EUROGRAPHICS Workshop on Sketch-Based Interfaces and Modeling (2005)
Takeo Igarashi, Joaquim A. Jorge (Editors)

ShapeShop: Sketch-Based Solid Modeling with BlobTrees

R. Schmidt[1], B. Wyvill[1], M. C. Sousa[1], J. A. Jorge[2]

[1]Dept. Computer Science
University of Calgary, Canada

[2] Dept. Information Systems and Computer Engineering
TU Lisbon, Portugal

Abstract

Various systems have explored the idea of inferring 3D models from sketched 2D outlines. In all of these systems the underlying modeling methodology limits the complexity of models that can be created interactively. The ShapeShop sketch-based modeling system utilizes Hierarchical Implicit Volume Models (BlobTrees) as an underlying shape representation. The BlobTree framework supports interactive creation of complex, detailed solid models with arbitrary topology. A new technique is described for inflating 2D contours into rounded three-dimensional implicit volumes. Sketch-based modeling operations are defined that combine these basic shapes using standard blending and CSG operators. Since the underlying volume hierarchy is by definition a construction history, individual sketched components can be non-linearly edited and removed. For example, holes can be interactively dragged through a shape. ShapeShop also provides 2D drawing assistance using a new curve-sketching system based on variational contours. A wide range of models can be sketched with ShapeShop, from cartoon-like characters to detailed mechanical parts. Examples are shown which demonstrate significantly higher model complexity than existing systems.

1. Introduction

A variety of underlying shape representations have been used in sketch-based free-form modeling systems, including triangle meshes [IMT99], subdivision surfaces [IH03], variational implicit surfaces [KHR02][AJ03], convolution surfaces [TZF04], spherical implicit functions [AGB04], and discrete volume data sets [ONNI03]. A common attribute of these systems is that the underlying shape representation heavily influences the sketch-based modeling operations that are implemented. For example, supporting automatic blending with triangle meshes is relatively complex, compared to implicit representations. These issues tend to limit prototype sketch-based modeling systems to operations that are practical to implement, which in turn restricts the types of models that can be sketched by the intended users.

None of the existing systems have been shown to support creation of complex models while retaining interactive performance. Again, the underlying shape representation can fundamentally restrict scalability. For example, variational implicit surfaces [KHR02][AJ03] are generated by solving a large matrix, which is not feasible in real-time except for relatively simple models.

In an attempt to mitigate these issues, we propose Hierarchical Implicit Volume Models (BlobTrees) [WGG99] as an underlying shape representation for sketch-based free-form modeling. A BlobTree procedurally defines an implicit volume using a tree of basic volumes (primitives) and composition operators, such as CSG and blending. Inside this framework, shape-modeling operations such as hole-cutting are easy to implement. The underlying model tree is also a construction history which supports non-linear editing of the model. Using a hierarchical spatial caching scheme [SWG05], complex models can be constructed and manipulated interactively.

We describe *ShapeShop*, a sketch-based 3D BlobTree modeling system in the style of Teddy [IMT99]. ShapeShop includes several sketch-based operations for hole cutting, oversketched blending, and adding surface detail (Section 3). We also introduce a technique for assisting the user with sketching smooth 2D curves, and describe some other gestural interface tools (Section 4).

Traditionally, BlobTree systems have used *skeletal primitives*, essentially offset surfaces from geometric entities. It is non-trivial to define a skeletal primitive such that the offset surface fits a sketched 2D contour [AGB04]. To support sketch-based modeling, we introduce a free-form BlobTree

primitive that closely approximates a closed 2D contour using variational interpolation (Section 5). The surface of the primitive can be defined such that it mimics the "inflation" algorithms of existing sketch-based systems [IMT99].

Finally, we provide several examples of models sketched with ShapeShop that display significantly higher surface complexity than previous systems (Section 7).

2. Related Work

Sketch-based 3D modeling systems can be categorized based on how the system creates 3D shapes in response to user input (sketches). *Suggestive* sketch-based modeling systems attempt to map rough sketches to linear geometry such as lines, planes, and polyhedra. These systems frequently use *expectation lists* which allow the user to resolve ambiguous situations. Examples of these systems include SKETCH [ZHH96], Chateau [IH01], and GiDES++ [JSC03].

In contrast, *Literal* sketch-based modeling systems create 3D surfaces directly from user strokes. Examples include Teddy [IMT99], BlobMaker [AJ03], and ConvMo [TZF04]. A fundamental operation in these systems is *inflation*, where user-sketched closed 2D contours become the 3D silhouettes of rounded shapes. Various systems support different sketch-based editing operations on inflated surfaces, including extrusion, cutting, and blending. These systems are frequently classified as *free-form* modeling tools. Our system falls into the Literal sketch-based modeling category.

The Teddy system [IMT99] pioneered the free-form sketch-based modeling concept. Closed triangle meshes were created by inflating user-sketched 2D contours using the *chordal axis* of the 2D polygon. Sketch-based extrusion, cutting, and smoothing operations were supported. Further work produced smoother results by re-meshing the surface based on local quadratic implicit surface approximation [IH03] [MCCH99]. The system was limited to models with spherical topology (genus 0) and low surface complexity.

A recent work by Cherlin et al. [CSSJ05] implements sketch-based modeling using interpolating parametric surfaces. A wide variety of shapes are created using a novel generalized surface-of-revolution scheme. No composition or grouping operations are supported, each surface is independent. The system can scale to a large number of individual surfaces, however each must be manually positioned to give the impression of a solid 3D model. While complex models can be created, the authors note that the requisite manual positioning is very time consuming.

Several attempts have been made to improve on Teddy using implicit surfaces. Variational implicit surfaces were used by Karpenko et al. [KHR02] and the BlobMaker system [AJ03]. Shape-editing was limited to blending and oversketching. In both cases blending was not procedural, the existing surfaces were replaced with a single combined surface. Karpenko's system did maintain a hierarchy of individual components, but this hierarchy was only used for maintaining spatial relationships. An $O(N^3)$ matrix inversion is necessary to solve for the variational function, limiting the number of constraint points (and hence the surface complexity).

Other implicit-based sketching systems have used convolution surfaces [TZF04] and spherical implicit functions [AGB04]. Neither system supports sharp edges, and in both cases only low-complexity models are shown.

A binary volume data set is used by Owada et al. [ONNI03] in a sketching system based on Teddy. The topological restrictions of Teddy are mitigated by the use of a volumetric representation. Novel methods for sketching internal cavities are presented which allow for more detailed models. This system is fundamentally limited by the resolution of the underlying volume data set.

We note that none of the literal sketch-based modeling systems published to date have been shown to scale to even moderately complex models. The stated goal of these tools is generally to support 3D modeling in the conceptual-design phase, and not replace existing shape modeling tools [IMT99][AJ03][TZF04]. However, it is unclear that low-complexity models can adequately represent the often highly-detailed sketches produced in conceptual design.

Tai et al. [TZF04] classify free-form sketch-based modeling systems as either *boundary-based* or *volume-based*. Of the above systems, only two are boundary-based [IMT99][CSSJ05]. However, only Owada et al.'s system [ONNI03] takes advantage of the benefits provided by a volumetric representation. The implicit-based systems largely ignore the extensive framework provided by hierarchical implicit volume modeling [WGG99], and instead focus on surface-smoothness properties. We address the benefits provided by integration of these concepts into a sketch-based modeling system in the following sections.

3. Sketch-Based Modeling Operations

We support construction of three types of surfaces based on sketches - "blobby" inflation in the style of Teddy, linear sweeps, and surfaces of revolution. Based on these three shapes, sketch-based cutting and blending operations are implemented using BlobTree composition operators.

A key benefit of BlobTrees is that the current volume is procedurally defined by the underlying model tree (Section 5.1). This tree represents both a scene graph and a full construction history. Single primitives, as well as entire portions of the tree, can be modified or removed at any time. This flexibility is exposed in ShapeShop mainly via gestural commands and 3D widgets (Section 4.4). However, we also implement a sketch-based resize operation that takes advantage of the BlobTree hierarchy.

3.1. Blobby inflation

A closed 2D contour can be inflated into a "blobby" shape using the technique described in Section 5.2. The 2D sketch (Figure 1a) is projected onto a plane through the origin parallel to the current view plane, and then inflated in both directions (Figure 1b). After creation, the width of the primitive can be manipulated interactively with a slider (Figure 1c). The inflation width is functionally defined and could be manipulated to provide a larger difference between thick and thin sections. One advantage of an implicit representation is that holes and disjoint pieces can be handled transparently.

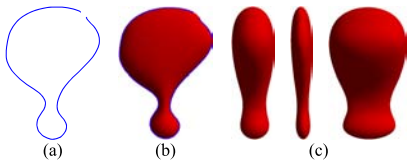

Figure 1: *Blobby inflation converts the 2D sketch shown in (a) into the 3D surface (b) such that the 2D sketch lies on the 3D silhouette. The width of the inflated surface can be manipulated interactively, shown in (c).*

3.2. Sweep surfaces

Our blobby inflation scheme is based on an underlying sweep surface representation which also supports linear sweeps (Figure 2a) and surfaces of revolution (Figure 2b). Linear sweeps are created in the same way as blobby shapes, with the sweep axis perpendicular to the view-parallel plane. The initial length of the sweep is proportional to the screen area covered by the bounding box of the 2D curve, but can be interactively manipulated with a slider. Surfaces of revolution are created by revolving the sketch around an axis lying in the view-parallel plane. Revolutions with both spherical and toroidal topology can be created.

Existing sketch-based systems have generally not included these types of shapes, with the exception of [CSSJ05]. However, we have found them invaluable. Surfaces of revolution are a class of shape that cannot be reproduced with blobby inflation.

Figure 2: *Sketched 2D curves can also be used to create (a) linear sweeps and (b) surfaces of revolution.*

3.3. Cutting

Since our underlying shape representation is a true volume model, cutting operations can be easily implemented using CSG operators. Users can either cut a hole through the object or remove volume by cutting across the object silhouette. Once a hole is created the user may transform the hole interactively. We provide a slider control to modify the depth of cutting operations. Cut regions are represented internally as linear sweeps, no additional implementation is necessary to support cutting in the BlobTree. As example is shown in Figure 3. This CSG-based cutting operation is both more precise and less restrictive than in existing systems.

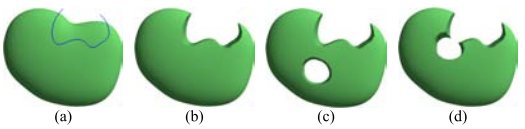

Figure 3: *Cutting can be performed (b) across the object silhoutte or (c) through the object interior. Holes can be interactively translated and rotated. Intersection with other holes is automatically handled, as shown in (d).*

3.4. Blending

We allow the user to blend new blobby primitives to the current volume via oversketching. To position the new blobby primitive, we intersect rays through the sketch vertices with the current implicit volume. The new primitive is centered at the average z-depth of the intersection points. The width of the new blobby primitive can be manipulated with a slider, as can the amount of blending. Blended volumes can be transformed interactively, an example is shown in Figure 4. Karpenko et al [KHR02] supported sketching of the blend profile but also noted that this technique was not robust and is very slow to compute. The level of interactive control over the blend surface in our system has not been previously available.

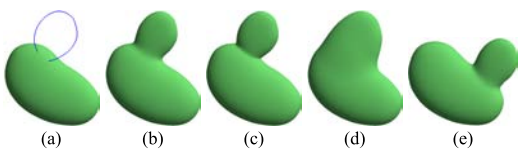

Figure 4: *The sketch-based blending operation (a) creates a new blobby inflation primitive (b) and blends it to the current volume. The blending strength is parameterized and can be interactively manipulated, the extreme settings are shown in (c) and (d). The blend region is recomputed automatically when the blended primitives move, as shown in (e).*

3.5. Surface drawing

Any BlobTree primitive can be used to add surface detail based on sketches. As an initial experiment, ShapeShop supports "surface-drawing". Rays through the 2D sketch are intersected with the current implicit volume. Point primitives, which produce spherical volumes, are placed at intersection points and blended together. Slider controls are provided to manipulate the radius of the point primitives. Results are shown in Figure 5. We are developing a more robust technique involving 3D sweep primitives passing through the intersection points.

Figure 5: *Surface-drawing is specified by a 2D sketch, as shown in (a). Blended skeletal implicit point primitives are placed along the line at intersection points with the model, shown in (b). In (c) the radius of the points is increased and then tapered along the length of the 2D curve.*

Surface drawing with implicit volumes is a very flexible technique. Any pair of implicit primitive and composition operator can be used as a type of "brush" to add detail to the current surface. For example, creases could be created by subtracting swept cone primitives using CSG operations. Implementing these alternative tools is trivial. In addition, since each surface-drawing stroke is represented independently in the model hierarchy, individual surface details can be modifed or removed using our existing modeling interface.

3.6. Sketch-Based Sweep Manipulation

We provide a sketch-based mechanism for resizing and repositioning linear sweeps and blobby shapes, similar to the method used in the SKETCH system [ZHH96]. The user selects a sweep primitive and rotates the view such that the sweep axis is perpendicular to the view direction. The user then draws a straight line which determines the new extents of the shape. Holes can be manipulated with this technique as well, since they are created using linear sweeps (Figure 6). This operation largely eliminates the need for slider widgets to control sweep length and blobby inflation width, except when very fine-grained manipulation is desired.

4. Modeling Interface

Our sketch-based modeling interface has been designed primarily to support use on large interactive displays, such as the touch-sensitive SmartBoard (Figure 7). These input systems lack any sort of modal switch (buttons). In some sense this is desirable, as pencils also lack buttons. However, tasks commonly initiated with mode-switching (such as keypresses or right mouse buttons) must be converted to alternate schemes, such as gestures or 2D widgets.

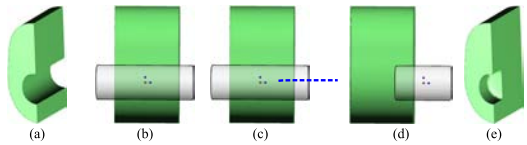

Figure 6: *The linear sweep volume subtracted from (a) is hilighted in (b). By drawing a straight-line stroke parallel to the sweep axis (c), the sweep can be repositioned and resized (d). The new surface is shown in (e).*

Since many 2D widgets can be difficult to use with large-display input devices (which frequently exhibit low accuracy and high latency), we borrow the stroke-based widget interaction techniques of CrossY [AG04]. For example, a button is "pressed" by drawing a stroke that crosses the buttton.

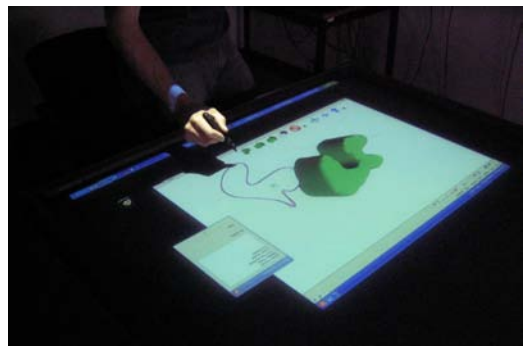

Figure 7: *Our sketch-modeling interface is designed to support non-modal input devices, like this touch-sensitive horizontal tabletop display.*

4.1. 2D Sketch Editing

Two-dimensional sketches form the basis for 3D shape creation in ShapeShop. We have implemented a 2D sketching system that assists with the creation of smooth 2D contours. This system is related to the *interactive beautification* techniques used in the Pegasus system [IMKT97]. Due to space constraints, we will only provide a high-level overview of these techniques.

A fundamental limitation of most standard input devices is that they provide only point samples to the operating system. This discrete data can be converted to a poly-line by connecting temporally-adjacent point samples. However, in the case of curves the poly-line is only an approximation to the smooth curve the user desires. In our system

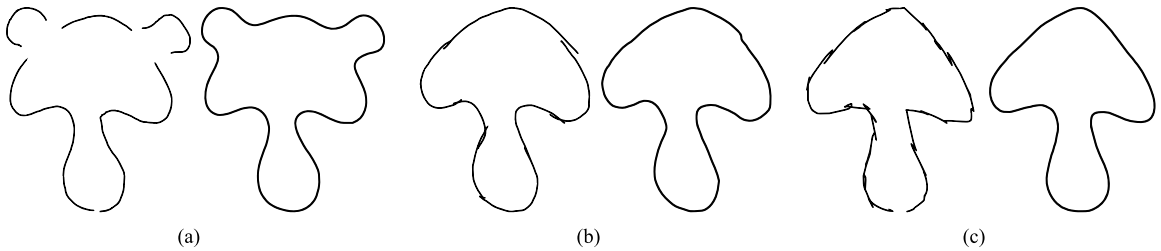

Figure 8: *The gap-filling and smoothing properties of variational curves simplify 2D curve sketching. In (a), multiple disjoint strokes are automatically connected by fitting a variational curve to the input samples. In (b), smoothing parameters are used to handle intersections between multiple strokes. Rough sketches with many self-intersections can also be automatically smoothed, as shown in (c).*

we do not create an approximate poly-line, but instead fit a smooth 2D variational implicit curve [TO02] to the discrete samples. Curve normals derived from the discrete poly-line are used to generate the necessary off-curve constraint points [CBC*01]. Variational curves provide many benefits, such as automatic smoothing and gap-closing with minimal curvature (Figure 8).

ShapeShop supports sketch-based editing of the set of point samples, but not the final variational curve. To simultaneously visualize these two different components, we render the current variational curve in black and the point sample poly-line in transparent blue (Figure 9).

We have implemented three gestural commands to assist users when drawing 2D sketches. The first, *eraser*, is initiated with a "scribble", as shown in Figure 9a. An oriented bounding-box is fit to the scribble vertices and used to remove point samples from the current 2D sketch. The variational curve is re-computed using the remaining samples.

Figure 9: *Examples of the* eraser *gesture (a) and* smooth *gesture (b). These gestures manipulate the parameters used to compute the final variational curve (dashed line).*

The second gestural command is *smooth*, initiated by circling the desired smoothing region a minimum of 2 times. Each point sample has a smoothing parameter associated with it which is incremented if the point is contained in the circled region. The variational curve is then re-computed with the new smoothing parameters (Figure 9b). This gesture can be applied multiple times to the same point samples to further smooth the 2D sketch.

Finally, the *pop* gesture is used to manipulate entire 2D sketches. Using the *erase* command to repair large sketching errors is tedious. Hence, we store individual sketches in a stack. The *pop* gesture, which is input as a quick stroke straight to the left, pops the topmost sketch and discards it.

We have found this system to be very effective for creating smooth 2D sketches. This in turn improves the efficiency of 3D modeling, since fewer corrections need to be made to the 3D shape. One current limitation is that sharp creases in the input sketch are lost, since the underlying variational curve is always C^2 continuous. We are developing additional gesture operations to allow specification of creases.

4.2. Expectation Lists

In ShapeShop the user specifies only 2D silhouettes of the desired 3D shapes. Under this constraint there is an unavoidable ambiguity regarding what shape-modeling operation the user intends. For instance, a given 2D contour can always be interpreted as a blobby shape and a linear sweep. One option is to require additional sketches or gestures to resolve this ambiguity. It is unclear that this extra complexity is more efficient than a visual representation. Hence, we have borrowed the *expectation lists* used in various sketch-based modeling systems [IH01][AJ03][FFJ04].

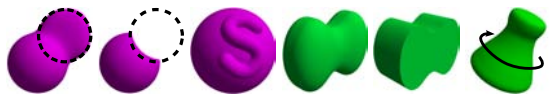

Figure 10: *An example of an expectation list in ShapeShop. The icons denote (from left to right) blend, cut, surface-drawing, blobby inflation, linear sweep, and surface of revolution. The icons are color-coded - green icons create new volumes while magenta icons modify the current volume.*

Existing systems have generally rendered small images of what the updated surface would look like for each expectation list icon. For complex models the user may be required to carefully inspect each image to find the desired action. Instead, We use color-coded iconic representations (Figure 10)

which can be easily distinguished. Mistakes can be quickly corrected by erasing nodes from the BlobTree.

The set of icons displayed in the expectation list is context-dependent. For example, if the user draws a stroke which produces a variational curve that is not closed, no shape-creation icons are displayed. However, in many contexts a single stroke can be interpreted as any sketch action. As the set of operations increases, additional strokes may be necessary to prevent the expectation list from becoming too large.

4.3. Dynamic 3D Clipping

Most of the sketch-based shape editing operations described in Section 3 are based on view-parallel planes and ray-surface intersections. It is frequently the case that the desired editing region is obscured by some other part of the current volume. To deal with this situation we use a dynamic cutting plane. Owada [ONNI03] used a dynamic cutting plane to support sketching of internal volumes. While this is possible in ShapeShop, we have found that the primary use for our dynamic cutting plane is in resolving viewing issues and depth-determination ambiguities.

The cutting plane is initiated by the **L**-shaped *clip* gesture. The user draws a straight line across the surface followed by a small perpendicular "tick" (Figure 11). The initial straight line determines the cutting plane orientation, the tick direction determines which side of the plane to clip. Owada's system kept the "right" side of the line, which we found unintuitive when drawing horizontal lines.

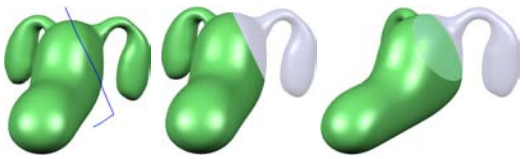

Figure 11: *A temporary cutting plane can assist with sketch-based editing. In (a), the user draws an L-shaped gesture to mark the cutting plane and orientation. Two different views are shown after cutting in (b) and (c).*

4.4. 3D Selection and Transformation

Procedurally-defined BlobTree volumes inherently support non-linear editing of internal tree nodes. However, before a primitive can be manipulated it must be selected. One option is to cast a ray into the set of primitives and select the first-hit primitive. This technique is problematic when dealing with blending surfaces, since the user may click on the visible surface but no primitive is hit.

Instead we implement picking by intersecting a ray with the current volume, then select the primitive which contributes most to the total field value at the intersection point. This algorithm selects the largest contributor in blending situations, and selects the 'hole' primitive when the user clicks on the inside of a hole surface. Since the shape of the selected primitive may not be obvious (if it is part of a blend), we have experimented with several rendering modes (Figure 12) that display the selected internal volume.

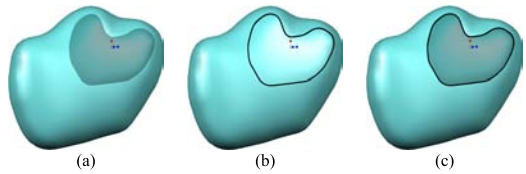

Figure 12: *Internal volumes can be displayed using (a) transparency, (b) silhouette lines, or (c) transparency and silhouttes.*

This selection system only allows for selection of primitives. To select composition nodes we implement a *parent* gesture, which selects the parent of the current node. The *parent* gesture is entered as a straight line towards the top of the screen. Other tree editing operations, such as cut-and-paste, currently require the use of a standard tree widget. An integrated tree visualization tool with gesture-based editing is a feature that we plan on exploring.

A selected primitive or composition node can be removed using the *eraser* gesture described in Section 4.1. Removing a compostion node is equivalent to cutting a branch from the model tree - all children are also removed.

To support 3D manipulation we have implemented standard 3D translation and rotation widgets. These widgets provide both free-translation/rotation in the view-parallel plane as well as constrained manipulation with respect to the unit axes. Compared to the fluid gestural commands used elsewhere in ShapeShop, these 3D widgets are rather crude.

5. Implementation Details

5.1. Hierarchical Implicit Volume Modeling

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

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

where v_{iso} is called the *iso-value*. We call $\mathcal{V}$ an *implicit volume*. The surface $\mathcal{S}$ of this volume is defined by replacing the inequality in (1) with an equality. We call this surface the *implicit surface* [Blo97]. This definition also holds in 2D, where $\mathcal{S}$ is a contour.

Two implicit volumes, defined by scalar functions f_1 and f_2, can be combined functionally using a composition operator $\mathcal{G}(f_1, f_2) \in \mathbb{R}_+$. Since $\mathcal{G}$ is also scalar function, composition operators can be applied recursively. A variety of

operators are available for performing Computational Solid Geometry (CSG), blending, and space deformation [Blo97].

Recursive application of composition operators results in a tree-like data structure with implicit volumes (*primitives*) at the leaves and composition operators at tree nodes. The final scalar field is evaluated at the root composition operator, which recursively evaluates its children, and so on. This type of procedurally-defined implicit volume model is often called a BlobTree [WGG99].

We restrict the set of primitives we use to those with *bounded* † scalar fields. A scalar field f is bounded if $f = 0$ 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.

One type of implicit volume primitive with a bounded scalar field is the *skeletal primitive*, defined by a geometric skeleton E (such as a point or line) and a one-dimensional function $g : \mathbb{R}_+ \rightarrow \mathbb{R}_+$. The scalar function f is then:

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

where d_{E} is a function that computes the minimum Euclidean distance from $\mathbf{p}$ to E. The shape of a skeletal primitive is primarily determined by E. We use the following function for g [Wyv05]:

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

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. We choose 0.5 as the iso-value.

The basic tree data structure can be augmented by attaching an affine transformation to each node, producing a scene graph suitable for animation. To avoid useless field value queries, a bounding volume containing the non-zero portion of the scalar field can be attached to each node.

To improve interactivity, we use Hierarchical Spatial Caching [SWG05]. *Cache nodes* containing lazily-evaluated discrete volume datasets are inserted into the BlobTree to approximate portions of the model tree. This technique provides interactive performance for complex models.

5.2. A Sketch-Based BlobTree Primitive

Our algorithm for inflating a 2D curve $\mathcal{C}$ consists of two steps. First, we create a bounded 2D scalar field f_M, such that the iso-contour $f_M = v_{iso}$ closely approximates $\mathcal{C}$. Then, we sweep this 2D field along an infinite 3D axis and bound it using g_{wyvill} (Equation 3). The following description is necessarily brief, see our technical report [SW05] for a detailed discussion of our blobby inflation technique, linear sweeps, and surfaces of revolution.

Computing the 2D scalar field f_M also consists of two steps, first creating an unbounded field and then bounding it with g_{wyvill}. We create an unbounded scalar field $f_{\hat{M}}$ such that the iso-contour $f_{\hat{M}} = g^{-1}_{wyvill}(0.5)$ approximates $\mathcal{C}$ by fitting a variational curve to a set of sample points lying on $\mathcal{C}$. To adequately constrain the result, we must also consider off-curve points when fitting the variational solution. We automatically generate inside and outside off-curve points along vectors normal to $\mathcal{C}$, similar to the normal constraints used in 3D variational surface fitting [CBC*01] [TO02]. Additional constraint points are created at a constant radius r_c from $\mathbf{c}$, the center of the bounding box of $\mathcal{C}$. The purpose of these additional constraint points is to attempt to force $f_{\hat{M}}$ to more closely approximate the distance field of $\mathcal{C}$. Distance fields are not used because they contains C^1 discontinuities which create the appearance of creases in the inflated surface.

Once we have computed the 2D variational scalar field $f_{\hat{M}}$, we define f_M at 2D points $\mathbf{u}$:

$$f_M(\mathbf{u}) = g_{wyvill}\left(f_{\hat{M}}(\mathbf{u})\right) \tag{4}$$

which is bounded inside a circle of radius 2 if $\mathcal{C}$ is scaled to fit inside a unit box before computing $f_{\hat{M}}$ and a value of 2 is used for r_c. The resulting scalar field is C^2 smooth and the iso-contour $f_M = v_{iso}$ closely approximates $\mathcal{C}$ (Figure 13).

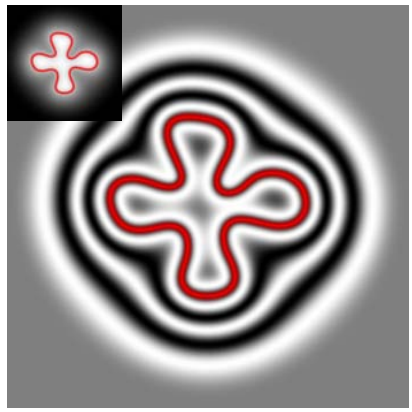

Figure 13: *2D scalar field created using Equation 4. Iso-contours hilighted using* sin *function before mapping to grayscale. Iso-surface is marked in red.*

Creating a 3D bounded scalar field based on f_M is relatively straightforward. Given an origin $\mathbf{o}$, normal $\mathbf{n}$, and two mutually perpendicular vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ in the plane defined by $\mathbf{n}$, we can define an infinite linear sweep of the field f_M. To evaluate f_M at some 3D point $\mathbf{p}$, we require a function

† We use the term *bounded*, rather than *compact support*, in an attempt to draw an analogy to the concept of bounding boxes that is ubiquitous in computer graphics.

F that maps **p** to a 2D point **u**:

$$\mathbf{F}(\mathbf{p}) = \mathbf{Rot}\begin{bmatrix}\mathbf{k}_1 & \mathbf{k}_2 & \mathbf{n}\end{bmatrix} \cdot \mathbf{Tr}\left[-(\mathbf{o}+s\mathbf{n})\right] \cdot \mathbf{p} \qquad (5)$$

where $s = (\mathbf{p}-\mathbf{o}) \cdot \mathbf{n}$, $\mathbf{Rot}[\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3]$ is a homogeneous transformation matrix with upper left 3x3 submatrix $[\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3]^\top$ and $\mathbf{Tr}[\mathbf{e}_t]$ is a homogeneous translation matrix with translation component $\mathbf{e}_t$. The z coordinate of $\mathbf{F}(\mathbf{p})$ is dropped, resulting in a 2D point **u**.

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

$$f_{linear}(\mathbf{p}) = f_M(\,\mathbf{F}(\mathbf{p})\,)$$

This function f_{linear} defines an scalar field of infinite extent along **n**. To bound the field, we multiply f_{linear} by g_{wyvill}:

$$f_{inflate}(\mathbf{p}) = g_{wyvill}\left(\frac{|s|}{d_{endcap}}\right) \cdot f_{linear}(\mathbf{p}) \qquad (6)$$

where d_{endcap} determines the width of the falloff region. The width of the implicit surface varies (Figure 1) because f_M has increasing values inside the 2D contour, as can be seen in Figure 13. Larger values of f_{linear} extend further along **n**, producing a variable-width surface that mimics the inflation techniques of Teddy [IMT99] and other systems.

Equation 6 is computationally expensive because evaluating $f_{\hat{M}}$ is is $O(N)$ in the number of constraint points. The non-zero region of f_M can be discretely approximated using a *field image*. An example is shown in the inset of Figure 13. The field image is sampled in constant time using a C^1 continuous biquadratic reconstruction filter [BMDS02]

The chordal axis techniques used in previous sketch modeling systems [IMT99] can be adapted to create implicit surfaces based on the skeletal primitive approach (Equation 2). However, the resulting scalar field contains C^1 discontinuities which produce unintuitive blending behavior. In addition, this skeletal approach is much slower than the field image-based technique we have described.

Figure 14: *A Mock-up of a mechanical part sketched with ShapeShop. This model was sketched in under 10 minutes.*

5.3. Sketch Modeling Implementation

Sketch-modeling operations are implemented by replacing the root node of the current BlobTree with a new composition operator. The existing root node is added as the first child, and the new primitive as the second. To implement cutting (Section 3.3), we create a new CSG difference node which subtracts a linear sweep from the current volume. We use a C^1 CSG difference function [BWdG04] which prevents unsightly gradient discontinuities. Blending (Section 3.4) is implemented with the parameterized Hyperblend [Ric73] [WGG99], which affords some control over the blend surface.

To visualize the implicit surface we use an optimized version of Bloomenthal's polygonizer [Blo94]. We provide control over the polygonizer resolution, allowing the user to determine the trade-off between accuracy and interactivity. The images in this paper were all rendered using high-resolution polygonizations which take approximately 5-10 seconds to compute. The Extended Marching Cubes [KBSS01] polygonization algorithm is used to recover sharp features (Figure 14).

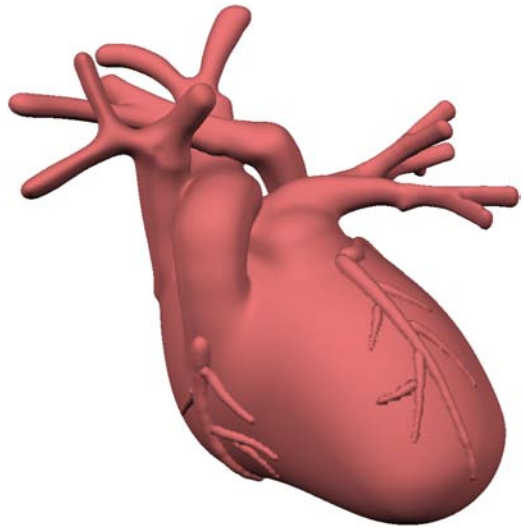

Figure 15: *Heart model sketched in approximately 30 minutes. Complex branching structures can be created quickly by blending simple parts. Our surface-drawing technique is useful for creating anatomical details such as veins.*

6. Results

The benefit of an underlying analytic representation is particularly apparent in CAD-style models (Figure 14). Sharp edges created with CSG are mathematically precise. Since the BlobTree is also a scene graph, the separate parts in this model can be animated. This could, for example, allow an engineer to easily create an interactive assembly manual.

The flexible blending capabilities of implicit modeling are useful when constructing biological models (Figure 15). Since smooth surface transitions are automatic, complex topologies can be assembled quickly from simple parts. The volumetric BlobTree representation supports sketching of internal volumes (Figure 16), which can be applied to biological models to aid in visualization and communication.

Many of the free-form sketch-based systems described in Section 2 have explored character modeling. However, the hierarchical nature of the BlobTree allows our character models to be fully articulated, even when the internal components are blended to form smooth surfaces (Figure 17). These articulated models can be animated directly.

7. Discussion

Hierarchical implicit volume modeling is a useful tool for a wide range of modeling tasks. Employing BlobTrees as an underlying shape representation has allowed us to design an interactive system that supports sketch-based creation of complex 3D models. The models displayed in Figures 14-17 exhibit significantly higher surface complexity than the models demonstrated in existing systems. Further, these models do not indicate the complexity limit of ShapeShop, but rather the point at which these models were considered "finished" by the creator (the primary author).

Only informal observations of graduate students using ShapeShop have been performed. The area that caused the most confusion was selection of non-primitive nodes. Users must understand the hierarchical BlobTree concept, however currently we do not visualize the tree and inferring it's structure by inspection is difficult. This must be improved in future systems. The 3D transformation widgets were also problematic, we plan on exploring techniques based on those described in the SKETCH system [ZHH96]. Many aspects of the sketch-based interface should be analyzed with formal usability studies.

Figure 16: *The body of this car model was initially sketched, and then the internal structure was carved out. Right image shows cut-away view.*

The 2D curve-sketching technique described in Section 4.1 is limited to smooth contours. Integration of methods for adding sharp creases would be beneficial, particularly in the case of CAD models. Techniques such as those described in suggestive sketch-based systems [ZHH96] [JSC03] would also be useful to assist with sketching CAD-style models.

A key property of implicit volume modeling is that composition operators do not depend on the shape of underlying volumes. We have demonstrated that with BlobTrees, CAD-style solid modeling and free-form modeling can be integrated into a single interface. While generality has practical advantages, a more fundamental benefit may come from giving designers a modeling tool which does not prescribe a particular modeling "style".

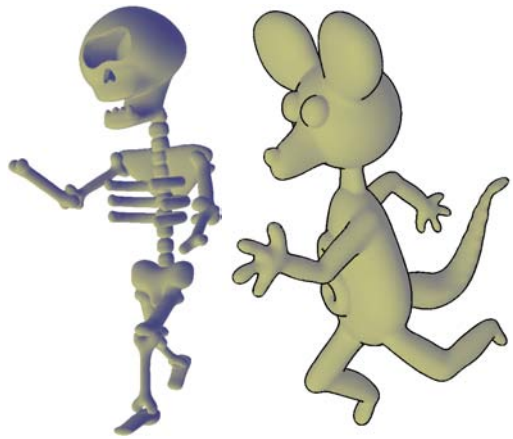

Figure 17: *Character models created with ShapeShop. The skeleton model (left) is composed of 36 primitives in a hierarchical arrangement that is suitable for direct animation.*

Acknowledgements

We would like to thank the anonymous reviewers for their comments and suggestions. This work was supported by the National Sciences and Engineering Research Council of Canada and iCore.

References

[AG04] APITZ G., GUIMBRETIÉRE F.: Crossy: a crossing-based drawing application. In *Proceedings of ACM UIST 2004* (2004), pp. 3–12. 4

[AGB04] ALEXE A., GAILDRAT V., BARTHE L.: Interactive modelling from sketches using spherical implicit functions. In *Proceedings of AFRIGRAPH 2004* (2004), pp. 25–34. 1, 2

[AJ03] ARAÚJO B., JORGE J.: Blobmaker: Free-form modelling with variational implicit surfaces. In *Proceedings of 12th Encontro Português de Computação Gráfica* (2003). 1, 2, 5

[Blo94] BLOOMENTHAL J.: *Graphics Gems IV*. Academic Press Professional Inc., 1994, ch. An implicit surface polygonizer, pp. 324–349. 8

[Blo97] BLOOMENTHAL J. (Ed.): *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997. 6, 7

[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. 8

[BWdG04] BARTHE L., WYVILL B., DE GROOT E.: Controllable binary csg operators for soft objects. *International Journal of Shape Modeling 10*, 2 (2004), 135–154. 8

[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 '01* (2001), pp. 67–76. 5, 7

[CSSJ05] CHERLIN J. J., SAMAVATI F., SOUSA M. C., JORGE J. A.: Sketch-based modeling with few strokes. In *Proceedings of the Spring Conference on Computer Graphics* (2005). 2, 3

[FFJ04] FONSECA M. J., FERREIRA A., JORGE J. A.: Towards 3d modeling using sketches and retrieval. In *Proceedings of the First Eurographics Workshop on Sketch-Based Interfaces and Modeling* (2004). 5

[IH01] IGARASHI T., HUGHES J. F.: A suggestive interface for 3d drawing. In *Proceedings of ACM UIST 2001* (2001), pp. 173–181. 2, 5

[IH03] IGARASHI T., HUGHES J. F.: Smooth meshes for sketch-based freeform modeling. In *Proceedings of the 2003 symposium on Interactive 3D graphics* (2003), pp. 139–142. 1, 2

[IMKT97] IGARASHI T., MATSUOKA S., KAWACHIYA S., TANAKA H.: Interactive beautification: a technique for rapid geometric design. In *Proceedings of ACM UIST '97* (1997), pp. 105–114. 4

[IMT99] IGARASHI T., MATSUOKA S., TANAKA H.: Teddy: A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH 99* (1999), pp. 409–416. 1, 2, 8

[JSC03] JORGE J. A., SILVA N. F., CARDOSO T. D.: Gides++. In *Proceedings of 12th Encontro Português de Computação Gráfica* (2003). 2, 9

[KBSS01] KOBBELT L. P., BOTSCH M., SCHWANECKE U., SEIDEL H.-P.: Feature-sensitive surface extraction from volume data. In *Proceedings of ACM SIGGRAPH 2001* (2001), pp. 57–66. 8

[KHR02] KARPENKO O., HUGHES J., RASKAR R.: Free-form sketching with variational implicit surfaces. *Computer Graphics Forum 21*, 3 (2002), 585 – 594. 1, 2, 3

[MCCH99] MARKOSIAN L., COHEN J. M., CRULLI T., HUGHES J. F.: Skin: A constructive approach to modeling free-form shapes. *Proceedings of SIGGRAPH 99* (1999), 393–400. 2

[ONNI03] OWADA S., NIELSEN F., NAKAZAWA K., IGARASHI T.: A sketching interface for modeling the internal structures of 3d shapes. In *Proceedings of the 4th International Symposium on Smart Graphics* (2003), pp. 49–57. 1, 2, 6

[Ric73] RICCI A.: A constructive geometry for computer graphics. *Computer Graphics Journal 16*, 2 (1973), 157–160. 8

[SW05] SCHMIDT R., WYVILL B.: *Implicit Sweep Surfaces*. Tech. Rep. 2005-778-09, University of Calgary, 2005. 7

[SWG05] SCHMIDT R., WYVILL B., GALIN E.: Interactive implicit modeling with hierarchical spatial caching. In *Proceedings of Shape Modeling International 2005* (2005), pp. 104–113. 1, 7

[TO02] TURK G., O'BRIEN J. F.: Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics 21*, 4 (2002), 855–873. 5, 7

[TZF04] TAI C.-L., ZHANG H., FONG J. C.-K.: Prototype modeling from sketched silhouettes based on convolution surfaces. *Computer Graphics Forum 23*, 1 (2004), 71–83. 1, 2

[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. 1, 2, 7, 8

[Wyv05] WYVILL G.: Wyvill function. Personal Communication, 2005. 7

[ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J. F.: Sketch: an interface for sketching 3d scenes. In *Proceedings of ACM SIGGRAPH 96* (1996), pp. 163–170. 2, 4, 9