Variational Implicit Surfaces

Greg Turk James F. O'Brien

Georgia Institute of Technology

Abstract

We introduce a new method of creating smooth implicit surfaces of arbitrary manifold topology. These surfaces are described by specifying locations in 3D through which the surface should pass, and also identifying locations that are interior or exterior to the surface. A 3D implicit function is created from these constraints using a variational scattered data interpolation approach. We call the iso-surface of this function a *variational implicit surface*. Like other implicit surface descriptions, these surfaces can be used for CSG and interference detection, may be interactively manipulated, are readily approximated by polygonal tilings, and are easy to ray trace. A key strength is that variational implicit surfaces allow the direct specification of both the location of points on the surface and surface normals. These are two important manipulation techniques that are difficult to achieve using other implicit surface representations such as sums of spherical or ellipsoidal Gaussian functions ("blobbies"). We show that these properties make variational implicit surfaces particularly attractive for interactive sculpting using the particle sampling technique introduced by Witkin and Heckbert in [30]. Our formulation also yields a simple method for converting a polygonal model to a smooth implicit model.

1 Introduction

The computer graphics, computer-aided design and computer vision literatures are filled with an amazingly diverse array of approaches to surface description. The reason for this variety is that there is no single representation of surfaces that satisfies the needs of every problem in every application area. This paper introduces variational implicit surfaces, a new surface creation method that we believe will be useful in several areas in 3D modeling. Variational implicit surfaces are smooth, exactly pass through a set of given constraint points, and can describe closed surfaces of arbitrary topology.

Figure 1 (left) shows a variational implicit curve, the 2D analog to a surface, in order to illustrate our basic approach. The small open circles in this figure are constraint positions at which a 2D implicit function must take on the value of zero. The single plus sign indicates the position at which the implicit function is to take on the value one, and in fact any positive value will do. The locations and values (zeros and ones) at the small circles and at the plus sign are constraints that are passed along to a scattered data interpolation routine. The interpolation routine yields a smooth 2D function that meets the given constraints. The desired curve is defined to be the locus of points at which the function takes on the value of zero. The curve exactly passes through each of the zero-value constraints, and its defining function is positive inside this curve and negative outside. For this 2D example, we use a variational technique that minimizes the aggregate curvature of the function that it creates, and this technique is often referred to as thin-plate interpolation.

We can create surfaces in 3D in exactly the same way as the 2D curves in Figure 1. Zero-valued constraints are defined by the modeler at 3D locations, and positive values are specified at one or more places that are to be interior to the surface. A variational interpolation technique is then invoked that creates a scalar-valued function over a 3D domain. The desired surface is simply the set of all points at which this scalar function takes on the value zero. Figure 2 (left) shows a surface that was created in this fashion by placing four zero-valued constraints at the vertices of a regular tetrahedron and placing a single positive constraint in the center of the tetrahedron. The result is a nearly spherical surface. More complex shapes such as that of Figure 2 (right) can be defined simply by specifying a larger number of constraints.

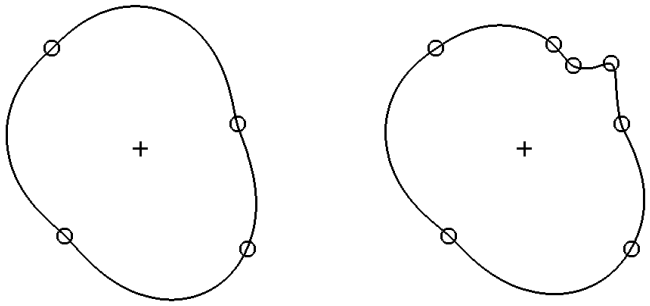

Figure 1: Curves defined using variational implicit functions. The curve on the left is defined by four zero-valued and one positive constraint. This curve is refined by adding three new zero-valued constraints (at right).

The remainder of this paper is organized as follows. In Section 2 we examine related work and review implicit surface and thin-plate interpolation techniques. We describe in Section 3 the mathematical framework for solving variational problems using radial basis functions. Section 4 presents three strategies that may be used together with variational methods to create implicit surfaces. These strategies differ in where they place the non-zero constraints. In Section 5 we show that variational implicit surfaces are well suited for interactive sculpting. In Section 6 we compare variational implicit surfaces with traditional thin-plate surface modeling and with implicit functions that are created using ellipsoidal Gaussian functions. Section 7 describes two rendering techniques, one that relies on polygonal tiling and another based on ray tracing. Finally, Section 8 indicates potential applications and directions for future research.

2 Background and Related Work

Variational implicit surfaces draw upon two areas of modeling: implicit surfaces and thin-plate interpolation. In this section we briefly review work in these two sub-areas.

2.1 Implicit Surfaces

An implicit surface is defined by an implicit function, a continuous scalar-valued function over the domain $\mathbf{R}^3$. The implicit surface of

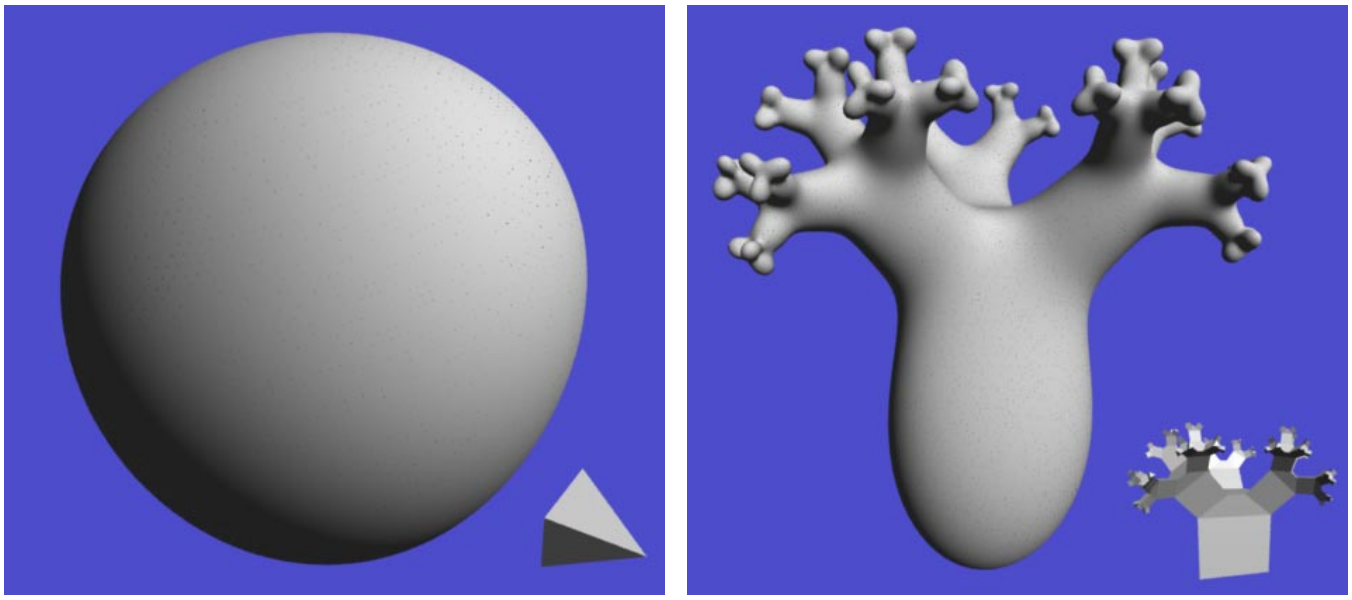

Figure 2: Surfaces defined by variational implicit functions. The left surface is defined by zero-valued constraints at the corners of a tetrahedron and one positive constraint in its center. The branching surface at the right was created using constraints from the vertices of the inset polygonal object.

such a function is the locus of points at which the function takes on the value zero. For example, a unit sphere may be defined using the implicit function $f(\mathbf{x}) = 1 - |\mathbf{x}|$. Points on the sphere are those locations at which $f(\mathbf{x}) = 0$. This implicit function takes on positive values interior to the sphere and is negative outside the surface, as will be the convention in this paper.

An important class of implicit surfaces are the *blobby* or *metaball surfaces* [3, 19]. The implicit functions of these surfaces are the sum of radially symmetric functions that have a Gaussian profile. Here is the general form of such an implicit function:

$$f(\mathbf{x}) = -t + \sum_{i=1}^{n} h_i(\mathbf{x}) \tag{1}$$

In the above equation, a single function h_i describes the profile of a "blobby sphere" that has a particular center and radius. The bold letter $\mathbf{x}$ represents a point in the domain of our implicit function, and in this paper we will use bold letters to represent such points, both in 2D and 3D. The value t is the iso-surface threshold. When the centers of two blobby spheres are close enough to one another, the implicit surface appears as though the two spheres have melted together. A typical form for a blobby sphere function h_i is the following:

$$h_i(\mathbf{x}) = e^{|\mathbf{x}-\mathbf{c}_i|^2/\sigma_i^2} \tag{2}$$

In this equation, the constant σ_i specifies the standard deviation of a Gaussian function, and thus is the control over the radius of a blobby sphere. The center of a blobby sphere is given by $\mathbf{c}_i$. Evaluating an exponential function is computationally expensive, so some authors have used piecewise polynomial expressions instead of exponentials to define these blobby sphere functions [19, 31]. A greater variety of shapes can be created with the blobby approach by using ellipsoidal rather than spherical atomic functions.

Another important class of implicit surfaces are the algebraic surfaces. These are surfaces that are described by polynomial expressions in x, y and z. If a surface is simple enough, it may be described by a single polynomial expression. A good deal of attention has been devoted to this approach, and a good entry point into these kinds of surfaces is the work of Gabriel Taubin [27]. Much of the work in this area has been devoted to fitting an algebraic surfaces to a given collection of points. Usually it is not possible to interpolation all of the data points, so error minimizing techniques are sought. Surfaces may also be described by piecing together many separate algebraic surface patches, and here again there is a large literature on the subject. Good introductions to these surfaces may be found in the chapter by Chanddrajit Bajaj and the chapter by Alyn Rockwood in [6]. It is easier to control the approximation or interpolation of a collection of data points using algebraic patches than with a single algebraic surface. The tradeoff, however, is that these patches require a good deal of machinery to create smooth joins across patch boundaries.

2.2 Thin-Plate Interpolation and Variational Techniques

Thin-plate spline surfaces are a class of height fields that are closely related to the variational implicit surfaces of this paper. Thin-plate interpolation is one approach to solving the *scattered data interpolation* problem. The two-dimensional version of this problem can be stated as follows: Given a collection of k constraint points $\{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_k\}$ that are scattered in the xy-plane, together with scalar height values at each of these points $\{h_1, h_2, \ldots, h_k\}$, construct a "smooth" surface that matches each of these heights at the given locations. We can think of this solution surface as a scalar-valued function $f(\mathbf{x})$ so that $f(\mathbf{c}_i) = h_i$, for $1 \leq i \leq k$. If we define the word *smooth* in a particular way then there is a unique solution to such a problem, and this solution is the thin-plate interpolation of the points. Consider the energy function E:

$$E = \int_{\Omega} f_{xx}^2(\mathbf{x}) + 2 f_{xy}^2(\mathbf{x}) + f_{yy}^2(\mathbf{x})\, d\mathbf{x} \tag{3}$$

The notation f_{xx} means the second partial derivative in the x direction, and the other two terms are similar partial derivatives, one of them mixed. This energy function is basically a measure of the aggregate curvature of $f(\mathbf{x})$ over the region of interest Ω (usually a portion of the plane). Any creases or pinches in a surface will result

in a larger value of E. A smooth surface that has no such regions of high curvature will have a lower value of E. Note that because there are only squared terms in the integral, the value for E can never be negative. The thin-plate solution to an interpolation problem is the function $f(\mathbf{x})$ that satisfies all of the constraints and that has the smallest possible value of E. Note that the most common form of thin-plate surfaces are restricted to be height fields, and thus they are in fact *parametric* surfaces.

This interpolation method gets its name because it is much like taking a thin sheet of metal, laying it horizontally and bending the it so that it just touches the tips of a collection of vertical poles that are set at the positions and heights given by the constraints of the interpolation problem. The metal plate resists bending so that it smoothly changes its height in the positions between the poles. This springy resistance is mimicked by the energy function E. Thin-plate interpolation is often used in the computer vision domain, where there are often sparse surface constraints [13, 28]. The above curvature minimization process is sometimes referred to as regularization, and can be thought of as an additional constraint that selects a unique surface out of an infinite number of surfaces that match a set of given height constraints. Solving such constrained problems draws from a branch of mathematics called the variational calculus, thus thin-plate techniques are sometimes referred to as variational methods.

The scattered data interpolation problem can be formulated in any number of dimensions. When the given points $\mathbf{c}_i$ are positions in N-dimensions rather than in 2D, this is called the N-dimensional scattered data interpolation problem. There are appropriate generalizations to the energy function and to thin-plate interpolation for any dimension. In this paper we will make use of variational interpolation in two and three dimensions.

3 Variational Methods and Radial Basis Functions

The scattered data interpolation task as formulated above is a variational problem where the desired solution is a function, $f(\mathbf{x})$, that will minimize equation 3 subject to the interpolation constraints $f(\mathbf{c}_i) = h_i$. There are several numerical methods that can be used to solve this type of problem. Two commonly used methods, finite elements and finite differencing techniques, discretize the region of interest, Ω, into a set cells or elements and define local basis functions over the elements. The function $f(\mathbf{x})$ can then expressed as a linear combination of the basis functions so that a solution can be found, or approximated, by determining suitable weights for each of the basis functions. This approach has been widely used for height-field interpolation and deformable models, and examples of its use can be found in [28, 26, 7, 29]. While finite elements and finite differencing techniques have proven useful for many problems, the discretization that they require can lead to artifacts.

An alternate approach is to express the solution in terms of radial basis functions centered at the constraint locations. Radial basis functions are radially symmetric about a single point, or center, and they have been widely used for function approximation. Remarkably, it is possible to choose these radial functions is such a way that they will automatically solve differential equations, such as the one required to solve equation 3, subject to constraints located at their centers. For the 2D interplation problem, equation 3 can be solved using the biharmonic radial basis function:

$$\phi(\mathbf{x}) = |\mathbf{x}|^2 \log(|\mathbf{x}|) \tag{4}$$

This is commonly know as the thin-plate radial basis function. For 3D interpolation, the appropriate radial basis function to use is $\phi(\mathbf{x}) = |\mathbf{x}|^3$. Duchon did much of the early work on variational interpolation [8], and the report by Girosi, Jones and Poggio is a good entry point into the mathematics of variational interpolation [11].

Using the appropriate radial basis functions, we can then write the interpolation function in the following form:

$$f(\mathbf{x}) = \sum_{j=1}^{n} d_j \phi(\mathbf{x} - \mathbf{c}_j) + P(\mathbf{x}) \tag{5}$$

In the above equation, $\mathbf{c}_j$ are the locations of the constraints, the d_j are the weights, and $P(\mathbf{x})$ is a degree one polynomial that accounts for the linear and constant portions of f. Solving for the weights d_j and the coefficients of $P(\mathbf{x})$ subject to the given constraints yields a function that both interpolates the constraints and that minimizes equation 3. The resulting function exactly interpolates the constraints, and is not subject to approximation or discretization errors. Also, the number of weights to be determined does not grow with the size of the region of interest Ω. Rather, it is only dependent on the number of constraints.

To solve for the set of d_j that will satisfy the interpolation constraints

$$h_i = f(\mathbf{c}_i) \tag{6}$$

substitute the right side of equation 5 for $f(\mathbf{c}_i)$ giving

$$h_i \quad = \quad \sum_{j=1}^{k} d_j \phi(\mathbf{c_i} - \mathbf{c_j}) + P(\mathbf{c_i}) \tag{7}$$

In the above equation, N is the dimension of the domain of our interpolation function (either $N = 2$ or $N = 3$ in this paper). Since this equation is linear with respect to the unknowns, d_j and the coefficients of $P(\mathbf{x})$, it can be formulated as a linear system. For interpolation in 3D, let $\mathbf{c}_i = (c_i^x, c_i^y, c_i^z)$ and let $\phi_{ij} = \phi(\mathbf{c}_i - \mathbf{c}_j)$. Then this linear system can be written as follows:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

It can be show that the above system is symmetric and positive semi-definite, so there will always be a unique solution for the d_i and p_i [12]. We used symmetric LU decomposition to solve this system of equations for all of the examples shown in this paper. Our implementation to set up the system, call the LU decomposition routine and evaluate the interpolating function of equation 5 for a given $\mathbf{x}$ consists of about 100 lines of commented C++ code. This code plus the public-domain polygonalization routine described in Section 7.1 is all that is needed to create variational implicit surfaces, and all of this can be assembled in an afternoon of programming.

Two concerns that arise with such matrix systems are computation times and ill-conditioned systems. For systems with up to a few thousand centers, including all of the examples in this paper, direct solution techniques such as LU decomposition and SVD are practical. However as the system becomes larger, the amount of work required to solve the system grows as $O(k^3)$. Work by Beatson

and by Suter describe fast evaluation techniques that can be used to evaluate equation 5 in $O(1)$ rather than the $O(k)$ implied by the summation [25, 1]. The cost of solving the system can be reduced to approximately $O(k^2)$ by using the a fast evaluation technique with an iterative solution method such as biconjugate gradient, and much larger systems become feasible.

As the number of constraints grows, the condition number of the matrix in equation 8 is also likely to grow, leading to instability for some solution methods. Although we have been on the lookout, we have not noticed artifacts in our models that would arise from such ill-conditioning. Even if such problems do arise, variational interpolation is such a well-studied problem that methods exist for improving the conditioning of the system of equations [10].

4 Creating Variational Implicit Surfaces

With tools for solving the scattered data interpolation problem in hand, we now turn our attention to creating implicit functions. In this section, we will examine three ways in which to define a variational implicit surface. Common to all three approaches is the specification of zero-valued constraints through which the surface must pass. The three methods differ in specifying where the implicit function takes on positive and negative values. We will look at creating both 2D variational implicit curves and 3D variational implicit surfaces. The 2D curve examples are for illustrative purposes, and our actual goal is the creation of 3D surfaces.

4.1 Positive Interior Constraints

The left portion of Figure 1 (earlier in this paper) shows the first method of describing a variational implicit curve. Four zero-valued constraints have been placed in the plane. We call such zero-value constraints *boundary constraints* because these points will be on the boundary between the interior and exterior of the shape that is being defined. In addition to the four boundary constraints, a single constraint with a value of one is placed at the location marked with a plus sign. We construct an implicit function from these five constraints simply by invoking the 2D variational interpolation technique described in earlier sections. The interpolation method returns a set of scalar coefficients w_i that weight a collection of radially symmetric functions ϕ that are centered at the constraint positions. The implicit curve shown in the figure is given by those locations at which the variationally-defined function takes on the value zero. The function takes on positive values inside the curve and is negative at locations outside the curve. Figure 1 (right) shows a refinement of the curve that is made by adding three more boundary constraints to the original set of constraints in the left portion of the figure.

Why does a positive constraint surrounded by zero-valued constraints yield a function that is negative beyond the boundary constraints? The key is that the energy function is larger for functions that take on positive values on both sides of a zero-valued constraint. Each boundary constraint acts much like a see-saw— pull one side up and the other side tends to move down.

Creating surfaces in 3D is accomplished in exactly the same way as the 2D case. Zero-valued constraints are specified by the modeler as those 3D points through which the surfaces should pass, and positive values are specified at one or more places that are to be interior to the surface. Variational interpolation is then invoked to create a scalar-valued function over $\mathbf{R}^3$. The desired surface is simply the set of all points at which this scalar function takes on the value zero. Figure 2 (left) shows a surface that was created in this fashion by placing four zero-valued constraints at the vertices of a regular tetrahedron and placing a single positive constraint in the center of the tetrahedron. The resulting implicit surface is nearly spherical.

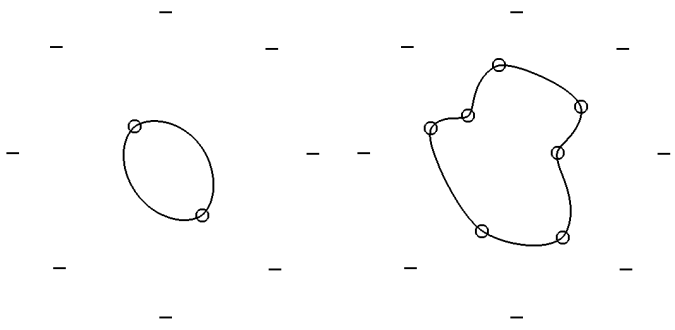

Figure 3: Curves defined using surrounding negative constraints. Just two zero-valued constraints yield an ellipse-like curve (on the left). More constraints create a more complex curve (at right).

Figure 2 (right) shows a recursive branching object that is a variational implicit surface. The basic building block of this object is a triangular prism. Each of the six vertices of a large prism specified the location of a zero-valued constraint, and a single positive constraint was placed in the center of this prism. Next, three smaller and slightly tilted prisms were placed atop the first large prism. Each of these smaller prisms, like the large one, contributes boundary constraints at its vertices and has a single positive-valued constraint placed at its center. Each of the three smaller prisms have even smaller prisms placed on top of them, and so on.

Placing one or more positive-valued constraints on the interior of a shape is an effective method of defining variational implicit surfaces when the shape one wishes to create is well-defined. We have found, however, that there is another approach that is even more flexible for interactive free-form surface sculpting.

4.2 Negative Exterior Constraints

Figure 3 illustrates a second approach to creating variational implicit functions. Instead of placing positive-valued constraints inside a shape, negative-valued constraints can be placed on the exterior of the shape that is being created. As before, zero-valued constraints specify locations through which the implicit curve will pass through. In Figure 3 (left), eight negative-valued constraints surround the region at which a curve is being created. As with positive-valued constraints, the magnitude of the values is unimportant, and we use the value negative one. These negative constraints, coupled with the curvature-minimizing nature of variational method, induce the interpolation function to take on positive values interior to the shape outlined by the zero-valued constraints. Even specifying just two boundary constraints defines a reasonable closed curve, as shown by the ellipse-like curve at the left in Figure 3. More boundary constraints result in a more complex curve, as shown on the right in Figure 3.

We have found that creating a circle or sphere of negative-valued constraints is the approach that is best suited to interactive free-form design of curves and surfaces. Once these negative constraints are defined, the user is free to place boundary constraints in any location interior to these negative constraints. Section 5 describes the use of exterior constraints for interactive sculpting.

4.3 Constraints Near the Boundary: Normal Constraints

For some applications we may have detailed knowledge about the shape that is to be modeled. In particular, we may know approximate surface normals at many locations on the surface to be created. In this case there is a third method of defining a variational

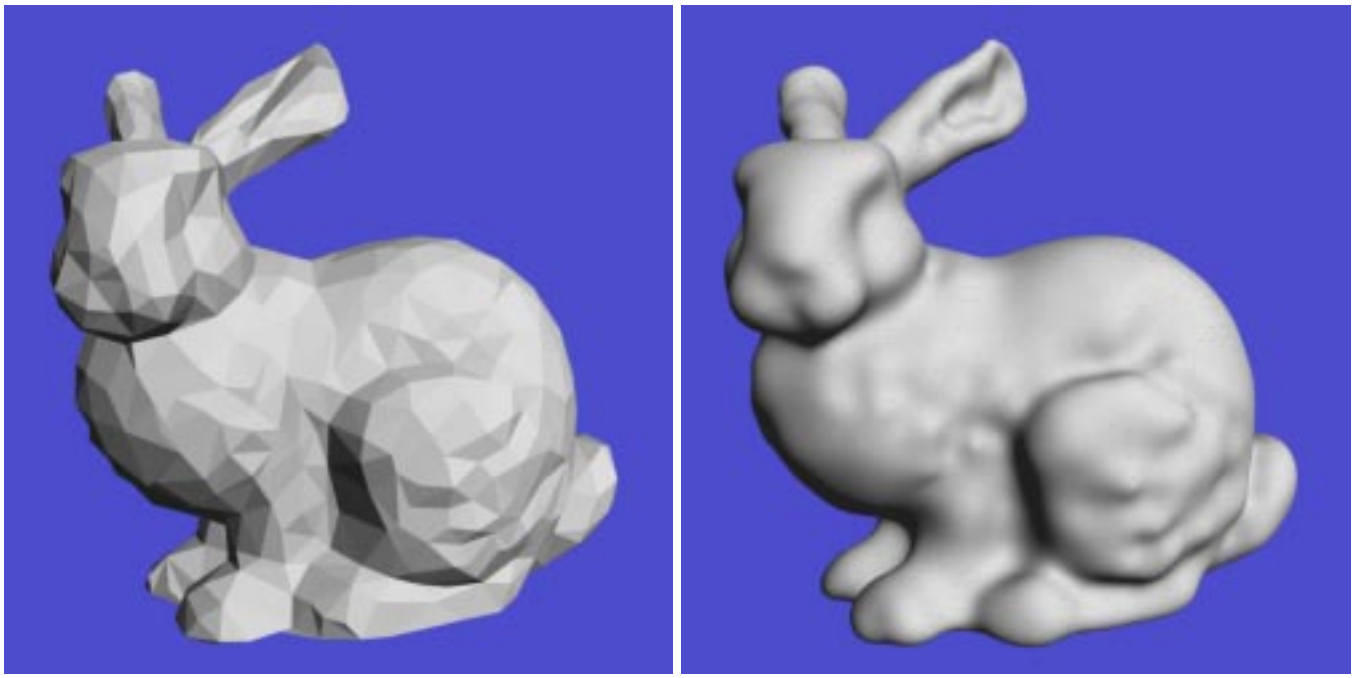

Figure 5: A polygonal surface (left) and the variational implicit surface defined by the 800 vertices and their normals (right).

implicit function that may be preferred over the two methods described above. Rather than placing positive or negative values far from the boundary constraints, we can create constraints very close to the boundary constraints. Figure 4 shows this method in the plane. In left portion of this figure, there are six boundary constraints and in addition there are six *normal constraints*. These normal constraints are positive-valued constraints that are placed very near the boundary constraints, and they are positioned towards the center of the shape that is being created. A normal constraint is created by placing a positive constraint a small distance in the direction $-\mathbf{N}$, where $\mathbf{N}$ is an approximate normal to the shape that we are creating. (Alternatively, we could choose to place negative-valued constraints in the outward-pointing direction.) A normal constraint is always paired with a boundary constraint, although not every boundary constraint requires a normal constraint. The right part of Figure 4 shows that a normal constraint can be used to bend a curve at a given point.

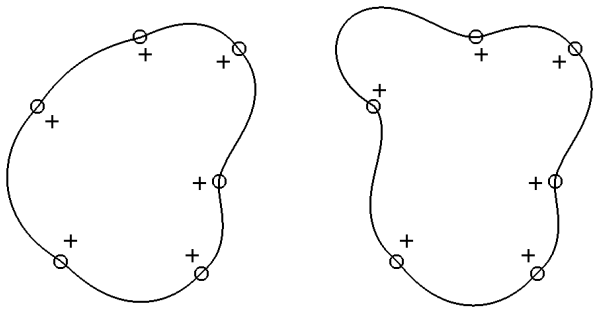

Figure 4: Two curves defined using nearly identical boundary and normal constraints. By moving just a single normal constraint (the north-west one), the curve on the left is changed to that shown on the right.

There are at least two ways in which a normal constraint might be defined. One way is to allow a user to hand-specify the surface normals of a shape that is being created. A second way allows us to create smooth surfaces based on polyhedral models. If we wish to create a variational implicit surface from a polyhedral model, we simply need to create one boundary constraint and one normal constraint for each vertex in the polyhedron. The location of a boundary constraint is given by the position of the vertex, and the location of a normal constraint is given by moving a short distance in a direction opposite to the surface normal at the vertex. We place normal constraints 0.01 units from the corresponding boundary constraints for objects that fit within a unit cube. Figure 5 (right) shows a variational implicit surface created in the manner just described from the polyhedral model in Figure 5 (left). This is a simple yet effective way to create an everywhere smooth analytically defined surface. This stands in contrast to the complications of patch stitching inherent in most parametric surface modeling approaches.

In this section we have seen three methods of creating variational implicit functions. These methods are in no way mutually exclusive, and a user of an interactive sculpting program could well use a mixture of these three techniques to define a single surface.

5 Interactive Model Building

Variational implicit surfaces seem ready-made for interactive 3D sculpting. In this section we will describe how they can be gracefully incorporated into an interactive modeling program.

In 1994, Andrew Witkin and Paul Heckbert presented an elegant method for interactive manipulation of implicit surfaces [30]. Their method uses two types of oriented particles that lie on the surface of an implicitly defined object. One class of particles, the floaters, are passive elements that are attracted to the surface of the shape that is being sculpted. Floaters repel one another in order to evenly cover the surface. Even during large changes to the surface, a nearly constant density of floaters is maintained by particle fissioning and particle death. A second type of particle, the control

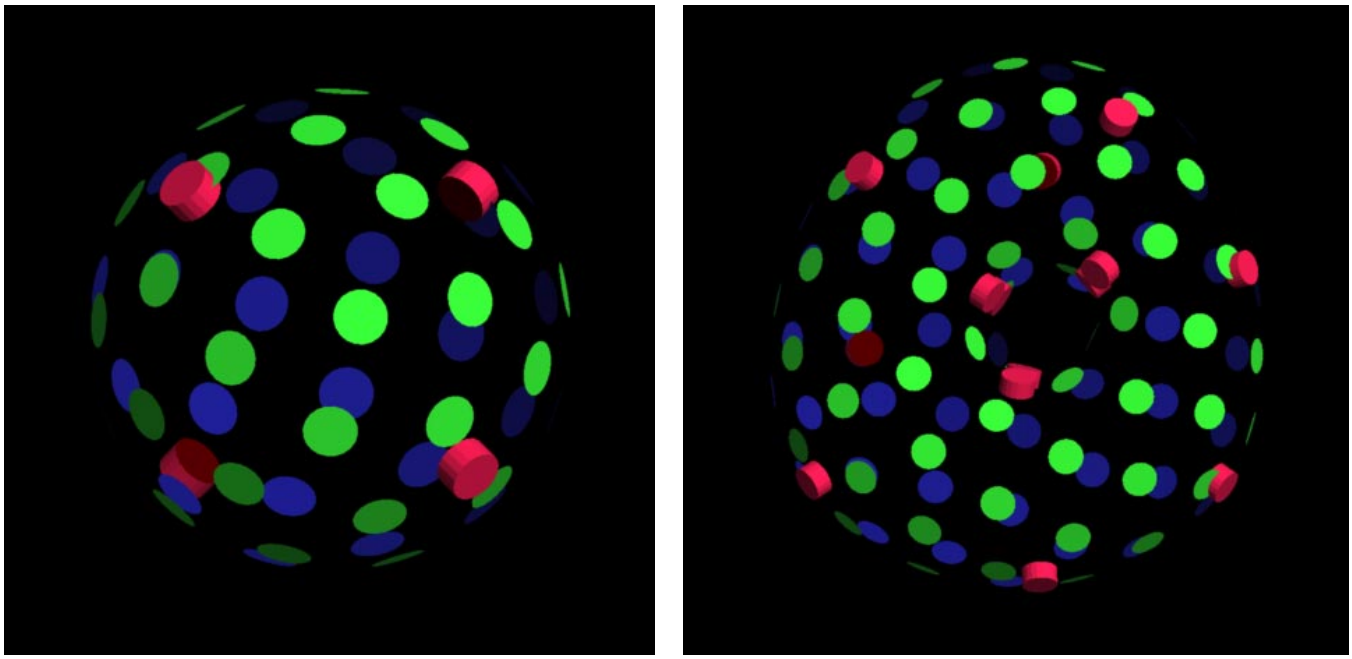

Figure 6: Interactive sculpting of variational implicit surfaces. The left image shows an initial configuration with four boundary constraints (the red markers). The right surface is a sculpted torus.

point, is the method by which a user interactively shapes an implicit surface. Control points provide the user with direct control of the surface that is being created. A control point tracks a 3D cursor position that is moved by the user, and the free parameters of the implicit function are adjusted so that the surface always passes exactly through the control point. The mathematical machinery needed to implement floaters and control points is presented clearly in Witkin and Heckbert's paper, and the interested reader should consult it for details.

The implicit surfaces used in Witkin and Heckbert's modeling program are blobby spheres and blobby cylinders. We have created an interactive sculpting program based on their particle sampling techniques, but we use variational implicit surfaces instead of blobbies as the underlying shape description. Our implementation of floaters is an almost verbatim transcription of their equations into code. The only change needed was to represent the implicit function as a sum of $\phi(\mathbf{x}) = |\mathbf{x}|^3$ radial basis functions and to provide an evaluation routine for this function and its gradient. Floater repulsion, fissioning and death work just as well as when using blobby implicit functions. As in the original system, the floaters provide a means of interactively viewing an object during editing that may even change the topology of the surface.

The main difference between our sculpting system and Witkin and Heckbert's is that we use an entirely different mechanism for direct interaction with a surface. Witkin/Heckbert control points provide an *indirect* link between a 3D cursor and the free parameters of a blobby implicit function. Because our system is based on variational implicit functions, however, we simply allow users to directly create and move boundary constraints.

We initialize a sculpting session with a simple variational implicit surface that is nearly spherical, and this is shown at the left in Figure 6. It is described by four boundary constraints at the vertices of a unit tetrahedron (the thick red disks) and with eight exterior (negative) constraints surrounding these at the corners of a cube with a side width of six. (The exterior constraints are not drawn.) A user is free to drag any of the boundary constraint locations using a 3D cursor, and the surface follows. The user may also create any number of new boundary constraints on the surface. The location of a new boundary constraint is found by intersecting the surface with a ray that passing through the camera position and the cursor. After a user creates or moves a boundary constraint, the matrix equation from Section 3 is solved anew. The floaters are then moved and displayed. The right portion of Figure 6 shows a toroidal surface that was created using this interactive sculpting paradigm. The interactive program continuously executes the following loop:

```
repeat
  create or move constraints based on user interaction
  solve new variational matrix equation
  adjust floater positions (with floater birth and death)
  render floaters
```

An important consequence of the matrix formulation given by equation 8 is that adding a new boundary constraint on the existing surface does not affect the surface shape at all. This is because the implicit function already takes on the value of zero at the surface, so adding new zero-valued constraint on the surface will not alter the weights of the old constraints. Only when such a new boundary constraint is moved does it begin to affect the shape of the surface. This ability to retain the exact shape of a surface while adding new boundary constraints is similar in spirit to knot insertion for polynomial spline curves and surfaces. We do not know of any similar capability for blobby implicit surfaces.

In addition to control of boundary constraints, we also allow a user to create and move normal constraints. By default, no normal constraint is provided for a newly created boundary constraint. At the user's request, a normal constraint can be created at any specified boundary constraint. The initial direction of the normal constraint is given by the gradient of the current implicit function. The value for such a constraint is given by the implicit function's value at the constraint location. A normal constraint is drawn as a spike that is fixed at one end to the disk of its corresponding boundary point. The user may drag the free end of this spike to adjust the normal to the surface, and the surface follows this new constraint.

What has been gained by using variational implicit functions instead of blobby spheres and cylinders? First, the variational implicit approach is easier to implement because the optimization machinery for control points of blobby implicits is not needed. Second, the user has control over the surface normal as well as the surface position. Finally, the user does not need to specify which implicit parameters are to be fixed and which are to be free at different times during the editing session. Using the blobby formulation, the user must choose at any given time which parameters such as sphere centers, radii of influence and cylinder endpoints may be altered by moving a control point. With the variational formulation, the user is always changing the position of just a single boundary or normal constraint. We believe that this direct control of the parameters of the implicit function is more natural and intuitive. Witkin and Heckbert state the following [30]:

> Another result of this work is that we have discovered that implicit surfaces are slippery: *when you attempt to move them using control points they often slip out of your grasp*. [emphasis from the original]

We have found that *variational* implicit surfaces are not at all slippery. Users easily grasp and re-shape these surfaces with no thought to the underlying parameters of the model.

6 Comparison to Related Methods

At this point it is useful to compare variational implicit surfaces to other representations of surface geometry. Although they share similarities with existing techniques, variational implicits are new and distinct from other forms of surface modeling.

6.1 Thin-Plate Surface Reconstruction

The scientific and engineering literature abound with surface reconstruction based on thin-plate interpolation. Aren't variational implicits just a slight variant on thin-plate techniques? The most important difference is that traditional thin-plate reconstruction creates a height field in order to fit a given set of data points. The use of a height field is a barrier towards creating closed surfaces and surfaces of arbitrary topology. For example, a height field cannot even represent a simple sphere-like object such as the surface shown in Figure 2 (left). Complex surfaces can be constructed using thin-plate techniques only if a number of height fields are stitched together to form a parametric quilt over the surface. Variational implicit surfaces, on the other hand, do not require multiple patches in order to represent a complex model. Both methods create a function based on variational methods, but they differ in the dimension of the scalar function that they create. Traditional thin-plate surfaces use a function with a 2D domain to create a *parametric* surface, whereas the variational implicit method uses a function with a 3D domain to specify the location of an *implicit* surface.

6.2 Sums of Implicit Primitives

Section 3 shows that a variational implicit function is in fact a sum of a number of functions that have radial symmetry (based on the $|\mathbf{x}|^3$ function). Isn't this similar to constructing an implicit function by summing a number of spherical Gaussian functions (blobby spheres or meta-balls)? Let us consider the process of modeling a particular shape using blobby spheres. The unit of construction is the single sphere, and two decisions must be made when we add new sphere to a model: the sphere's center and its radius. We cannot place the center of the sphere where we want the surface to be– we must displace it towards the object's center and adjust its radius to compensate for this displacement. What we are doing is much like guessing the location of part of the medial axis of the object that we are modeling. (The medial axis is the locus of points that are equally distant from two or more places on an object's boundary.) In fact, the task is more difficult than this because summing multiple blobby spheres is not the same as calculating the union of the spheres. The interactive method of Witkin and Heckbert relieves the user from some of this complexity, but still requires the user to select which blobby primitives are being moved and which are fixed. These issues never come up when modeling using variational implicit surfaces because we can directly specify locations that the surface must pass through.

Fitting blobby spheres to a surface is an art, and indeed many beautiful objects have been sculpted in this manner. Can this process be entirely automated? Shigeru Muraki demonstrated a way in which a given range image may be approximated by blobby spheres [18]. The method begins with a single blobby sphere that is positioned to match the data. Then the method repeatedly selects one blobby sphere and splits it into two new spheres, invoking an optimization procedure to determine the position and radii of the two spheres that best approximates the given surface. Calculating a model composed of 243 blobby spheres "took a few days on a UNIX workstation (Stardent TITAN3000 2 CPU)." Similar blobby sphere data approximation by Eric Bittar and co-workers was limited to roughly 50 blobby spheres [2]. In contrast to these methods, the bunny in Figure 5 (right) is a variational implicit surface with 800 boundary and 800 normal constraints. It required 1 minute 43 seconds to solve the matrix equation for this surface, and the iso-surface extraction required 7 minutes 43 seconds. Calculations were performed on an SGI O2 with a 195 MHz R10000 processor.

7 Rendering

In this section we examine two traditional approaches for rendering iso-surfaces that both perform well for variational implicit surfaces.

7.1 Conversion to Polygons

One way to render an implicit surface is to create a set of polygons that approximate the surface. The most common method of creating polygons for an implicit surface is to divide up a region of 3-space into regular cells such as cubes or tetrahedra. If the value of the implicit function takes on a mixture of positive and negative values at the corners of a given cell, then the implicit surface must pass through the cell. At such cells, a small set of polygons can be created that approximates the behavior of the surface within the cell. Perhaps the best known approach of this type is the Marching Cubes algorithm [17].

When an implicit function is to be extracted from a measured dataset such as from medical CT or MRI, an isosurface extraction algorithm typically examines each voxel of the given volume. For an analytically-defined implicit function such as the kind described in this paper, there is no pre-defined set of voxels to traverse. We perform iso-surface extraction using an algorithm known as a *continuation* approach. Both models in Figure 2 and the right model in Figure 5 are collections of polygons that were created using the continuation method. This method first locates any position that is on the surface to be tiled. This first point can be thought of as a single corner of a cube that is one of an infinite number of cubes in a regular lattice. The continuation method examines the values of the implicit function at neighboring points on the cubic lattice and creates polygons within each cube that the surface must pass through. The neighboring vertices of these cubes are examined in turn, and the process eventually crawls over the entire surface defined by the implicit function. We use the implementation of this method from [5] that is described in detail by Bloomenthal in [4].

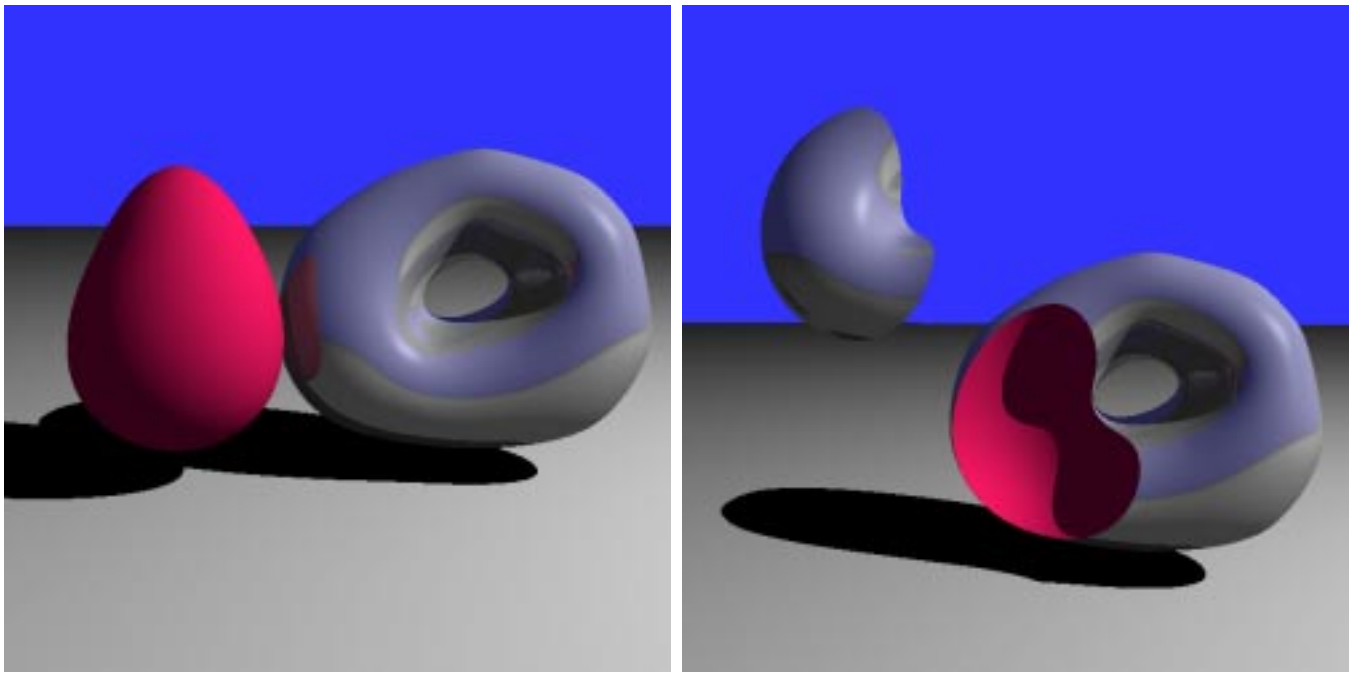

Figure 7: Ray tracing of variational implicit surfaces. The left image shows reflection and shadows of two implicit surfaces, and the right image illustrates constructive solid geometry.

7.2 Ray Tracing

There are a number of techniques that may be used to ray trace implicit surfaces, and a review of these techniques can be found in [14]. We have produced ray traced images of variational implicit surfaces using a particular technique introduced by Hart that is known as sphere tracing [15]. The basis for this method is that some implicit functions (including those that we are interested in) have what is called the *Lipschitz property*. A function f is said to have the Lipschitz property if and only if there exists some positive constant k such that:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq k\|\mathbf{x} - \mathbf{y}\| \tag{9}$$

The smallest k that satisfies the above equation is called the Lipschitz constant.

Sphere tracing is based on the idea that we can find the intersection of a ray with a surface by traveling along the ray in steps that are small enough to avoid passing through the surface. We declare that we have intersected the surface when the value of $f(\mathbf{x})$ falls below some tolerance ε. At any point $\mathbf{x}$ on a ray, we decide on step size by determining the radius of a sphere that is guaranteed not to intersect the surface. Assuming that k is the Lipschitz constant for the function f, the radius of such a "safe" sphere at $\mathbf{x}$ is $f(\mathbf{x})/k$. For variational implicit functions, we have used a simple heuristic to determine an appropriate value for k. We sample the space in and around our implicit surface at 2000 positions, and we use for k the maximum gradient magnitude over all of these locations. For extremely pathological surfaces this heuristic may fail, although it has worked well for all of our images. Our simple ray traced images should only be taken as a proof-of-concept that variational implicits can indeed be ray traced. Finding guaranteed bounds for the Lipschitz constant of a variational implicit function is a good area for future work. It is also likely that other techniques can successfully be applied to ray tracing these surfaces, such as the LG-surfaces approach of Kalra and Barr [16].

Figures 7 (left) is an image of two variational implicit surfaces that were ray traced using sphere tracing. Note that this figure includes shadows and reflections. Figure 7 (right) illustrates constructive solid geometry with variational implicit surfaces. The figure shows (from left to right) intersection and subtraction of two implicit surfaces. This figure was created using standard ray tracing CSG techniques as described in [22].

The rendering techniques of this section highlight a key point– variational implicit surfaces may be used in almost all of the contexts in which other implicit formulations have been used. This new representation may provide fruitful alternatives for a number of problems that use implicit surfaces. With this in mind, we now turn to future work.

8 Conclusion and Future Work

In this paper we have introduced variational implicit surfaces, a new paradigm for creating implicit surfaces. Specific advantages of this method include:

- Direct specification of points on the implicit surface
- Specification of surface normals
- Conversion of polygon models to smooth implicit forms
- Intuitive controls for interactive sculpting
- Addition of new control points that leave the surface unchanged (like knot insertion)

A number of techniques have been developed for working with implicit surfaces. Many of these techniques could be directly applied to variational implicits, indicating several directions for future work. The critical point analysis of Stander and Hart could be used to guarantee topologically correct tessellation of such surfaces [24]. Interval techniques, explored by Duff, Snyder and others, might be applied to tiling and ray tracing of variational implicits [9, 23]. The

interactive texture placement methods of Pedersen should be directly applicable to variational implicit surfaces [20, 21]. Finally, many marvelous animations have been produced using blobby implicit surfaces [3, 31]. We anticipate that the interpolating properties of variational implicit surfaces may provide animators with an even greater degree of control over implicit surfaces.

Beyond extending existing techniques for this new form of implicit surface, there are also research directions that are suggested by issues that are specific to our variational technique. Like blobby sphere implicits, variational implicit surfaces are everywhere smooth. Perhaps there are ways in which sharp features such as edges and corners can be incorporated into a variational implicit model. We have showed how gradients of the implicit function may be specified indirectly using positive constraints that are near zero constraints, but it may be possible to modify the approach to allow the exact specification of gradient values. Finally, these surfaces are well behaved even with sparse sets of constraints such as those found using computer vision techniques. Thin-plate methods have been used extensively in computer vision, but the resulting surfaces have been limited to a height-field topology. We plan to investigate application areas such as surface reconstruction from stereo depths that could benefit from the topological freedom of the variational implicit surface approach.

References

[1] R. K. Beatson and W. A. Light. Fast evaluation of radial basis functions: Methods for two-dimensional polyharmonic splines. *IMA Journal of Numerical Analysis*, 17:343–372, 1997.

[2] Eric Bittar, Nicolas Tsingos, and Marie-Paule Gascuel. Automatic reconstruction of unstructured 3d data: Combining a medial axis and implicit surfaces. *Computer Graphics Forum (Proceedings of Eurographics '95)*, 14(3):457–468, 1995.

[3] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

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

[5] Jules Bloomenthal. An implicit surface polygonizer. In Paul S. Heckbert, editor, *Graphics Gems IV*, pages 324–349. Academic Press, 1994.

[6] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997.

[7] George Celniker and Dave Gossard. Deformable curve and surface finite-elements for free-form shape design. *Computer Graphics (SIGGRAPH 91)*, 25(4):257–266, July 1991.

[8] J. Duchon. Spline minimizing rotation-invariant semi-norms in sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions on Several Variables, Lecture Notes in Mathematics 571*, Berlin, 1977. Springer-Verlag.

[9] Tom Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics (SIGGRAPH 92)*, 26(2):154–168, July 1992.

[10] Nira Dyn. Interpolation of scattered data by radial basis functions. In L. L. Schumaker C. K. Chui and F. I. Utreras, editors, *Topics in Multivariate Approximation*, pages 47–61. Academic Press, Inc., 1987.

[11] Federico Girosi, Michael Jones, and Tomaso Poggio. Priors, stabilizers and basis functions: from regularization to radial, tensor and additive splines. Technical report, MIT Artificial Intelligence Laboratory, June 1993. A.I. Memo No. 1430.

[12] Gene H. Golub and Charles F. van Loan. *Matrix Computations*. John Hopkins University Press, 1996.

[13] W. E. L. Grimson. Surface consistancy constraints in vision. *Computer Vision, Graphics, and Image Processing*, 24(1):28–51, October 1983.

[14] John Hart. Ray tracing implicit surfaces. *Siggraph 93 Course Notes: Design, Visualization and Animation of Implicit Surfaces*, pages 1–16, 1993.

[15] John Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, 12(10):527–545, 1997.

[16] Devendra Kalra and Alan Barr. Guarenteed ray intersection with implicit surfaces. *Computer Graphics (SIGGRAPH 89)*, 23(4):297–306, 1989.

[17] William Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3-d surface construction algorithm. *Computer Graphics (SIGGRAPH 87)*, 21(4):163–169, July 1987.

[18] Shigeru Miraki. Volumetric shape description of range data using 'blobby model'. *Computer Graphics (SIGGRAPH 91)*, 25(4):227–235, July 1991.

[19] Hitoshi Nishimura, Makoto Hirai, Toshiyuki Kawai, Toru Kawata, Isao Shirkawa, and Koichi Omura. Object modeling by distribution function and a method of image generation. *Transactions of the Institute of Electronics and Communication Engineers of Japan*, J68-D(4):718–725, 1985.

[20] Hans Pedersen. Decorating implicit surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 95)*, pages 291–300, August 1995.

[21] Hans Pedersen. A framework for interactive texturing on curved surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 96)*, pages 295–302, August 1996.

[22] Scott Roth. Ray casting as a method for solid modeling. *Computer Graphics and Image Processing*, 18(2):109–144, 1982.

[23] John Snyder. Interval analysis for computer graphics. *Computer Graphics (SIGGRAPH 92)*, 26(2):121–130, July 1992.

[24] Barton T. Stander and John C. Hart. Guaranteeing the topology of an implicit surface polygonization for interactive modeling. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 97)*, pages 279–286, August 1997.

[25] David Suter. Fast evaluation of splines using poisson formula. *International Journal of Scientific Computing and Modeling*, 1(1):70–87, 1994.

[26] Richard Szeliski. Fast surface interpolation using hierarchical basis functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(6):513–528, June 1990.

[27] Gabriel Taubin. An improved algorithm for algebraic curve and surface fitting. In *Fourth International Conference on Computer Vision (ICCV '93)*, pages 658–665, Berlin, Germany, May 1993.

[28] Demetri Terzopoulos. The computation of visible-surface representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(4):417–438, July 1988.

[29] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 94)*, pages 247–256, July 1994.

[30] Andrew P. Witkin and Paul S. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 94)*, pages 269–278, July 1994.

[31] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structures for soft objects. *The Visual Computer*, 2(4):227–234, 1986.