

# Interactive Re-Polygonization of Blobby Implicit Curves

Barton T. Stander

John C. Hart

School of EECS  
Washington State University  
Pullman, WA 99164-2752  
{bstander,hart}@eecs.wsu.edu

## Abstract

An interactive modeling system for implicit curves is presented. Unlike other interactive algorithms, ours displays connected polygonal approximations instead of several unconnected points. We also guarantee that the topology of our polygonal approximation matches the topology of the true curve. The system currently handles only 2-D blobby curves, but will be extended to include general 3-D implicit surfaces.

**Keywords:** implicit curves and surfaces, vector field topology, polygonization, particle dynamics, interactive modeling.

## 1 Introduction

An implicit curve is the set of points  $x \in \mathbf{R}^2$  that satisfy  $F(x) = 0$ . Implicit curves and surfaces have some modeling advantages over their parametric counterparts including their ability to easily be combined and blended in various fashions. [Blinn, 1982; Rockwood & Owen, 1987]. But due to the difficulty of sampling them, until recently it has been difficult to model implicit curves and surfaces in an interactive environment. Furthermore, there is seldom an obvious correlation between the parameters of the defining function and the resulting shape, making them difficult tools for interactive sculpting [Witkin & Heckbert, 1994].

Witkin and Heckbert recently proposed a solution to these problems [Witkin & Heckbert, 1994]. Their work will be discussed in detail in the next section. The drawback of their solution is that the final rendering consists of oriented particles, not polygons, because their solution ignores topology.

Our purpose is to remedy this shortcoming, so that implicit curves and surfaces can still be modeled in

real time, but rendered with polygons of guaranteed topology.

We begin with a review of previous work, giving special attention to Witkin and Heckbert's contributions. We then list an overview of our algorithm, as an extension to theirs. The remaining sections give the specific details of our algorithm. We end with a summary and future work.

## 2 Previous Work

Our interactive repolygonization method for implicit curve and surface manipulation combines tools from three related areas: surface interrogation, particle systems, and vector-field topology.

### 2.1 Polygonization

Whereas parametric surfaces lend themselves to forward visible-surface algorithms, such as the z-buffering, implicit surfaces are better suited for backward algorithms, such as ray tracing. As many graphics workstations implement z-buffering in hardware but few similarly support ray tracing, real-time manipulation demands forward rendering.

The implicit formulation enjoys many benefits, though its main drawback is that its surface is difficult to interrogate. Hence fast "forward" rendering typically requires a polygonization step.

Polygonization algorithms typically interrogate implicit surfaces through spatial sampling. There are several spatial sampling techniques for interrogating implicit surfaces, and all divide space into cells and search for only those cells that intersect the implicit surface. Ning and Bloomenthal review several variations on this theme [Ning & Hesselink, 1993], focusing on topological consistency and correctness, although

none of their surveyed algorithms guarantees topological correctness.

Surface tracking techniques [Norton, 1982; Wyvill *et al.*, 1986; Lorensen & Cline, 1987] partition space into small cells and begin with a cell known to straddle the implicit surface (some vertices in, some out). The technique then recursively finds straddling cells among its neighbors until all neighbors have been checked, yielding a collection of cells representing the geometry of the implicit surface.

Spatial subdivision techniques [Bloomenthal, 1988; Kalra & Barr, 1989; Snyder, 1992] begin with one large cell known to bound the implicit surface, then repeatedly subdivide cells intersecting the implicit surface. Lipschitz bounds or interval analysis can guarantee that the implicit surface is bounded by the resulting cells, hence yielding a guarantee on surface topology.

Spatial interrogation techniques, particularly in guaranteed form, remain too costly for real-time use.

## 2.2 Particle Systems

An alternative to spatial interrogation constrains a particle system to the implicit surface. Such physically-based approaches tend to be much faster at interrogating an implicit surface. The technique scatters particles randomly throughout space and then forces them to migrate to the implicit surface using its defining function’s sign and gradient direction. Once the particles reach the surface, they repel each other in order to achieve a more uniform sampling distribution [Turk, 1991; Bloomenthal & Wyvill, 1990; de Figueiredo *et al.*, 1992].

In an interactive environment, such sample points may “fall off” when the user quickly alters the surface, and otherwise find it difficult to maintain a balanced distribution. Witkin and Heckbert [Witkin & Heckbert, 1994] overcame these problems by solving for particle velocities in terms of surface parameter velocities  $q$ . This resulted from the key observation that  $\partial F/\partial q$  is continuous. This kept the particles on the surface, nearly eliminating the dependency on “feedback” terms. They balanced the distribution of sample points by introducing an intricate particle “birth and death” scheme.

Witkin and Heckbert also provide a direct manipulation interface by allowing the user to choose one or more “control particles” which also reside directly on the surface. The user drags or nails these control particles, which in turn alter the actual surface

parameters. They keep the surface on the control particles by solving the reverse problem from above. That is, they solve for surface parameter velocities in terms of control point velocities. In both cases, a high-order differential equation solver and a feedback term keep the particles and the actual surface exactly in sync.

The rendering technique used by Witkin and Heckbert’s algorithm consists of drawing each particle as a disk oriented tangent to the surface. This gives a somewhat useful representation of the underlying surface, but it lacks the important depth cues of hidden surface removal, and topological information is not always discernible. Furthermore, a topologically-interesting component such as a lake or an island may be missed altogether if the initial set of randomly placed particles never found it, or if it came into existence later at a place disjoint from the known components.

Particle system approaches swiftly interrogate the surface with sample points, but “connecting the dots” can easily kill an otherwise interactive environment. Figueiredo, for example, used Delaunay triangulation to build a polygonal mesh [de Figueiredo *et al.*, 1992]. The result is not interactive.

## 2.3 Vector-Field Topology

Another approach to the topology problem is to use critical points and skeletal vector fields to separate space into topologically uniform regions [Delmarcelle & Hesselink, 1994; Helman & Hesselink, 1991; Globus *et al.*, 1991]. We base our solution on this foundation.

## 3 Algorithm Overview

The key to our solution is in the realization that a change in the topology of a curve or surface is always accompanied by a change in the set of critical points (i.e., points where the gradient vector vanishes). Thus, by merely watching the critical points, the great burden of detecting topological change can largely be ignored. The three kinds of critical points are maximums, minimums, and saddle points. Initially, we find all critical points in a guaranteed fashion with an exhaustive space search using interval arithmetic (Section 3.3). Critical point  $(a, b)$  of function  $F$  is then classified using the following tests [Taylor, 1955]:

$$\text{Let } A = Fxx(a, b)$$

Let  $B = F_{xy}(a, b)$

Let  $C = F_{yy}(a, b)$

If  $B * B - A * C < 0$  and  $A > 0$  then  $F$  has a relative minimum at  $(a, b)$ .

If  $B * B - A * C < 0$  and  $A < 0$  then  $F$  has a relative maximum at  $(a, b)$ .

If  $B * B - A * C > 0$  then  $F$  has a saddle point at  $(a, b)$ .

During user interaction, the critical points move and change sign. Sometimes two critical points combine and cease to exist, other times two critical points are created out of nothing [Delmarcelle & Hesselink, 1994].

Our algorithm consists of an initialization stage followed by an interactive loop of user input, model update, and model display. The details of the following steps are provided in subsequent sections.

We begin with some initial set of blobbies with parameters set to various values. We initialize the system so that there are several well distributed particles on each topologically disjoint piece of the true curve, and so that these particles are correctly connected to each other in one or more closed polygons, which we refer to as “components.” We also identify and classify all critical points, which we use to detect changes in topology. The problem then becomes one of maintaining this correct picture during user interaction.

For each time step, we:

1. Allow the user to modify the curve with the mouse, and we alter the implicit curve’s parameters accordingly [Witkin & Heckbert, 1994].
2. Solve for the updated positions of the particles so that they stay exactly on the updated curve [Witkin & Heckbert, 1994].
3. Solve for the updated positions of the critical points (Section 3.1).
4. Delete pairs of critical point that collide.
5. Perform a quick (interactive) search for newly created critical points (Section 3.2).
6. Perform an exhaustive (non-interactive) search for newly created critical points whenever user interaction ceases (Section 3.3).
7. Detect any change in topology using the list of critical points (Section 3.4).
8. Correct any topology changes (Section 3.5).

9. Render each of the components as closed polygons.

### 3.1 Tracking Critical Points

When the user alters the blobby curve’s parameters, the positions of some or all critical points also change. At present, the critical points are tracked by using gradient descent to find the closest location where the first derivative vanishes. This feedback method by itself is somewhat dangerous because if a step size is too large then a critical point tracker following gradient descent floats in the direction opposite from the actual critical point. We will later solve for critical point velocities in terms of parameter velocities, much the same way Witkin and Heckbert did for particle velocities.

### 3.2 Finding Critical Points with Feelers

Critical points are always created in pairs - either a saddle point and a maximum or a saddle point and a minimum. New maximum points are relatively easy to find by placing a feeler at the peak of every blob, and letting them float upward to all the maximum points. When a new maximum point is detected, its matching saddle point is also found with relative ease using a localized search.

Detecting new minimum point/saddle point pairs is harder to do in real time. We continually attempt to find such critical points by placing feelers at random locations and letting them search for critical points using gradient descent toward locations where the first derivative vanishes. Feelers that roam into known critical points or that run off the outermost blobbies are reset to other random locations. When a minimum/saddle pair is first created, its area of detectability by a feeler can be arbitrarily small, but the new critical point is usually detected before it changes sign, which is ultimately all we care about.

### 3.3 Finding All Critical Points with an Exhaustive Search

Using interval arithmetic in a recursive quad-tree fashion, we find all critical points within a few seconds (Figure 1). Unfortunately, that is too long for an interactive environment, so this guaranteed process is only performed at initialization and when user activity subsides. A yellow light, green light approach

indicates to the user when all critical points are definitely known, and thus that the topology is guaranteed to be correct.

### 3.4 Detecting a Topology Change

The following observations are given at present without proof:

1. Two topologically separate components merge into one if and only if the function value at some saddle point changes sign from negative to positive.
2. Likewise, one component separates into two if and only if the function value at some saddle point changes sign from positive to negative.
3. A new, disjoint component is formed if and only if the functional value at a minimum point changes sign from positive to negative.
4. Likewise, a component disappears exactly when a minimum point's sign changes from negative to positive.

(Note that with blobby circles, maximum points never change sign.)

### 3.5 Correcting a Topology Change

Whenever two components merge, or when one component separates, the alteration always occurs about the saddle point. In both cases, exactly two segments become invalid (i.e., they attempt to approximate pieces of true curve which are no longer there). The two offending segments are detected by examining all segments in the vicinity of the saddle point. The four points involved are reconnected in the other direction (Figure 2).

When a new component is formed, three particles are placed on it by initializing them near the minimum point and letting them float to the surface as discussed earlier. More in-between points are added until there are enough particles on the new component to fulfill geometric accuracy requirements (Figure 3).

When a component ceases to exist, all particles that were on it are deleted.

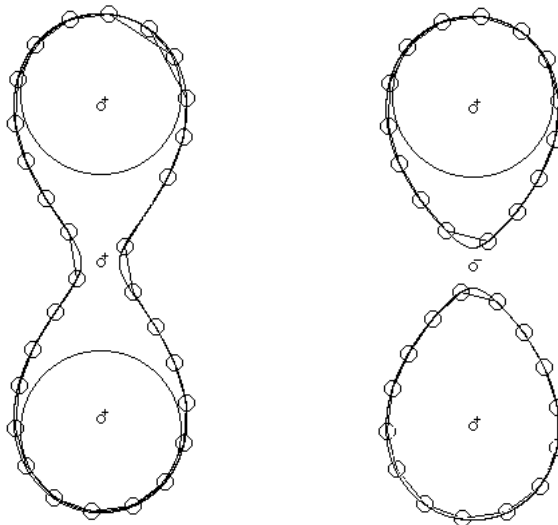


Figure 2: When the functional value of a saddle point changes sign from positive to negative, one component separates into two, so we reconnect the 4 nearby points to contain valid segments (left). Note that these 4 points straddle the saddle point. When a saddle point changes sign, reconnect the 4 nearby points containing the invalid segments (right).

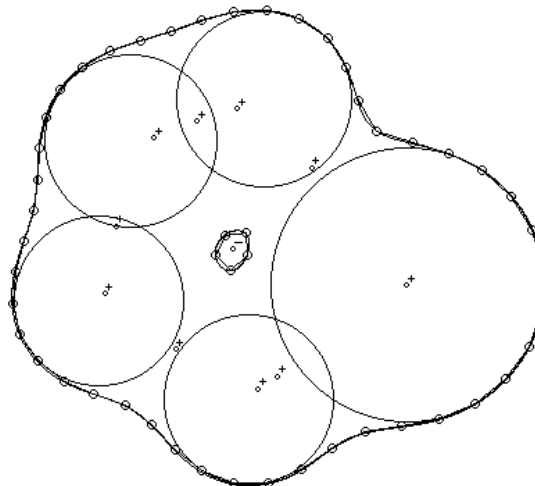


Figure 3: When the functional value of a minimum point changes sign from positive to negative, a "lake" is created.

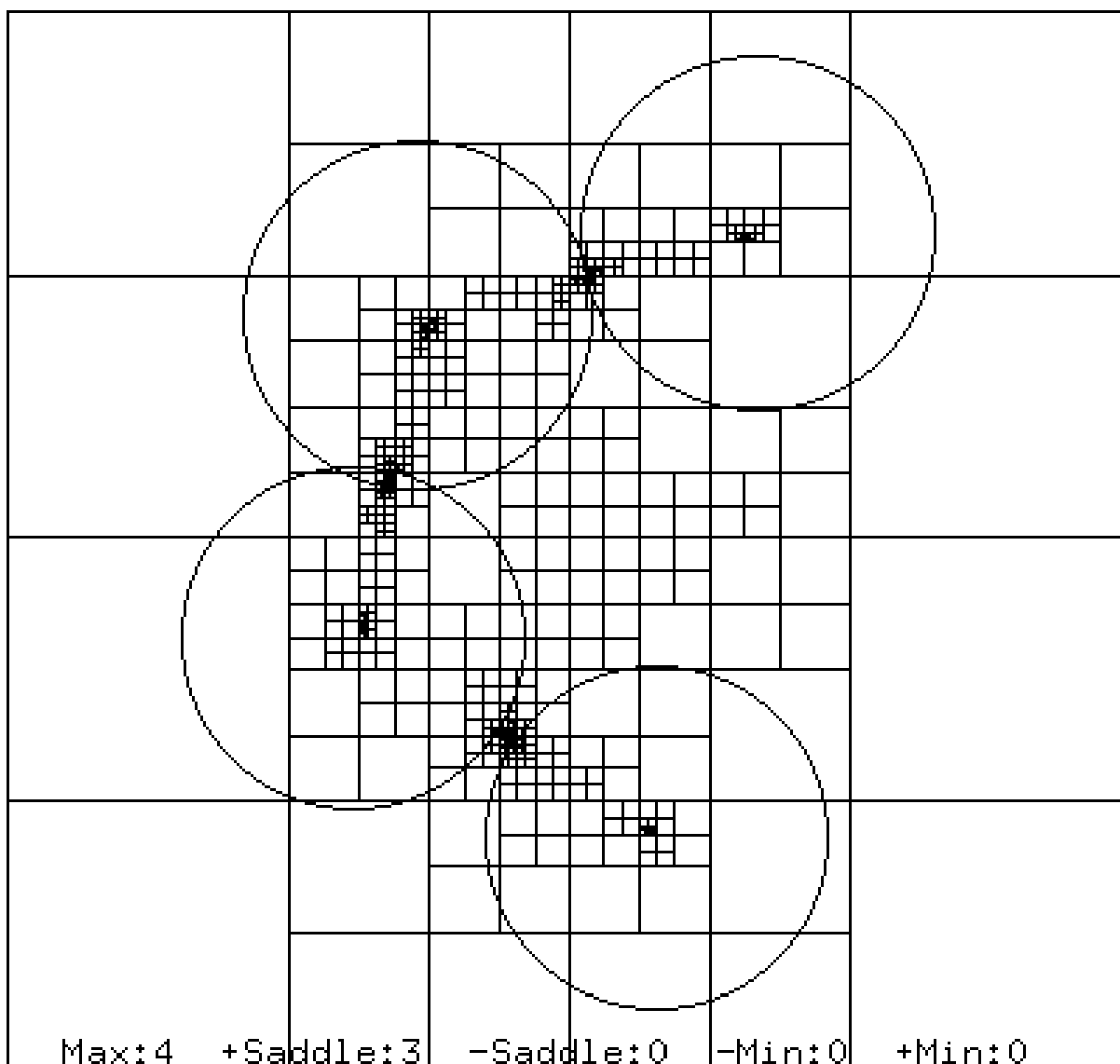


Figure 1: We find all critical points using interval arithmetic

## 4 Conclusion

In summary, we extended the interactive modeling system of Witkin and Heckbert so that the particles also have connectivity information. We then render polygons instead of points or disks, and we guarantee that the topology of our polygonal approximation is consistent with the topology of the actual implicit curve.

### 4.1 Future Work

Future work consists of extending the environment to include 3-D blobby ellipsoids, and eventually to include general implicit surfaces as well. We also need to complete the work described in this paper.

### 4.2 Acknowledgments

The authors are supported in part by the NSF under Research Initiation Award #CCR-9309210. This research was performed at the Imaging Research Laboratory, which is supported in part by the NSF under grant #CDA-9121675.

## References

- [Blinn, 1982] Blinn, J. F. A generalization of algebraic surface drawing. *ACM Transactions on Graphics* 1(3), July 1982, pp. 235–256.
- [Bloomenthal & Wyvill, 1990] Bloomenthal, J. and Wyvill, B. Interactive techniques for implicit modeling. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, March 1990, pp. 109–116.
- [Bloomenthal, 1988] Bloomenthal, J. Polygonization of implicit surfaces. *Computer Aided Geometric Design* 5(4), Nov. 1988, pp. 341–355.
- [de Figueiredo *et al.*, 1992] de Figueiredo, L. H., de Miranda Gomes, J., Terzopoulos, D., and Velho, L. Physically-based methods for polygonization of implicit surfaces. *Graphics Interface '92*, May 1992, pp. 250–257.
- [Delmarcelle & Hesselink, 1994] Delmarcelle, T. and Hesselink, L. The topology of symmetric, second-order tensor fields. *Proceedings IEEE Visualization '94*, October 1994, pp. 140–147.
- [Globus *et al.*, 1991] Globus, A., Levit, C., and Lasinski, T. A tool for visualizing the topology of three-dimensional vector fields. *Proceedings IEEE Visualization '91*, October 1991, pp. 33–40.
- [Helman & Hesselink, 1991] Helman, J. L. and Hesselink, L. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics and Applications*, May 1991, pp. 36–46.
- [Kalra & Barr, 1989] Kalra, D. and Barr, A. H. Guaranteed ray intersections with implicit surfaces. *Computer Graphics* 23(3), July 1989, pp. 297–306.
- [Lorensen & Cline, 1987] Lorensen, W. E. and Cline, H. E. Marching cubes: A high resolution 3-d surface construction algorithm. *Computer Graphics* 21(4), July 1987, pp. 163–170.
- [Ning & Hesselink, 1993] Ning, P. and Hesselink, L. Fast volume rendering of compressed data. In Nielson, G. M. and Bergeron, D., eds., *Proc. of Visualization '93*. IEEE Computer Society Press, 1993, pp. 11–18.
- [Norton, 1982] Norton, A. Generation and rendering of geometric fractals in 3-D. *Computer Graphics* 16(3), 1982, pp. 61–67.
- [Rockwood & Owen, 1987] Rockwood, A. P. and Owen, J. C. Blending surfaces in solid modeling. In Farin, G., ed., *Geometric Modelling*, pp. 367–383. SIAM, 1987.
- [Snyder, 1992] Snyder, J. M. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH '92 Proceedings)*, July 1992, pp. 121–130.
- [Taylor, 1955] Taylor, A. E. *Advanced Calculus*. Ginn and Company, 1955.
- [Turk, 1991] Turk, G. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics (SIGGRAPH '91 Proceedings)*, July 1991, pp. 289–298.
- [Witkin & Heckbert, 1994] Witkin, A. and Heckbert, P. Using particles to sample and control implicit surfaces. *Computer Graphics (SIGGRAPH '94 Proceedings)*, July 1994, pp. 269–277.
- [Wyvill *et al.*, 1986] Wyvill, G., McPheeters, C., and Wyvill, B. Data structure for soft objects. *Visual Computer* 2(4), 1986, pp. 227–234.