Appears in ACM Transactions on Graphics, Vol 21, No 4, October 2002

Modelling with Implicit Surfaces that Interpolate

Greg Turk
GVU Center, College of Computing
Georgia Institute of Technology

James F. O'Brien
EECS, Computer Science Division
University of California, Berkeley

Abstract

We introduce new techniques for modelling with *interpolating implicit surfaces*. This form of implicit surface was first used for problems of surface reconstruction [24] and shape transformation [30], but the emphasis of our work is on model creation. These implicit 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, and the iso-surface of this function describes a 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 for model creation is that interpolating implicit surfaces allow the direct specification of both the location of points on the surface and the 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 this form of implicit surface particularly attractive for interactive sculpting using the particle sampling technique introduced by Witkin and Heckbert in [32]. Our formulation also yields a simple method for converting a polygonal model to a smooth implicit model, as well as a new way to form blends between objects.

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 is about modelling with *interpolating implicit surfaces*, a surface representation that we believe will be useful in several areas in 3D modeling. These implicit surfaces are smooth, exactly pass through a set of given constraint points, and can describe closed surfaces of arbitrary topology.

In order to illustrate our basic approach, Figure 1 (left) shows an interpolating implicit curve, the 2D analog of an interpolating implicit surface. The small open circles in this figure indicate the location of constraints where the 2D implicit function must take on the value of zero. The single plus sign corresponds to an additional constraint where the implicit function must take on the value of some arbitrary positive constant, which for this example is one. These constraints are passed along to a scattered data interpolation routine that generates a smooth 2D function meeting 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 for creating a function 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 surfaces such as the branching shape in Figure 2 (right) can be defined simply by specifying more constraints. Figure 3 show an example of an interpolating implicit surface created from polygonal data.

The remainder of this paper is organized as follows. In Section 2 we examine related work, including implicit surfaces 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 interpolating implicit surfaces are well suited for interactive sculpting. In Section 6 we present a new method of creating soft blends between objects, based on interpolating implicits. Section 7 describes two rendering techniques, one that relies on polygonal tiling and another based on ray tracing. In Section 8 we compare interpolating implicit surfaces with traditional thin-plate surface modeling and with implicit functions that are created using ellipsoidal Gaussian functions. Finally, Section 9 indicates potential applications and directions for future research.

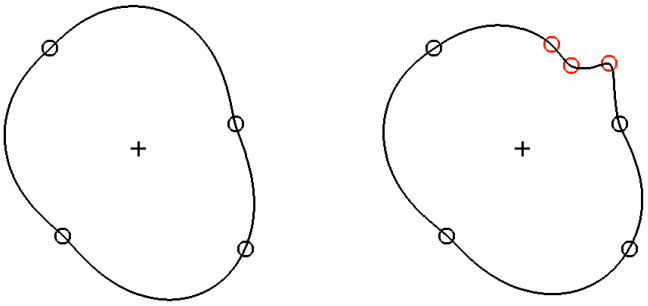

Figure 1: Curves defined using interpolating implicit functions. The curve on the left is defined by four zero-valued and one positive constraint. This curve is refined by adding three new zero-valued constraints (shown in red at right).

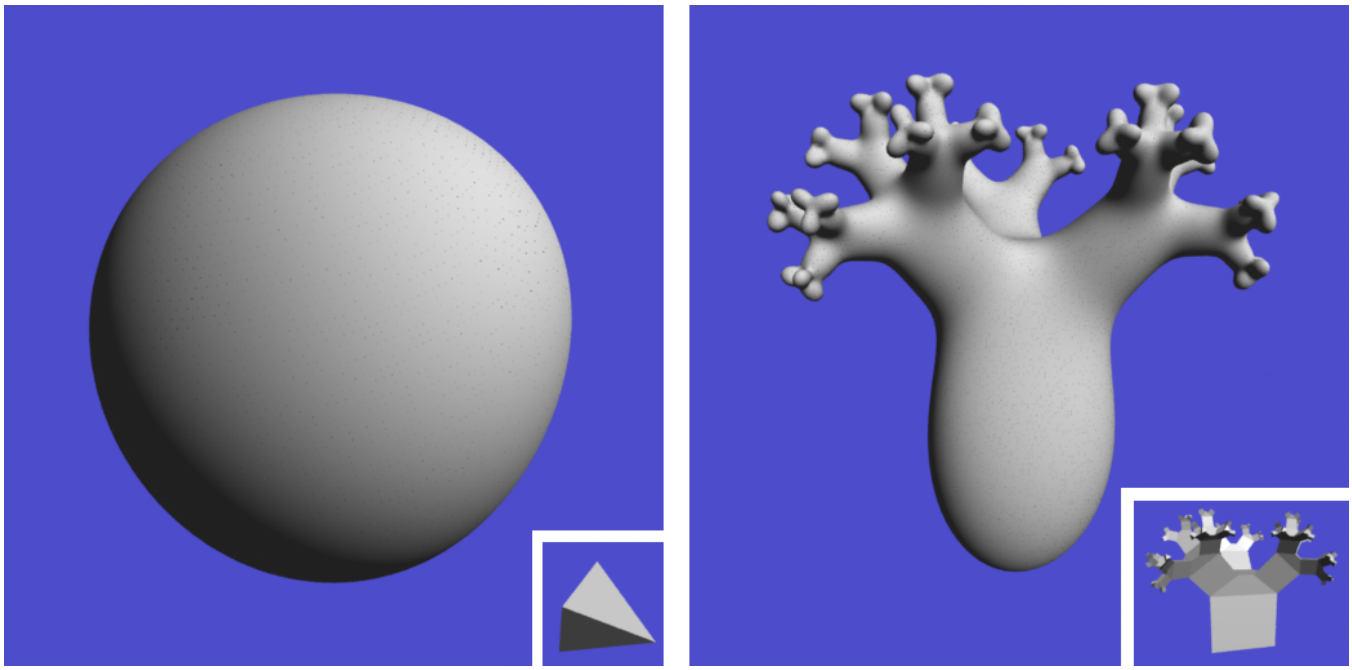

Figure 2: Surfaces defined by interpolating implicit functions. The left surface is defined by zero-valued constraints at the corners of a tetrahedron and one positive constraint in its center. The branching surface at the right was created using constraints from the vertices of the inset polygonal object.

2 Background and Related Work

Interpolating 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. Interpolating implicit surfaces are not new to graphics, and at the close of this section we describe earlier published methods of creating interpolating implicit surfaces.

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 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 $f(\mathbf{x}) = 1 - |\mathbf{x}|$, for points $\mathbf{x} \in \mathbf{R}^3$. Points on the sphere are those locations at which $f(\mathbf{x}) = 0$. This implicit function takes on positive values inside 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* [2, 20]. 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:

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

In the above equation, a single function g_i describes the profile of a "blobby sphere" (a Gaussian function) that has a particular center and standard deviation. 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, and it specifies one particular surface from a family of nested surfaces that are defined by the sum of Gaussians. 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 g_i is the following:

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

In this equation, the constant σ_i specifies the standard deviation of the 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 [20, 33]. A greater variety of shapes can be created with the blobby approach by using ellipsoidal rather than spherical 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 we recommend Gabriel Taubin [28] and Keren and Gotsman [16] as starting points in this area. Much of the work on this method has been devoted to fitting an algebraic surfaces to a given collection of points. Usually it is not possible to interpolate 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 [5]. It is easier to create complex surfaces using a collection of algebraic patches rather than using a single algebraic surface. The tradeoff, however, is that a good deal of machinery is required to create smooth joins across patch boundaries.

We have only described some of the implicit surface representations that are most closely related to our own work. There are many other topics within the broad area of implicit surfaces, and we refer the interested reader to the excellent book by Bloomenthal and his co-authors [5].

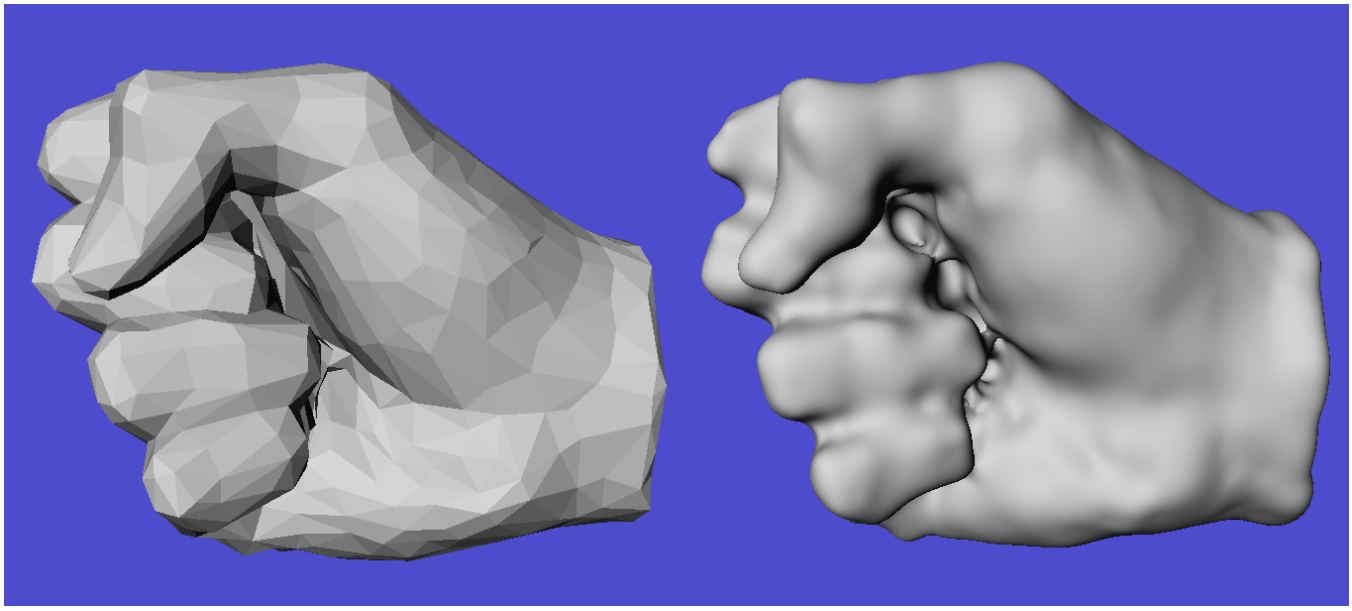

Figure 3: Polygonal surface of a human fist with 750 vertices (left) and an interpolating implicit surface created from the polygons (right).

2.2 Thin-Plate Interpolation

Thin-plate spline surfaces are a class of height fields that are closely related to the interpolating 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, there is a unique solution to such a problem, and this solution is the thin-plate interpolation of the points. Consider the energy function $E(f)$ that measures the smoothness of a function f:

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

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 (a portion of the plane). Any creases or pinches in a surface will result in a larger value of E. A smooth function 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 thin-plate surfaces are 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 [12, 29]. 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.

2.3 Related Work on Implicit Surfaces

The first publication on interpolating implicits of which we are aware is that of Savchenko et al. [24]. We consider this to be a pioneering paper in implicit surfaces, and feel it deserves to be known more widely than it is at present. Their research was on the creation of implicit surfaces from measured data such as range data or contours. Their work did not, however, describe techniques for modelling. Their approach to implicit function creation is similar to our method in the present paper in that both solve a linear system to get the weights for radial basis functions. The work of [24] differs from our own in that they use a *carrier solid* to suggest what part of space should be interior to the surface that is being created. We believe that the three methods that we describe for defining the interior of a surface in Section 4 of this paper give more user control than a carrier solid and are thus more appropriate for modelling.

The implicit surface creation methods described in this paper are an outgrowth of earlier work in shape transformation by Turk and O'Brien [30]. They created implicit functions in $n+1$ dimensions to interpolate between pairs of n-dimensional shapes. These implicit functions were created using the *normal constraint* formula-

tion of interpolating implicit surfaces, as described in Section 4.3 of this paper. The present paper differs from that of [30] in its introduction of several techniques for defining interpolating implicit surfaces that are especially useful for model creation.

Recently techniques have developed that allow the methods discussed above to be applies to system with a large numbers of constraints [19, 6]. The work of Morse et al. [19] uses Gaussian-like compactly supported radial basis functions to accelerate the surface building process, and they are able to create surfaces that have tens of thousands of constraints. Carr et al. use fast evaluation methods to reconstruct surfaces using up to a half millions basis functions [6]. They use the radial basis function $\phi(\mathbf{x}) = |\mathbf{x}|$, the biharmonic basis function. Both of these improvements for creating surfaces with many constraints are complementary to the work of the present paper, and the new techniques that we describe in Sections 4, 5 and 6 should work gracefully with the methods in both of these papers.

3 Variational Methods and Radial Basis Functions

In this section we review the necessary mathematical background for thin-plate interpolation. This will provide the tools that we will then use in Section 4 to create interpolating implicit surfaces.

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 of cells or elements and define local basis functions over the elements. The function $f(\mathbf{x})$ can then be 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 [29, 27, 7, 31]. While finite elements and finite differencing techniques have proven useful for many problems, the fact that they rely on discretization of the function's domain is not always ideal. Problems that can arise due to discretization include visibly stair-stepped surfaces and the inability to represent fine details. In addition, the cost of using such methods grows cubically as the desired resolution grows.

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, one commonly used radial basis function is $\phi(\mathbf{x}) = |\mathbf{x}|^3$, and this is the basis function that we use. We note that Carr et al. [6] used the basis function $\phi(\mathbf{x}) = |\mathbf{x}|$. 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 write the interpolation function in this form:

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

In the above equation, $\mathbf{c}_j$ are the locations of the constraints, the w_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 w_j and the coefficients of $P(\mathbf{x})$ subject to the given constraints yields a function that both interpolates the constraints and minimizes equation 3. The resulting function exactly interpolates the constraints (if we ignore numerical precision issues), 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 w_j that will satisfy the interpolation constraints, we begin with the criteria that the surface must interpolate our constraints:

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

We now substitute the right side of equation 5 for $f(\mathbf{c}_i)$ to give us:

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

Since the above equation is linear with respect to the unknowns, w_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:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1k} & 1 & c_1^x & c_1^y & c_1^z \\ \phi_{21} & \phi_{22} & \dots & \phi_{2k} & 1 & c_2^x & c_2^y & c_2^z \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{k1} & \phi_{k2} & \dots & \phi_{kk} & 1 & c_k^x & c_k^y & c_k^z \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \\ c_1^x & c_2^x & \dots & c_k^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_k^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_k^z & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_k \\ p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_k \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{8}$$

The sub-matrix in equation 8 consisting of the ϕ_{ij}'s is conditionally positive-definite on the subspace of vectors that are orthogonal to the lasts four rows of the full matrix, so equation 8 is guaranteed to have a solution. 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 interpolating implicit surfaces.

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 $O(k^3)$. We have used direct solution methods for systems with up to roughly 3,000 constraints. LU decomposition becomes impractical for more constraints than this. We are pleased that other researchers, notably the authors of [19, 6], have begun to address this issue of computational complexity.

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. For the systems we have worked with, ill-conditioning has not been a problem. If problems arose for larger systems, variational interpolation is such a well-studied problem that methods exist for improving the conditioning of the system of equations [10].

4 Creating Interpolating 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 interpolating 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. These methods are based on using three different kinds of constraints: *interior*, *exterior*, and *normal constraints*. We will look at creating both 2D interpolating implicit curves and 3D interpolating implicit surfaces. The 2D curve examples are for illustrative purposes, and our actual goal is the creation of 3D surfaces.

4.1 Interior Constraints

The left portion of Figure 1 (earlier in this paper) shows the first method of describing a interpolating 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 use the term *interior constraint* when referring to such a positive valued constraint that helps to determine the interior of the surface. 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 an interior 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 the surface up on one side of a boundary constraint (using an interior constraint) 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 interior constraint in the center of the tetrahedron. The resulting implicit surface is nearly spherical.

Figure 2 (right) shows a recursive branching object that is a interpolating 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 interior 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 interior constraint placed at its center. Each of the three smaller prisms have even smaller prisms placed on top of them, and so on.

Why does this method of creating an implicit function create a smooth surface? We are creating the scalar-valued function in 3D that matches our constraints and that minimizes a 3D energy functional similar to Equation 3. This energy functional selects a smoothly changing implicit function that matches the constraints. The iso-surface that we extract from such a smoothly changing function will almost always be smooth as well. It is *not* the case in general, however, that this iso-surface is also the minimum of a curvature-based functional over surfaces. Satisfying the 3D energy functional does not give any guarantee about the smoothness of the resulting 2D surface.

Placing one or more positive-valued constraints on the interior of a shape is an effective method of defining interpolating 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 Exterior Constraints

Figure 4 illustrates a second approach to creating interpolating 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. We call each such negative-valued constraint an *exterior constraint*. As before, zero-valued constraints specify locations through which the implicit curve will pass through. In Figure 4 (left), eight exterior 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 exterior 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 4. More boundary constraints result in a more complex curve, as shown on the right in Figure 4.

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 exterior constraints are defined, the user is free to place boundary constraints in any location interior to this cage of exterior constraints. Section 5 describes the use of exterior constraints for interactive sculpting.

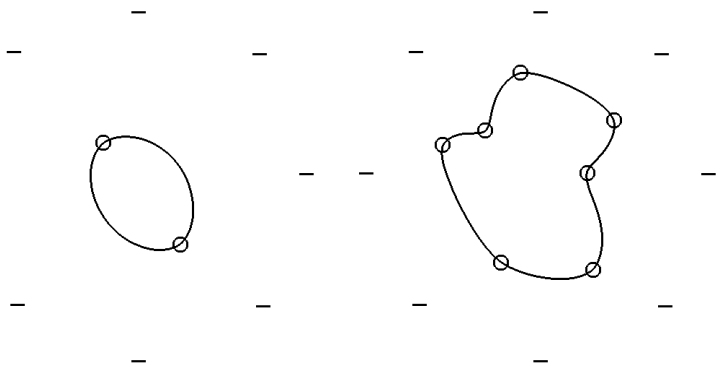

Figure 4: Curves defined using surrounding exterior constraints. Just two zero-valued constraints yield an ellipse-like curve (on the left). More constraints create a more complex curve (at right).

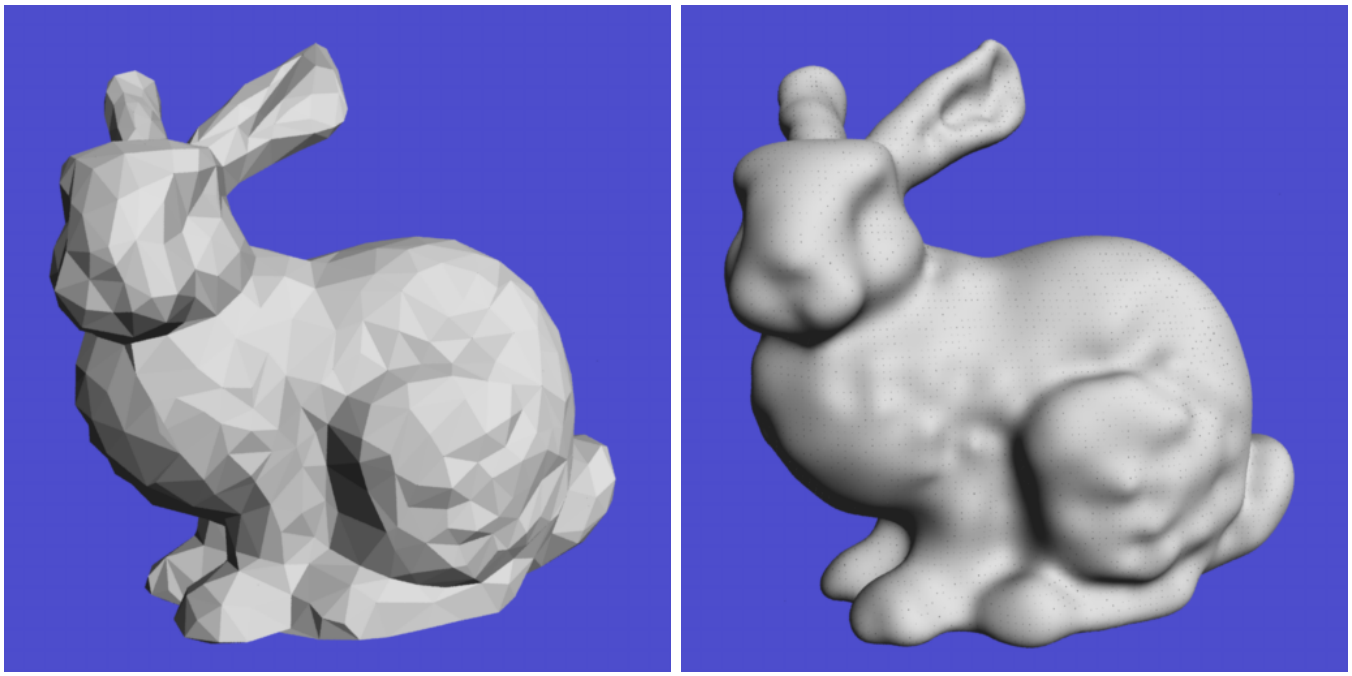

Figure 5: A polygonal surface (left) and the interpolating implicit surface defined by the 800 vertices and their normals (right).

4.3 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 interpolating implicit function that may be preferred over the two methods described above, and this method was originally described in [30]. Rather than placing positive or negative values far from the boundary constraints, we can create constraints very close to the boundary constraints. Figure 6 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 6 shows that a normal constraint can be used to bend a curve at a given point.

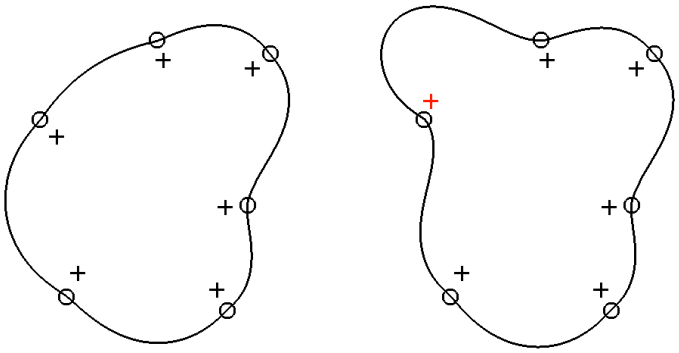

Figure 6: Two curves defined using nearly identical boundary and normal constraints. By moving just a single normal constraint (the north-west one, shown in red), the curve on the left is changed to that shown on the right.

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 interpolating 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 interpolating 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. Figure 3 is another example of converting polygons (a fist) to an implicit surface.

4.4 Review of Constraint Types

In this section we have seen three methods of creating interpolating 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. Table 4.4 lists each of the three kinds of constraints, when we believe each is appropriate to use, and which figures in this paper were created using each of the methods.

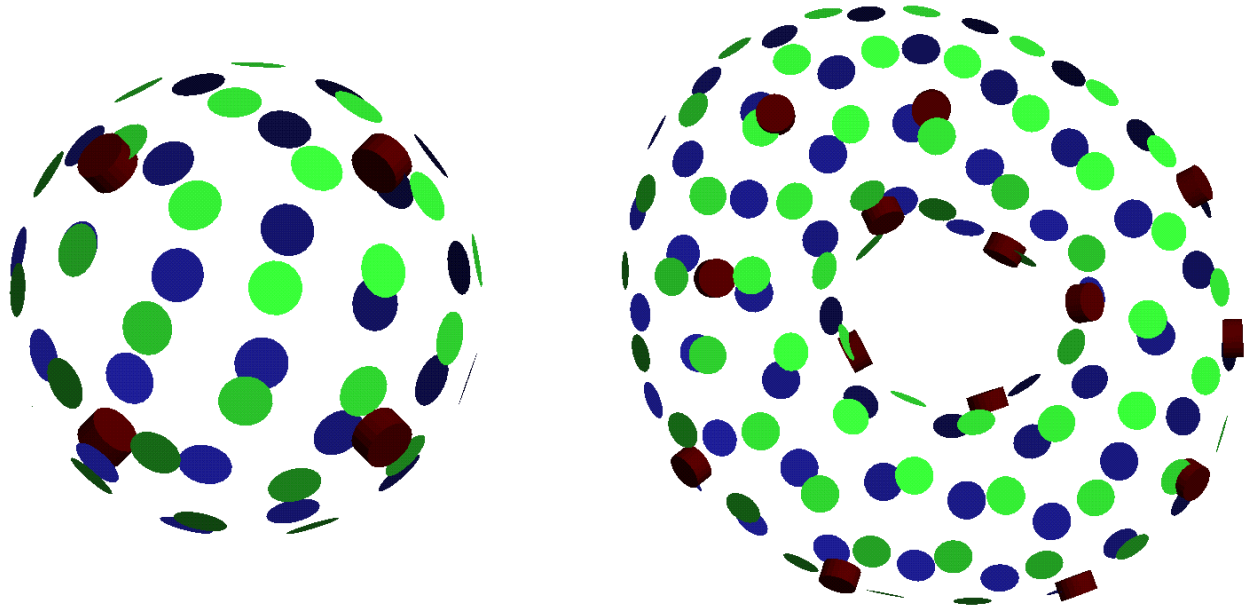

Figure 7: Interactive sculpting of interpolating implicit surfaces. The left image shows an initial configuration with four boundary constraints (the red markers). The right surface is a sculpted torus.

Constraint Types	When to Use	2D Figure	3D Figure
Interior constraints	Planned model construction	Figure 1	Figure 2
Exterior constraints	Interactive modelling	Figure 4	Figures 7, 8, 10
Normal constraints	Conversion from polygons	Figure 6	Figures 3, 5, 9

Table 1: Constraint Types

5 Interactive Model Building

Interpolating 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 [32]. 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 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 interpolating 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 for interpolating implicits 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. We do *not* make use of Witkin and Heckbert's control particles in our interactive modelling program. Instead, we simply allow users to create and move the boundary constraints of an interpolating implicit surface. This provides a direct way to manipulate the surface.

We initialize a sculpting session with a simple interpolating implicit surface that is nearly spherical, and this is shown at the left in Figure 7. 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 7 shows a toroidal surface that was created using this interactive sculpting paradigm. The interactive program repeatedly executes the following steps:

1. Create or move constraints based on user interaction.
2. Solve new variational matrix equation.
3. Adjust floater positions (with floater birth and death).

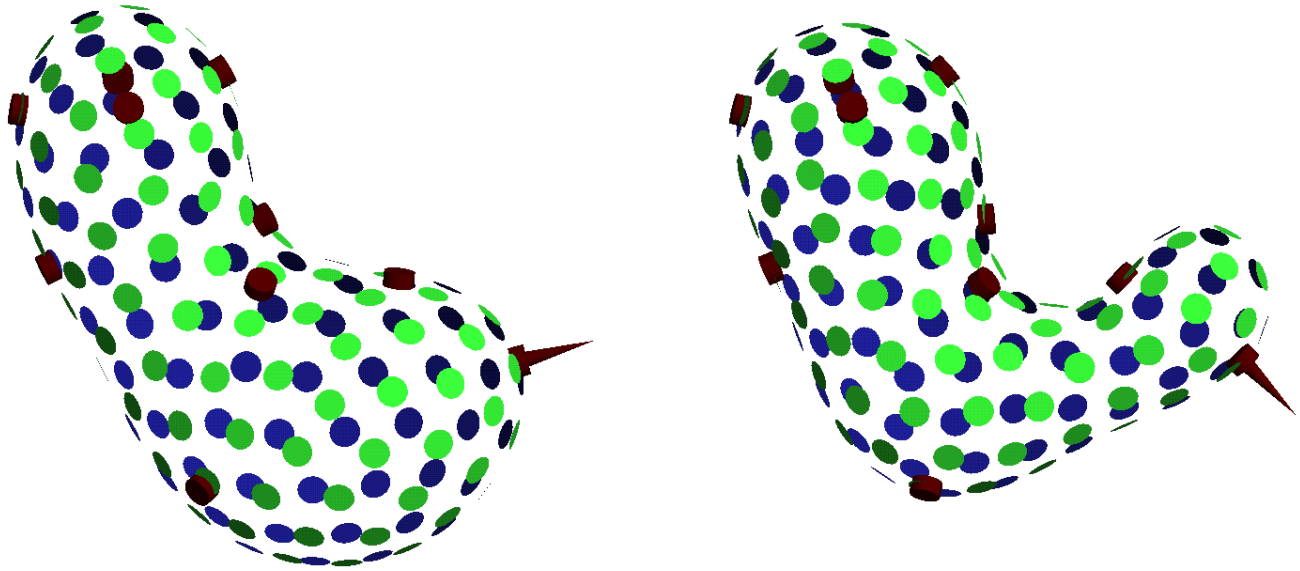

Figure 8: Changing a normal constraint. Left image shows original surface, and right image shows the same surface after changing a normal constraint (shown as a red spike).

4. 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 surface. 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. Figure 8 shows an example of changing a normal constraint during an interactive modelling session.

What has been gained by using interpolating implicit functions instead of blobby spheres and cylinders? First, the interpolating 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 [32]:

> 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 paper)

In contrast to blobby implicits, we have found that *interpolating* implicit surfaces are not at all slippery. Users easily grasp and reshape these surfaces with no thought to the underlying parameters of the model.

6 Object Blending

A *blend* is a portion of a surface that smoothly joins two sub-parts of an object. One of the more useful attributes of implicit surfaces is the ease with which they allow two objects to be blended together. Simply summing together the implicit functions for two objects often gives quite reasonable results for some applications. In some instances, however, traditional implicit surface methods have been found to be problematic when creating certain kinds of blends. For example, it is difficult to get satisfactory results when summing together the implicit functions for two branches and a trunk of a tree. The problem is that the surface will bulge at the location where the trunk and the two branches join. Bulges occur because the contribution of multiple implicit functions causes their sum to take on large values in the blend region, and this results in the new function reaching the iso-surface threshold in locations further away from the blend than is desirable. Several solutions have been proposed for this problem of bulges in blends, but these methods are either computationally expensive or are fairly limited in the geometry for which they can be used. For an excellent description of various blending methods, see Chapter 7 of [5].

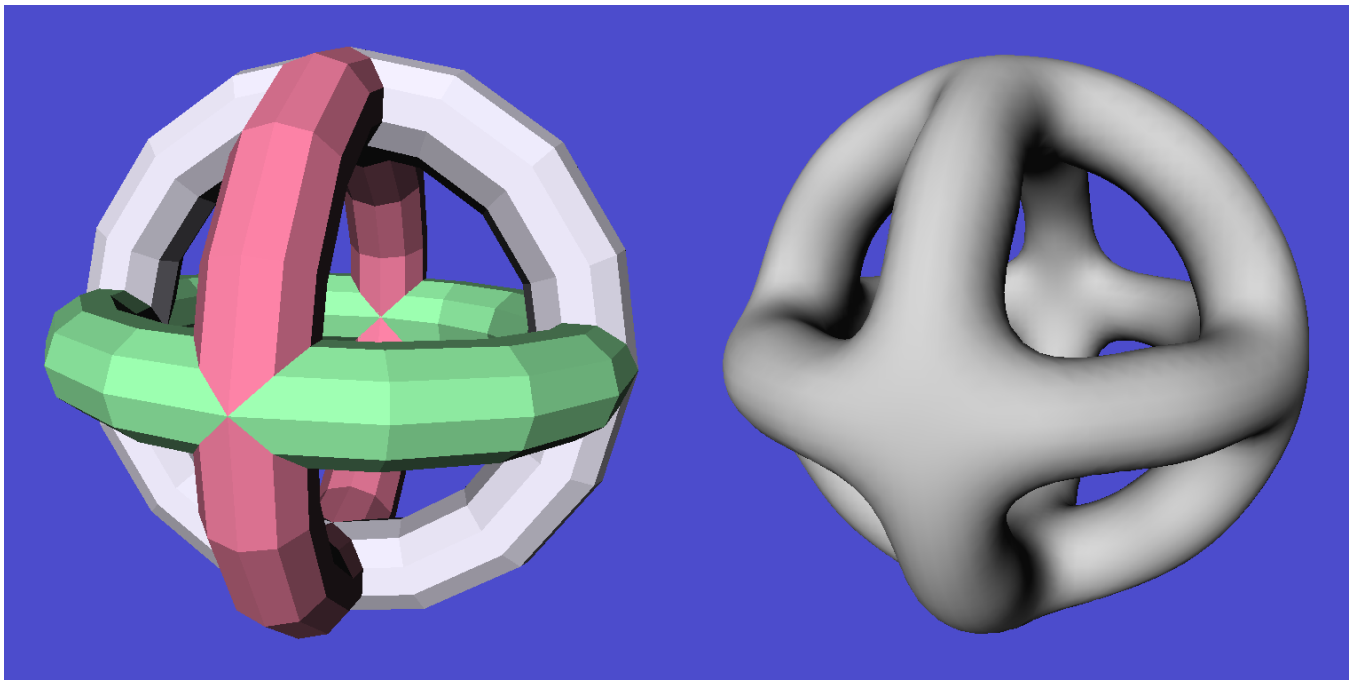

Figure 9: Three polygonal tori (left), and the soft union created with interpolating implicits (right).

Interpolating implicit surfaces provide a new way in which to create blends between objects. Objects that are blended using this new approach are free of the bulging problems found using some other methods. Our approach to blending together surfaces is to form one large collection of constraints by collect together the constraints that define of all the surfaces to be blended. The new blended surface is the surface defined by this new collection of constraints. It is important to note that simply using *all* of the constraints from the original surfaces will usually produce poor results. The key to the success of this approach is to throw out those constraints that would cause problems.

Consider the task of blending together two shapes A and B. If we used all of the constraints from both shapes, the resulting surface is not likely to be what we wish. The task of selecting which constraints to keep is simple. Let $f_A(\mathbf{x})$ and $f_B(\mathbf{x})$ be the implicit functions for shapes A and B respectively. We will retain those constraints from object A that are outside of B. That is, a constraint from A with position $\mathbf{c_i}$ will be kept if $f_B(\mathbf{c_i}) < 0$. All other constraints from A will be discarded. Likewise, we will keep only those constraints from object B that are outside of object A. To create a blended shape, we collect together all of the constraints that pass these two tests and form a new surface based on these constraints.

This approach can used to blend together any number of objects. Figure 9 (left) shows three polygonal tori that overlap one another in 3D. To blend these objects together, we first create a set of boundary and normal constraints for each object, using the approach described in Section 4.3. We then keep only those constraints from each object that are outside of each of the other two objects, as determined by their implicit functions. Finally, we create a single implicit function using all of the constraints from the three objects that were retained. Figure 9 (right) shows the result of this procedure. Notice that there are no bulges in the locations where the tori meet.

7 Rendering

In this section we examine two traditional approaches for rendering implicit surfaces that both perform well for interpolating implicits.

7.1 Conversion to Polygons

One way to render an implicit surface is to create a set of polygons that approximate the surface and then render these polygons. The topic of iso-surface extraction is well-studied, especially for regularly sampled volumetric data. Perhaps the best known approach of this type is the Marching Cubes algorithm [17], but a number of variants of this method have been described since the time of its publication.

We use a method of iso-surface extraction known as a *continuation* approach [3] for many of the figures in this paper. The models in Figure 2 and in the right images of Figures 5 and 9 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 then 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 [4] that is described in detail by Bloomenthal in [3].

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 [13]. We have produced ray traced images of interpolating implicit surfaces using a particular technique introduced by Hart that is known as *sphere tracing* [14]. Sphere tracing is based on the idea that we can find the intersection of a ray with a surface by traveling

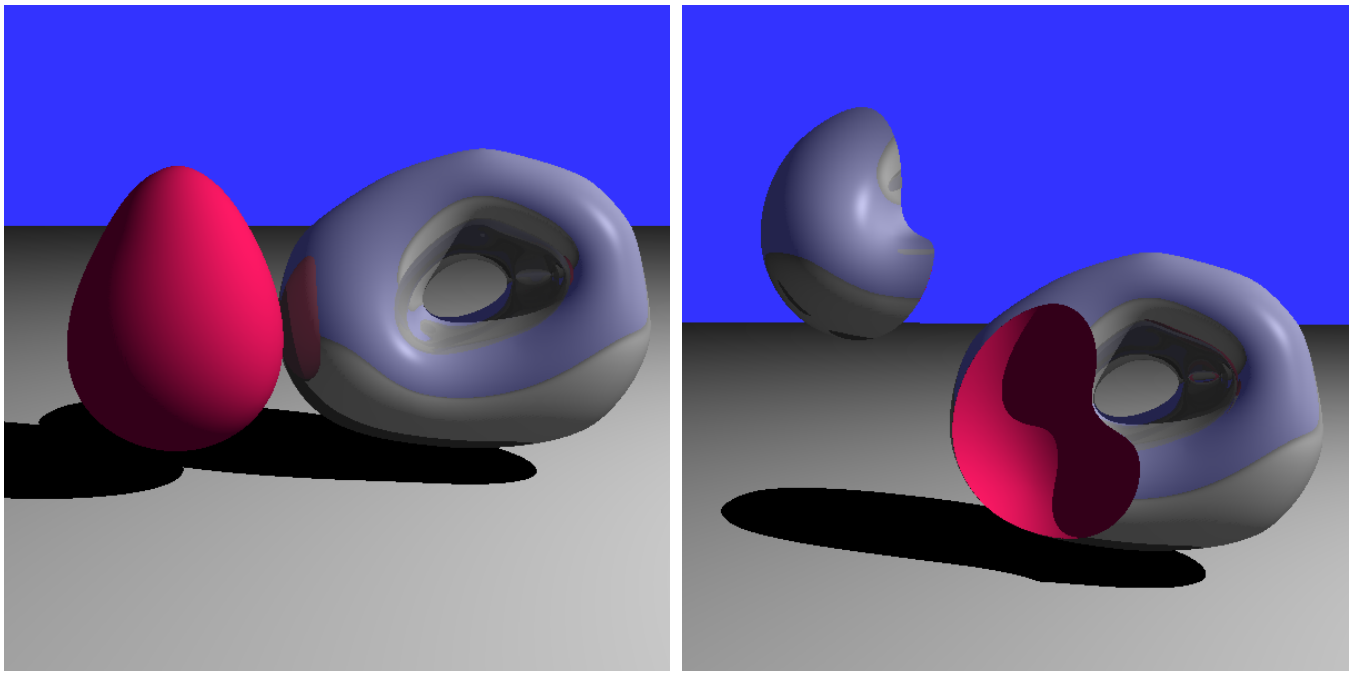

Figure 10: Ray tracing of interpolating implicit surfaces. The left image shows reflection and shadows of two implicit surfaces, and the right image illustrates constructive solid geometry.

along the ray in steps that are small enough to avoid passing through the surface. At each step along the ray the method conservatively estimates the radius of a sphere that will not intersect the surface. We declare that we are near enough to the surface when the value of $f(\mathbf{x})$ falls below some tolerance ε. We currently use a heuristic to determine the radius of the spheres during ray tracing. We sample the space in and around our implicit surface at 2000 positions, and we use the maximum gradient magnitude over all of these locations as the Lipschitz constant for sphere tracing. For extremely pathological surfaces this heuristic may fail, although it has worked well for all of our images. Coming up with a sphere radius that is guaranteed not to intersect the surface is a good area for future research. We think it is likely that other ray tracing techniques can also be successfully applied to ray tracing of interpolating implicits, such as the LG-surfaces approach of Kalra and Barr [15].

Figures 10 (left) is an image of two interpolating implicit surfaces that were ray traced using sphere tracing. Note that this figure includes shadows and reflections. Figure 10 (right) illustrates constructive solid geometry with interpolating 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 [23].

The rendering techniques of this section highlight a key point – interpolating 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.

8 Comparison to Related Methods

At this point it is useful to compare interpolating implicit surfaces to other representations of surface geometry. Although they share similarities with existing techniques, interpolating implicits are distinct from other forms of surface modeling. Because interpolating implicits are not yet well known, we provide a comparison of them to two more well-known modelling techniques.

8.1 Thin-Plate Surface Reconstruction

The scientific and engineering literature abound with surface reconstruction based on thin-plate interpolation. Aren't interpolating 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. This also presupposes that the topology of the shape to be modelled is already known. Interpolating 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 interpolating implicit method uses a function with a 3D domain to specify the location of an *implicit* surface.

8.2 Sums of Implicit Primitives

Section 3 shows that a interpolating 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 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 interpolating 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 [1]. In contrast to these methods, the bunny in Figure 5 (right) is a interpolating 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.

9 Conclusion and Future Work

In this paper we have introduced new approaches for model creation using interpolating 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 new approach to blending objects

A number of techniques have been developed for working with implicit surfaces. Many of these techniques could be directly applied to interpolating 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 [26]. Interval techniques, explored by Duff, Snyder and others, might be applied to tiling and ray tracing of interpolating implicits [9, 25]. The interactive texture placement methods of Pedersen should be directly applicable to interpolating implicit surfaces [21, 22]. Finally, many marvelous animations have been produced using blobby implicit surfaces [2, 33]. We anticipate that the interpolating properties of these 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 technique. Like blobby sphere implicits, interpolating implicit surfaces are everywhere smooth. Perhaps there are ways in which sharp features such as edges and corners can be incorporated into a interpolating 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 the gradient.

Another direction for future research is to find higher-level interactive modelling techniques for creating these implicit surfaces. Perhaps several new constraints could be created simultaneously, maybe arranged in a line or in a circle for greater surface control. It might also make sense to be able to move the positions of more than one constraint at a time. Another modelling issue is the creation of surfaces with boundaries. Perhaps a second implicit function could specify the presence or absence of a surface. Another issue related to interactivity is the possibility of displaying the surface with polygons rather than with floaters. With sufficient processor power, creating and displaying a polygonal isosurface of the implicit function could be done at interactive rates.

10 Acknowledgments

This work was funded under ONR grant N00014-97-0223. We thank the members of the Georgia Tech Geometry Group for their ideas and enthusiasm. Thanks also goes to Victor Zordan for helping with video.

References

[1] 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.

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

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

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

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

[6] Jonathan C. Carr, Tim J. Mitchell, Richard K. Beatson, Jon B. Cherrie, W. Richard Fright, Bruce C. McCallum, and Tim R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 2001)*, pages 67–76, August 2001.

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

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

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

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

[16] D. Keren and C. Gotsman. Tight fitting of convex polyhedral shapes. *International Journal of Shape Modeling*, pages 111–126, 1998.

[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] Bryan Morse, Terry S. Yoo, Penny Rheingans, David T. Chen, and K.R. Subramanian. Interpolating implicit surfaces from scattered surface data using compactly supported radial basis functions. *Shape Modelling International*, May 2001.

[20] 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.

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

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

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

[24] Vladimir V. Savchenko, Alexander A. Pasko, Oleg G. Okunev, and Tosiyasu L. Kunni. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, October 1995.

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

[26] 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.

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

[28] 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.

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

[30] Greg Turk and James O'Brien. Shape transformation using variational implicit functions. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH 1999)*, pages 335–342, August 1999.

[31] 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.

[32] 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.

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