that the constants mentioned assume $\mathcal{C}$ has been translated and uniformly scaled such that it is contained in a unit square centered at the origin.

4.1. Signed distance fields

One approach to creating a 2D scalar field representing a closed contour $\mathcal{C}$ is to create a *signed distance field*. A *signed distance field* is a mapping from $\mathbb{R}^3$ to $\mathbb{R}$, where each point $\mathbf{p}$ is mapped to the minimum distance from $\mathbf{p}$ to $\mathcal{C}$. Points inside $\mathcal{C}$ are mapped to negative distances. Applying g_{wyvill} (Equation 3) to the infinite signed distance field creates a bounded field but also an offset contour. Adding a constant distance shift, determined by inverting g_{wyvill}, aligns the iso-contour v_{iso} with $\mathcal{C}$.

Unfortunately this technique does not result in a continuous field. If $\mathcal{C}$ is a circle, a single point of C^1 discontinuity exists at the center of the circle. As $\mathcal{C}$ stretches into an ellipse, the discontinuity stretches into a line. If $\mathcal{C}$ is non-convex, C^1 discontinuity lines exist inside and outside the curve (Figure 2). We conclude that signed distance fields are incompatible with our continuity requirement.

4.2. Variational psuedo-distance fields

Discontinuity lines are inherent in the definition of a distance field. Yet a distance field is desirable - using g_{wyvill} (Equation 3) we can convert from a distance field to a bounded scalar field with good blending properties. Our solution is to define a *psuedo-distance field*, which is an approximation to a distance field that is smooth near the discontinuity lines.

Convolution [SW97] can be used to smooth a distance field. However, convolution modifies the iso-contour v_{iso}. Instead we approximate contour $\mathcal{C}$ with a psuedo-distance field $f_{\mathcal{C}}$ generated using variational interpolation (Section 3.1). We then apply g_{wyvill} to $f_{\mathcal{C}}$ to create a bounded, continous 2D scalar field f_M:

$$f_M(\mathbf{u}) = g_{wyvill} \circ f_{\mathcal{C}}(\mathbf{u}) \qquad (6)$$

It is critical that the sample points used to solve for $f_{\mathcal{C}}$ be defined such that the iso-contour $g_{wyvill} \circ f_{\mathcal{C}}(\mathbf{u}) = v_{iso}$ is coincident with $\mathcal{C}$.

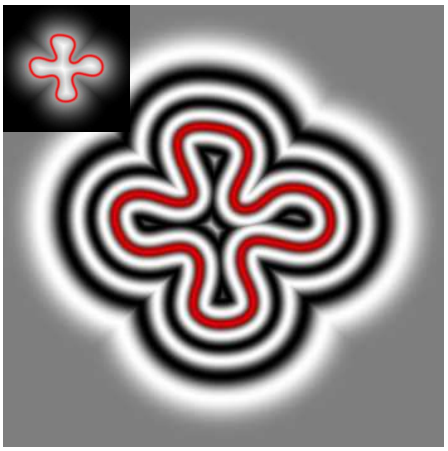

Figure 2. 2D scalar field created using Equation 6. Iso-contours are hilighted using $\sin$ **function before mapping to grayscale. Iso-surface is marked in red. Sharp creases in contours are** C^1 **discontinuities.**

We begin by creating a set of constraint points $(\mathbf{c}_i, s)$ which lie on $\mathcal{C}$. The value s is determined by inverting Equation 3 and evaluating at our desired iso-value, v_{iso}. Unfortunately the miminal-curvature solution for these constraints is a scalar field with constant value s. Additional constraints are necessary to create a psuedo-distance field.

For each point $\mathbf{c}_i$, we add two more constraint points $(\mathbf{c}_i \pm \mathbf{n}_i \cdot \Delta s, s \pm \Delta s)$, where $\mathbf{n}_i$ is the normal to $\mathcal{C}$ at $\mathbf{c}_i$. The value Δs is a small positive distance. We use 0.05, which produces a reasonable approximation to a distance field for many curves. If $\mathcal{C}$ has thin sections a smaller value may be necessary (see Section 4.3). The location of these points is illustrated in Figure 3

The constraint points shown in Figure 3a are not sufficient to guarantee that $f_{\mathcal{C}}$ approximates a distance field at points far from $\mathcal{C}$. Near sharply concave features the field values may decrease away from $\mathcal{C}$. This is due to the curvature-minimization property of Equation 5.

We add a final set of constraint points $(\mathbf{z}_j, z)$, where points $\mathbf{z}_j$ lie on a circle of radius z. Since $\mathcal{C}$ lies in a unit box centered at the origin, we use a radius of 2. These points force $f_{\mathcal{C}}$ to increase away from $\mathcal{C}$, and guarantee that that field will be bounded within the circle of radius 2.

We can now compute the variational solution. The resulting function $f_{\mathcal{C}}$ can be applied in Equation 6 to produce a bounded, C^2 continuous scalar field where the iso-contour v_{iso} lies on $\mathcal{C}$. An example is shown in Figure 4. A useful property of this formulation is that we are not limited to a single closed contour. The constraint points for multiple disjoint contours, including "hole" contours, can be solved simultaneously to produce a single variational field.

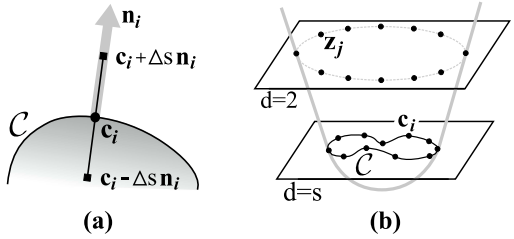

Figure 3. Variational constraints at point $\mathbf{c}_i$ **on contour** $\mathcal{C}$**. (a) shows location of additional constraints along contour normal** $\mathbf{n}_i$**. (b) shows constraints with "distance" as third dimension. Constraints at** $\mathbf{z}_j$ **ensure that the variational solution (gray line) increases away from** $\mathcal{C}$**.**

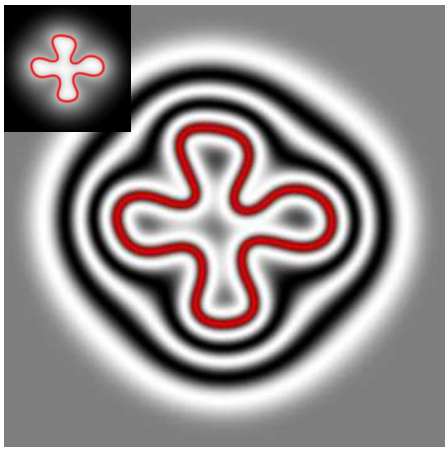

Figure 4. 2D scalar field created using Equation 6. Iso-contours hilighted using $\sin$ **function before mapping to grayscale. Iso-surface is marked in red. Field is globally** C^2 **continuous.**

It is possible to approximate Equation 6 directly with a variational formulation. Although this does produce a bounded field and an iso-contour v_{iso} that lies on $\mathcal{C}$, the zero-crossing of the function generally has a non-zero tangent. This results in a C^1 discontinuity at the outer edge of the field, creating undesirable blending artifacts.

4.3. Limitations of variational approximation

The iso-contour $f_M = v_{iso}$ only approximates the initial contour $\mathcal{C}$. Approximation accuracy depends on the sampling of $\mathcal{C}$ and placement of the normal projection points

(Figure 3a). [CBC*01] and [YT02] treat these issues in depth.

Tangent discontinuities (creases) in $\mathcal{C}$ cannot be accurately reproduced because $f_{\mathcal{C}}$ is globally C^2 continuous. This is an inherent property of variational interpolation. Crease representation can be improved by increasing the curve sampling rate, but this can lead to instability in the variational solution.

It is not strictly guaranteed that $f_{\mathcal{C}}$ increase away from $\mathcal{C}$ in all directions. However, consider Figure 3b. For our technique to fail the grey curve must reverse direction between the planes at $d = s$ and $d = 2$ while still interpolating all points z_j and minimizing global curvature. It may be possible that a long, thin protrusion in $\mathcal{C}$ could result in $f_{\mathcal{C}}$ "escaping" between two z_j's. This situation can be detected by sampling between z_j's and corrected by increasing the sampling density. We note that while using only 15 equally-spaced samples z_j we have yet to encounter this issue.

4.4. Discrete approximation with field images

Evaluating the variational function $f_{\mathcal{C}}$ is $O(N)$ in the number of constraint points used to solve for the variational solution. This results in a field function (Equation 6) that is too expensive for interactive use. The cost can be reduced by discretizing f_M. We create a *field image* by sampling f_M at regular intervals. The field image is essentially a grayscale image, as can be seen in the inset of Figure 4 . Smooth reconstruction filters are then applied to the field image to create an approximation f_M^* to f_M that can be evaluated in constant time.

We implement f_M^* using the Biquadratic reconstruction filter [BMDS02]. Nine samples are necessary to evaluate the Biquadratic filter at $\mathbf{u}$ but the result is C^1 continuous. An alternative is the Bilinear reconstruction filter, which requires only 4 samples and is inexpensive to evalaute, but also produces C^0 discontinuities [BMDS02] across pixel boundaries. All figures in this paper were generated using biquadratic reconstruction applied to field images with a resolution of 128^2. The results are visually indistinguishable from actual evaluation of f_M.

5. Sweep primitives

An implicit sweep primitive is defined by a 2D *sweep template field* f_M (Equation 6) and a 3D *sweep trajectory* $\mathcal{T}$. We will assume that $\mathcal{T}$ is a parametric function:

$$\mathcal{T}(s) = (t_x(s), t_y(s), t_z(s)), \qquad 0 \le s \le 1$$

The sweep surface $\mathcal{S}$ is defined by a 3D scalar field $f_{\mathcal{S}}$. Given a point $\mathcal{T}(s)$ on the sweep trajectory, we can define a geometric transformation $\mathbf{F}(s)$ that maps 2D points $\mathbf{u}$ to 3D points $\mathbf{p}$. We assume that $\mathbf{F}(s)$ is affine and maps the 2D origin to $\mathcal{T}(s)$, implying that points $\mathbf{F}(s) \cdot \mathbf{u}$ are co-planar (Figure 5).

Note that when converting 2D point $\mathbf{u}$ to 3D point $\mathbf{p}$, we append 0 as the third coordinate. Going from 3D to 2D, we simply drop the third coordinate. Application of this conversion will be determined by context and not explicitly denoted.

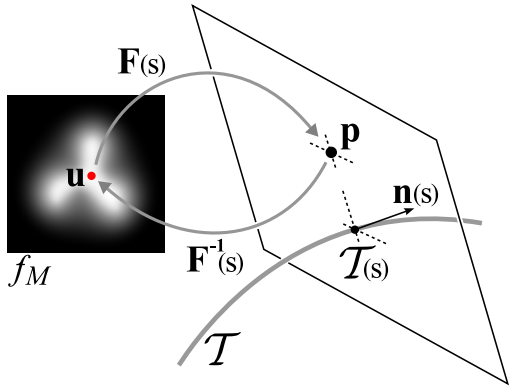

Figure 5. Forward and inverse mapping from sweep template field f_M to 3D space. Function $\mathbf{F}(s)$ maps 2D point $\mathbf{u}$ to 3D point $\mathbf{p}$ lying in plane defined by $\mathcal{T}(s)$ and $\mathbf{n}(s)$. Inverse map $\mathbf{F}^{-1}(s)$ maps from $\mathbf{p}$ to $\mathbf{u}$.

A possible definition for $f_{\mathcal{S}}$ is:

$$f_{\mathcal{S}}(\mathbf{p}) = f_M(\mathbf{u}), \qquad \mathbf{F}(s) \cdot \mathbf{u} = \mathbf{p} \tag{7}$$

Unfortunately, given only a point $\mathbf{p}$ we cannot directly evaluate this equation because the parameter value s and 2D point $\mathbf{u}$ are unknown. Since $\mathbf{F}(s)$ is affine we can compute the inverse mapping $\mathbf{F}^{-1}(s)$:

$$f_{\mathcal{S}}(\mathbf{p}) = f_M(\mathbf{u}), \qquad \mathbf{u} = \mathbf{F}^{-1}(s) \cdot \mathbf{p} \tag{8}$$

However, s is still unknown, and not necessarily unique (Figure 6).

$\mathbf{F}(s)$ transforms the 2D plane into some 3D plane passing through $\mathcal{T}(s)$ with normal $\mathbf{n}(s)$ (Figure 5). Since there may be more than one plane that passes through $\mathbf{p}$ (Figure 6), we must define a set of parameter values $\mathbb{S}(\mathbf{p}) = s_i$ such that $\mathbf{p}$ lies in the plane at s_i:

$$\mathbb{S}(\mathbf{p}) = \{s_i \ : \ (\mathbf{p} - \mathcal{T}(s)) \cdot \mathbf{n}(s) = 0\} \tag{9}$$

To compute the field value $f_{\mathcal{S}}(\mathbf{p})$ we must evaluate Equation 8 for each parameter value $s_i \in \mathbb{S}(\mathbf{p})$. The resulting field values are combined with a composition operator

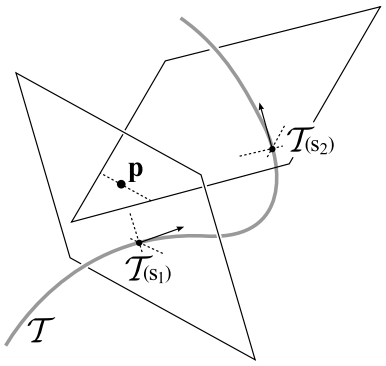

Figure 6. Point p lies on planes at points $\mathcal{T}(s_1)$ and $\mathcal{T}(s_2)$. Unique inverse mapping $\mathbf{F}^{-1}(s)$ does not exist.

$\mathcal{G}$, such as a CSG union operator [Ric73], to produce a single field value. Our final definition:

$$f_{\mathcal{S}}(\mathbf{p}) = \mathcal{G}\left(\{ f_M(\mathbf{F}^{-1}(s_i) \cdot \mathbf{p}) \ : \ s_i \in \mathbb{S}(\mathbf{p}) \} \right) \quad (10)$$

can be evaluated for any trajectory where the parameter set $\mathbb{S}(\mathbf{p})$ is computable.

5.1. Linear Trajectory

A simple and efficient sweep primitive is the linear sweep, or *extrusion* [BDS*03], where the sweep trajectory is a straight line between points **a** and **b**. In this case $\mathbf{F}(s)$ is unique, $s = (\mathbf{p} - \mathbf{a}) \cdot \mathbf{n}$ where **n** is the unit vector $(\mathbf{b} - \mathbf{a})/\|\mathbf{b} - \mathbf{a}\|$. Given two mutually perpendicular vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ which lie in the plane defined by **n**, we define $\mathbf{F}^{-1}_{linear}(s)$:

$$\mathbf{F}^{-1}_{linear}(s) = \mathbf{Rot}\begin{bmatrix}\mathbf{k}_1 & \mathbf{k}_2 & \mathbf{n}\end{bmatrix} \cdot \mathbf{Tr}\left[-(\mathbf{a} + s\mathbf{n})\right] \quad (11)$$

where $\mathbf{Rot}[\mathbf{k}_1\ \mathbf{k}_2\ \mathbf{n}]$ is homogeneous transformation matrix with upper left 3x3 submatrix $[\mathbf{k}_1 \mathbf{k}_2 \mathbf{n}]^\top$ and $\mathbf{Tr}[-(\mathbf{a}+s\mathbf{n})]$ is a homogeneous translation matrix with translation component $-(\mathbf{a} + s\mathbf{n})$ Twisting and scaling can be introduced into the sweep surface by varying $\mathbf{k}_1$ and $\mathbf{k}_2$ along the trajectory.

The linear sweep scalar field f_{linear} is then defined as

$$f_{linear}(\mathbf{p}) = f_M\left(\mathbf{F}^{-1}_{linear}(s) \cdot \mathbf{p}\right)$$

The function f_{linear} defines an scalar field of infinite extent along **n** which must explicitly be bounded to cap the ends of the sweep surface. The field should extend for some distance d_{endcap} beyond the sweep line endpoints to permit blending. It is desirable to have some control over the endcap shape. Existing implicit sweeps [CBS96] create endcaps by linearly interpolating the sweep template values to zero. We define three alternative endcap styles.

Blobby Endcap A blobby endcap is created by evaluating f_{linear} past the end of the sweep line and scaling it down to zero. We use g_{wyvill} (Equation 3) to smoothly terminate the field. A parameter d_{endcap} determines the extent of the field beyond the end of the sweep line. If the parameter value at the line endpoint is s_{max}, we evaluate:

$$f_{endcap}(\mathbf{p}) = g_{wyvill}\left(\frac{|s| - s_{max}}{d_{endcap}}\right) \cdot f_{linear}(\mathbf{p}) \quad (12)$$

Field continuity is preserved by the zero tangents of g_{wyvill}. The endcap width varies (Figure 7) because f_{linear} has increasing values inside the sweep contour.

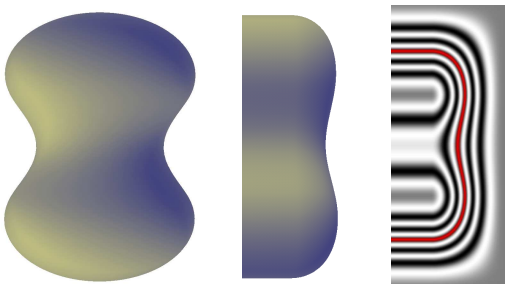

Figure 7. Blobby endcap style. Sweep profile (left) determines width of endcap at each point (middle). Width is dependent on field value inside the profile (right). Isocontour is marked in red.

Flat Endcap With Smooth Transition [BWdG05] defines hybrid CSG/blending operators that smooth out the CSG transition zone but are otherwise identical to sharp CSG operators. We perform an intersection with an infinite plane to smoothly transition from the sweep surface to a flat endcap. The size of the transition zone is controllable, Figure 8 shows endcaps with varying parameter values. This operator converges to a C^1 field discontinuity as the transition zone size decreases. Note that to preserve field continuity the operator should be applied to the entire sweep. We intersect with a constant field value of 1 in the sweep region.

Flat Endcap With Sharp Transition [BWdG05] defines another type of CSG operator that produces the traditional CSG surface but preserves C^1 continuity in the field away from the surface. In this case intersection with an infinite plane creates a sharp transition that preserves field continuity. We intersect with a constant field value of 1 in the sweep region to preserve continuity. Figure 9 compares the field created with this operator and the intersection operator of [Ric73], which creates C^1 discontinuities.

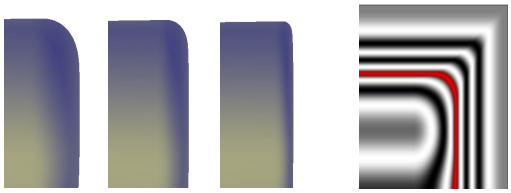

Figure 8. Flat endcap style with smooth transition. Transition parameter varies from left to right $30°$, $35°$, $40°$. Rightmost image shows scalar field for $35°$ parameter. Isocontour is marked in red.

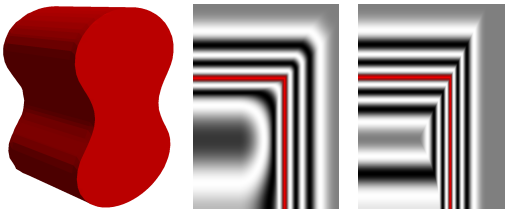

Figure 9. Flat endcap with sharp transition. C^1 discontinuity on surface produces sharp crease (left) but does not propagate away from the surface with Barthe CSG intersection (middle). Ricci intersection produces C^1 discontinuity away from surface (right). Isocontour is marked in red.

5.2. Cubic Bezier Trajectory

As an example of a general trajectory we consider a 3D cubic Bezier curve $\mathcal{B}(s)$, $s \in [0,1]$. Assume that the normal function $\mathbf{n}(s)$ (Equation 9) is the tangent vector $\mathcal{B}'(s)$. The following polynomial in s must be solved to find the parameter value set $\mathbb{S}(\mathbf{p})$:

$$(\mathbf{p} - \mathcal{B}(s)) \cdot \mathcal{B}'(s) = 0 \tag{13}$$

This polynomial is degree 5 for a cubic Bezier curve, precluding analytic solution. Numerical root-finding techniques such as [Sch90] can be used to solve for the roots s.

Once $\mathbb{S}(\mathbf{p})$ is computed we can evaluate $\mathbf{F}^{-1}_{bezier}(s)$:

$$\mathbf{F}^{-1}_{bezier}(s) = \mathbf{Rot}\begin{bmatrix}\mathbf{k}_1 & \mathbf{k}_2 & \mathcal{B}'(s)\end{bmatrix} \cdot \mathbf{Tr}\begin{bmatrix}-\mathcal{B}(s)\end{bmatrix}$$

Care needs be taken when defining the vectors $\mathbf{k}_1$ and $\mathbf{k}_2$. One option is to calculate the *Frenet frame* [FvDF*93]:

$$\mathbf{k}_1 = \mathcal{B}'(s) \times \mathcal{B}''(s)$$
$$\mathbf{k}_2 = \mathbf{k}_1 \times \mathcal{B}'(s)$$

However $\mathcal{B}''(s) = \mathbf{0}$ in degenerate cases such as straight sections. The Frenet frame can also flip direction across points of inflection, creating C^0 discontinuities. An alternative is the rotation minimizing frame of [Blo90]. This frame is procedurally defined based on an initial frame at $\mathcal{B}(0)$ and cannot be calculated analytically. In order to generate a frame at any s, a table of rotation minimizing frames $\mathbf{F}_j$ can be calculated at points s_j along the curve. $\mathbf{F}^{-1}_{bezier}(s)$ is then found by locating the nearest $s_j < s$ and computing a rotation minimizing frame from $\mathbf{F}_j$.

Open trajectories should be capped. The cap styles described in Section 5.1 are all applicable to arbitrary trajectories and are applied in the same way. Twisting and scaling can also be applied based on s or an arc-length parameterization of $\mathcal{B}(s)$.

The implicit surface behavior in self-intersecting cases, which frequently occur with curving trajectories, is entirely determined by the operator $\mathcal{G}$ (Equation 10). $\mathcal{G}$ should be applied to all field values produced with the solutions to Equation 13, as well as the endcap field values. The CSG Union operator of [Ric73] produces a closed manifold surface but also C^1 discontinuities in the scalar field (Figure 10). Barthe's CSG Union operator with sharp transition [BWdG05] can be used to preserve both the surface creases and gradient continuity in the field away from the surface.

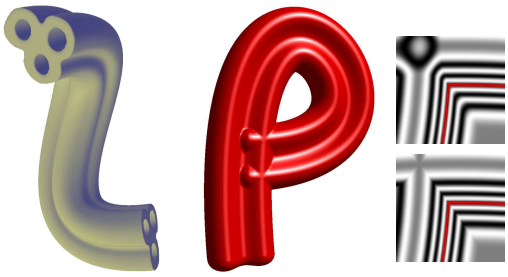

Figure 10. Bezier curve sweeps. Holes in template are supported (left). Scalar field in self-intersecting cases (middle) is C^1 continuous using Barthe CSG union operator (right top). Ricci union operator produces C^1 discontinuities (right bottom).

5.3. Circular Trajectory

Circular sweeping trajectories, described by a point $\mathbf{o}$ and a normalized axis $\mathbf{n}$, result in *surfaces of revolution*. Several restrictions are necessary to provide an implicit circular sweep definition that is valid for any trajectory and sweep template. The sweep function, Equation 10, cannot

be evaluated at points lying on the axis $\mathbf{n}$ because the set $\mathbb{S}(\mathbf{p})$ (Equation 9) is infinite. To define circular sweeps we make a simplifying assumption - that the template orientation and scaling is constant. Under this constraint the infinite parameter set maps to a single point $\mathbf{u}$.

Since we disallow twisting and scaling, it is possible to compute point $\mathbf{u}$ in the sweep template corresponding to a point $\mathbf{p}$ without finding the angle θ on the circular trajectory. The 2D coordinates are found geometrically:

$$\mathbf{u}_y = (\mathbf{p} - \mathbf{o}) \cdot \mathbf{n}$$
$$\mathbf{u}_x = ||(\mathbf{p} - \mathbf{o}) - \mathbf{u}_y \mathbf{n}||$$

The 2D coordinates corresponding to the opposing side of the circle are $(-\mathbf{u}_x, \mathbf{u}_u)$. These sample points result in two field values which we compose with the Barthe Union operator [BWdG05] to produce a gradient-continuous scalar field.

To support partial circular sweeps the parameter θ must be computed. The necessary equations can be found in [WC02]. Our endcap techniques should be applied to partial sweeps to prevent C^0 discontinuity when the sweep terminates.

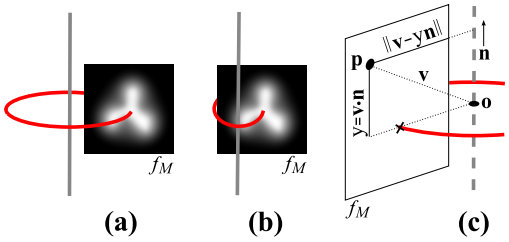

Figure 11. Invertible sweeps produce torus-like surfaces (a), while self-intersecting sweeps (b) are non-invertible because degenerate points on n lie inside the field bounds. Geometric computation of 2D sweep template coordinates is shown in (c).

Note that $\mathbf{u}_x \geq 0$, implying that the portion of the sweep template to the left of the axis in Figure 11b is never sampled. This can be desirable as it is analogous to revolving an open curve around an axis between the curve endpoints. Unfortunately it also produces a C^1 discontinuity at points on $\mathbf{n}$ where f_M is not continuous C^1 continuous across the Y axis.

We also note that in certain cases (Figure 11a) field values along $\mathbf{n}$ are outside the bounds of f_M and therefore 0. Here twisting can be safely applied, although for full circular sweeps the twist angle needs be a whole multiple of 2π to avoid C^0 discontinuities. A similar limitation applies for scaling.

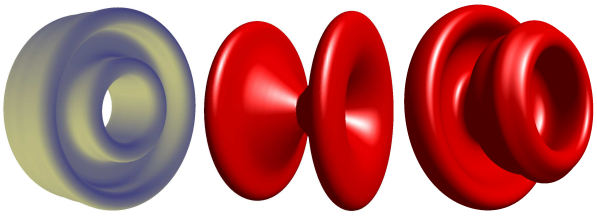

Figure 12. Circular trajectory sweep surfaces.

6. Applications

6.1. Interactive BlobTree modeling

The sweep primitives described in Section 5 can be used directly in implicit modeling systems such as the *BlobTree* [WGG99]. Sweep primitives implemented using the discrete approximation technique of section 4.4 are very efficient. We use them in an interactive BlobTree modeling tool.

A key limitation for interactive implicit modeling is visualizing the current surface. We visualize BlobTree models by generating a polygonal mesh that approximates the surface [Blo94]. The main cost of polygonization is evaluating the field function. Linear and circular sweeps can be polygonized at resolutions adequate for modeling, provided the sweep contour does not have very small features. Performance quickly degrades when composing several sweeps in a BlobTree. However, with hierarchical spatial caching [SWG05] a large number of implicit sweep primitives can be manipulated interactively, including bezier curve sweeps. The models in Figures 13 and 16 were all created using our interactive system. We note that zero-length linear sweeps with blobby endcaps are very useful for character modeling, as shown in Figure 16.

We note that while solving the linear system defined by Equation 5 is $O(N^3)$, the number of constraint points is usually less than one thousand. We use an SIMD-optimized LAPACK implementation that solves these systems with double precision in a few seconds on a 1.6Ghz Pentium 4 processor.

6.2. Parametric to Implicit sweep conversion

Many existing parametric modeling tools define sweep surfaces using a boundary representation. In this case the sweep template is the contour $\mathcal{C}$, usually defined parametrically, and the sweep trajectory is a 3D parametric curve $\mathcal{T}(s)$. The boundary representation has several limitations. Operations such as CSG, or even simple point-inclusion testing, require complex numerical techniques.

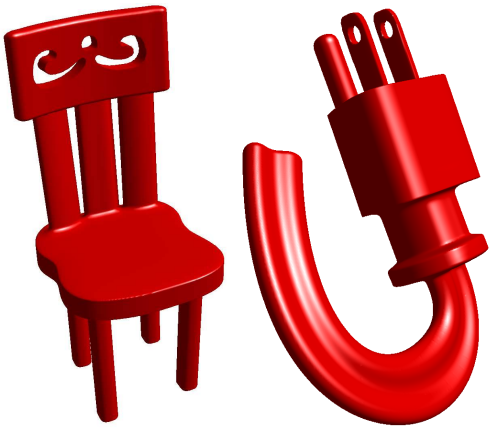

Figure 13. Implicit BlobTree models composed entirely of sweeps, blending, and CSG. Each model was created interactively in under 30 minutes.

Self-intersecting sweeps are difficult to handle [AMBJ00] because the definition of "inside" is ambiguous. Algorithms for measuring physical properties, such as volume, often produce nonsense results in these degenerate cases.

Our implicit sweep representation easily handles degenerate conditions, CSG is well-supported [Ric73] [WGG99] [BWdG05], and point-inclusion testing is trivial using the implicit volume definition (Equation 4). Conversion from parametric sweeps to our implicit sweeps is trivial, we simply re-use the parametric contour contour $\mathcal{C}$ and trajectory $\mathcal{T}$ (Section 5).

An important consideration when converting parametric sweeps to implicit sweeps is error introduced by the conversion. The 2D scalar field is produced using variational techniques that only approximate $\mathcal{C}$ (Section 4.3). However, by increasing the curve sampling rate when solving for $f_{\mathcal{C}}$ (Section 4.2) the variational solution can be tightly constrained. An example of parametric conversion is shown in Figure 14. In this case the maximum surface deviation is less than 0.5%, which occurs at the high-curvature features.

Interactive tools for implicit modeling can take advantage of this coherence between parametric and implicit representations. The sweep surface can be quickly visualized during interactive manipulation using parametric techniques. Once manipulation is complete, the implicit model is re-polygonized.

6.3. Surface reconstruction from parallel contours

It is possible to interpolate between the two sweep template fields f_{M_1} and f_{M_2} along a sweep with respect to a parameter $t \in [0, 1]$:

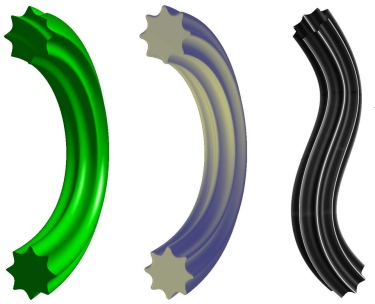

Figure 14. Parametric sweep (left) and corresponding implicit sweep (middle). Parametric sweep vertices colored by implicit approximation error (right). Error values have been scaled by 1000**.**

$$f_{M_{interp}}(t) = (1-t)f_{M_1} + (t)f_{M_2}$$

A sweep surface with varying cross section can be created by sweeping the template $f_{M_{interp}}(s)$, where s is scaled to $[0, 1]$. The resulting surface smoothly transitions between the two templates regardless of topology. Interpolation between n templates is accomplished by successively interpolating between pairs of templates. Given a set of contours $\mathcal{C}_i$, a closed surface can be reconstructed that passes through all the contours.

A variety of techniques for implicit surface reconstruction from contours are available [SPOK95] [AG04] [CBC*01] [YT02]. One benefit of our approach is that the resulting field is very efficient to evaluate. In Figure 15 a human L3 vertebra is reconstructed from 93 segmented CT slices (See also Figure 18) . The reconstruction process takes approximately 6 minutes, however once complete the implicit object (which is simply a linear sweep) can be interactively edited in our BlobTree system

7. Conclusion

We have presented new C^1 continuous implicit sweep objects that have a bounded scalar field. Our sweep template fields can be generated from any set of closed 2D contours, including contours with "holes". Implicit sweep objects generated with our template fields are visually indistinguishable from triangle meshes created by sweeping the same contour, even when using a C^1 discrete approximation to the 2D scalar field. Our endcap styles for open trajectories preserve field continuity, as do the CSG union operators applied for self-intersecting sweeps.

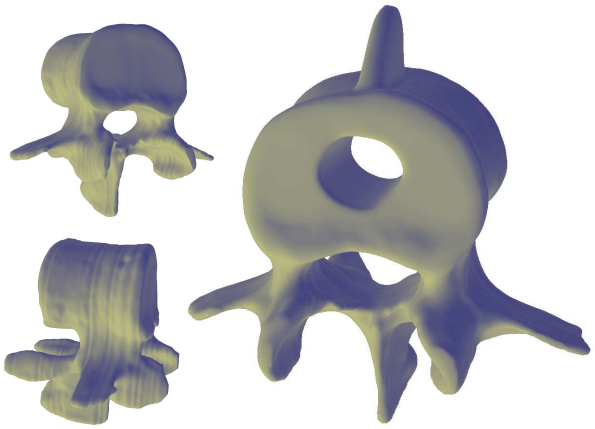

Figure 15. Surface reconstruction from contours. Human L3 vertebra is reconstructed from 93 slices (left) and interactively modified in a BlobTree modeling system (right). High-frequency ridges on vertebra surface are due to manual segmentation errors.

These contributions permit direct specification and manipulation of the implicit surface for a large class of free-form shapes. Direct surface interaction is desirable for shape modeling, but has not been available for implicit surfaces with bounded fields. Bounded fields are necessary to achieve scalable interactive performance [SWG05] and preserve the "principle of least surprise" that is key for interactive tools.

We have found that we rely on these sweep primitives almost exclusively when using our interactive BlobTree modeling tool. However the traditional modeling interface is very limited when working with complex hierarchical models. One avenue for future research is the development of new interface metaphors and tools appropriate for interactive BlobTree modeling.

We have performed only very limited exploration into parametric to implicit sweep conversion. Though the surface error in our tests was less than 0.5%, this may still be outside of the acceptable tolerances for CAD/CAM applications. A direction for future work would be to represent the 2D contours exactly by integrating the functional curve representation of [PSS96a] with the improved operators of [BDS*03].

Finally, we have presented some basic results on surface reconstruction from parallel contours. Implicit techniques for this problem have been previously developed. The benefit of our method is that the reconstructed model can be imported directly into our BlobTree modeling tool. This provides another avenue to leverage existing models. The case of medical data is particularly interesting - the interactivity afforded by our approach may have significant applications in surgical training simulators.

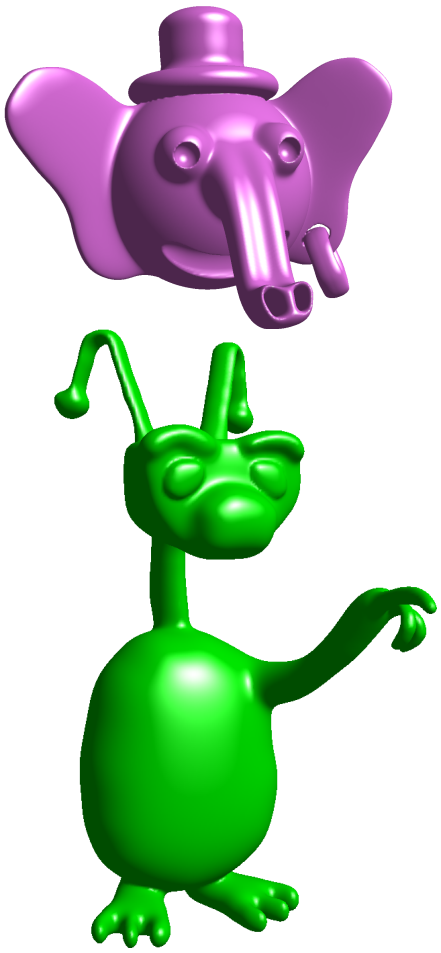

Figure 16. Implicit BlobTree models composed mostly of zero-length linear sweeps with blobby endcaps. Top model also uses Bezier sweep for nose and revolution for hat.

References

[AG04] AKKOUCHE S., GALIN E.: Implicit surface reconstruction from contours. *The Visual Computer 20*, 6 (2004), 380–391.

[AMBJ00] ABDEL-MALEK K., BLACKMORE D., JOY K.: Swept volumes: Foundations, perspectives and applications. *submitted to International Journal of Shape Modeling* (2000).

[AMYB00] ABDEL-MALEK K., YANG J., BLACKMORE D.: Closed-form swept volume of implicit surfaces. In

Proceedings of ASME Design Engineering Technical Conferences (2000).

[BDS*03] BARTHE L., DODGSON N. A., SABIN M. A., WYVILL B., GAILDRAT V.: Two-dimensional potential fields for advanced implicit modeling operators. *Computer Graphics Forum 22*, 1 (2003), 23–33.

[Blo90] BLOOMENTHAL J.: *Graphics Gems*. Morgan Kaufmann, 1990, ch. Calculation of references frames along a space curve, pp. 567–571.

[Blo94] BLOOMENTHAL J.: *Graphics Gems IV*. Academic Press Professional Inc., 1994, ch. An implicit surface polygonizer, pp. 324–349.

[Blo97] BLOOMENTHAL J. (Ed.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997.

[BMDS02] BARTHE L., MORA B., DODGSON N., SABIN M.: Interactive implicit modelling based on c^1 reconstruction of regular grids. *International Journal of Shape Modeling 8*, 2 (2002), 99–117.

[BWdG05] BARTHE L., WYVILL B., DE GROOT E.: Controllable binary csg operators for soft objects. *International Journal of Shape Modeling* (2005). (To Appear).

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3d objects with radial basis functions. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 67–76.

[CBS96] CRESPIN B., BLANC C., SCHLICK C.: Implicit sweep objects. *Computer Graphics Forum 15*, 3 (Aug. 1996), 165–174.

[Duc77] DUCHON J.: *Constructive Theory of Functions of Several Variables*. Springer-Verlag, 1977, ch. Splines minimizing rotation-invariant semi-norms in Sobolev spaces, pp. 85–100.

[FvDF*93] FOLEY J. D., VAN DAM A., FEINER S. K., HUGHES J. F., PHILLIPS R.: *Introduction to Computer Graphics*. Addison-Wesley, 1993.

[Gri99] GRIMM C.: Implicit generalized cylinders using profile curves. In *Implicit Surfaces 99* (1999), pp. 33–41.

[PSS96a] PASKO A., SAVCHENKO A., SAVCHENKO V.: *Implicit curved polygons*. Tech. Rep. University of Aizu, Japan, Technical Report 96-1-004, 1996.

[PSS96b] PASKO A., SAVCHENKO A., SAVCHENKO V.: Polygon-to-function conversion for sweeping. In *Proceedings of Implicit Surfaces 96* (1996), pp. 163–171.

[Req80] REQUICHA A. A. G.: Representations for rigid solids: theory, methods and systems. *Computing Surveys 12*, 4 (1980), 437–464.

[Ric73] RICCI A.: A constructive geometry for computer graphics. *Computer Graphics Journal 16*, 2 (1973), 157–160.

[RV82] REQUICHA A. A. G., VOELCKER H. B.: Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics and Applications 2*, 2 (1982), 9–24.

[Sch90] SCHNEIDER P. J.: *Graphics Gems*. Morgan Kaufmann, 1990, ch. A Bezier curve-based root finder, pp. 409–415.

[SP96] SOURIN A., PASKO A.: Function representation for sweeping by a moving solid. *IEEE Transactions on Visualization and Computer Graphics 2*, 1 (1996), 11–18.

[SPOK95] SAVCHENKO V., PASKO A., OKUNEV O., KUNII T.: Function representation of solids reconstructed from scattered surface poitns and contours. *Computer Graphics Forum 14*, 4 (1995), 181–188.

[SW97] SEALY G., WYVILL G.: Representing and rendering sweep objects using volume models. In *Computer Graphics International 1997* (June 1997), pp. 22–27.

[SWG05] SCHMIDT R., WYVILL B., GALIN E.: *Interactive implicit modeling with hierarchical spatial caching*. Tech. Rep. 2005-770-01, University of Calgary, 2005.

[WC02] WINTER A. S., CHEN M.: Image-swept volumes. *Computer Graphics Forum 21*, 3 (2002), 441–450.

[WGG99] WYVILL B., GUY A., GALIN E.: Extending the csg tree. warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum 18*, 2 (1999), 149–158.

[Wyv] WYVILL G.: Wyvill function. personal communication.

[YT02] YNGVE G., TURK G.: Robust creation of implicit surfaces from polygonal meshes. *IEEE Transactions on Visualization and Computer Graphics 8*, 4 (2002), 346–359.

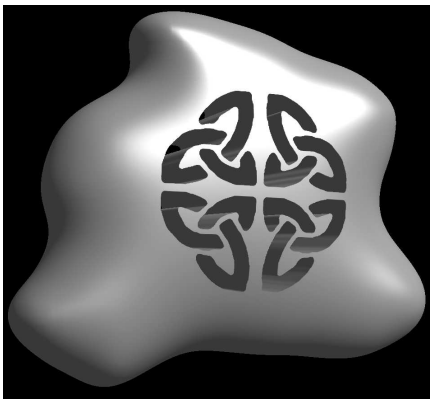

Figure 17. Implicit sweep subtracted from blobby object. Sweep contours are specified by bitmap image.

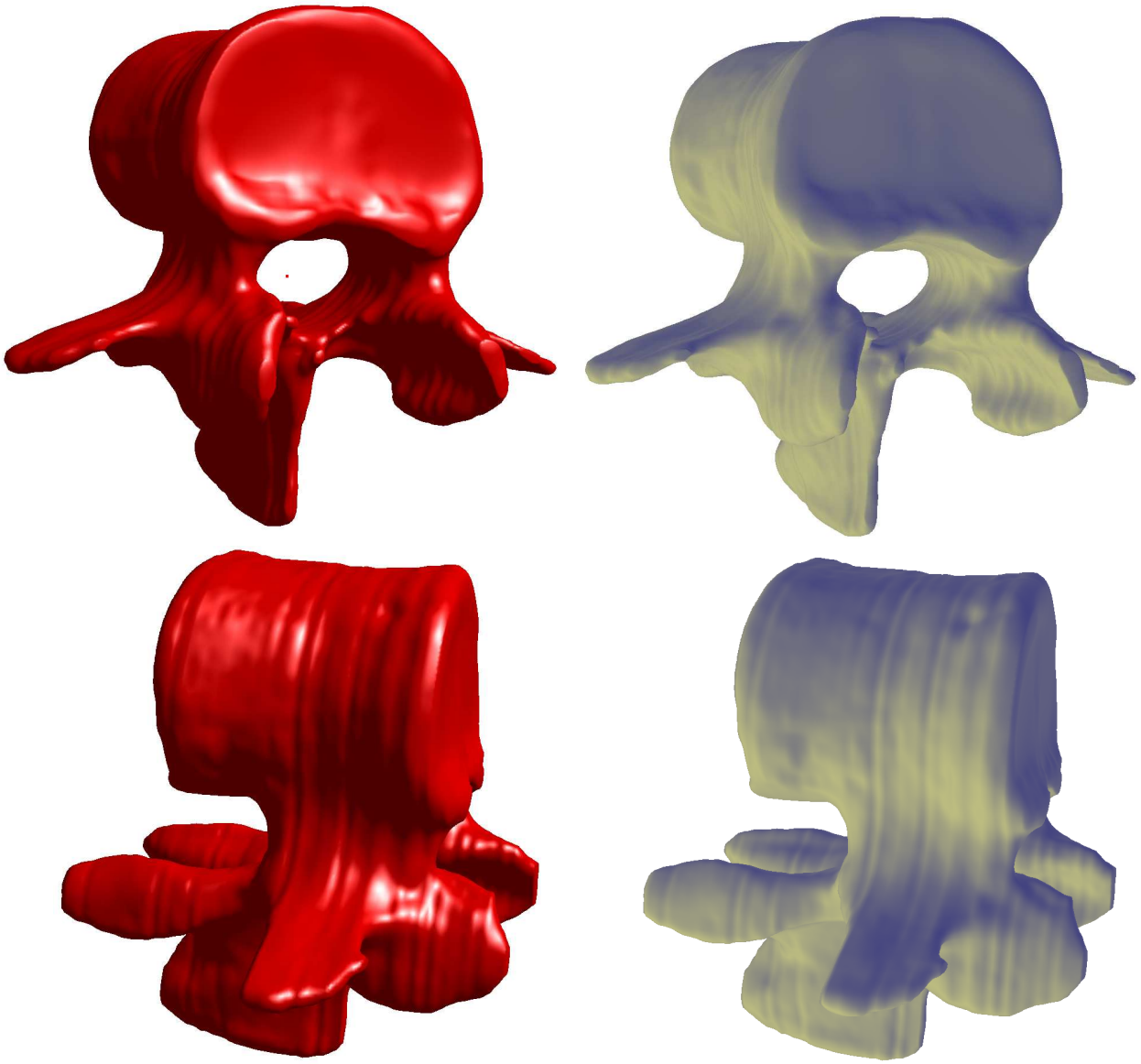

Figure 18. Various views of Human L3 Vertebra reconstructed by interpolating between planar implicit sweeps.