Interpolating Implicit Surfaces From Scattered Surface Data Using Compactly Supported Radial Basis Functions

Bryan S. Morse[1], Terry S. Yoo[2], Penny Rheingans[3], David T. Chen[2], K. R. Subramanian[4]

[1]Department of Computer Science, Brigham Young University

morse@byu.edu

[2]Office of High Performance Computing and Communications, National Library of Medicine

{yoo | dave}@nlm.nih.gov

[3]Department of Computer Science and Electrical Engineeering, University of Maryland Baltimore County

rheingan@cs.umbc.edu

[4]Department of Computer Science, University of North Carolina at Charlotte

krs@cs.uncc.edu

Abstract

We describe algebraic methods for creating implicit surfaces using linear combinations of radial basis interpolants to form complex models from scattered surface points. Shapes with arbitrary topology are easily represented without the usual interpolation or aliasing errors arising from discrete sampling. These methods were first applied to implicit surfaces by Savchenko, et al. and later developed independently by Turk and O'Brien as a means of performing shape interpolation. Earlier approaches were limited as a modeling mechanism because of the order of the computational complexity involved. We explore and extend these implicit interpolating methods to make them suitable for systems of large numbers of scattered surface points by using compactly supported radial basis interpolants. The use of compactly supported elements generates a sparse solution space, reducing the computational complexity and making the technique practical for large models. The local nature of compactly supported radial basis functions permits the use of computational techniques and data structures such as k-d trees for spatial subdivision, promoting fast solvers and methods to divide and conquer many of the subproblems associated with these methods. Moreover, the representation of complex models permits the exploration of diverse surface geometry. This reduction in computational complexity enables the application of these methods to the study of shape properties of large complex shapes.

1 Introduction

As a research field of growing interest, implicit surface representations have a colorful history, with their foundations established early in the development of 3D curves and surfaces in computer graphics [2]. However, the computation of implicit surfaces has often been hampered by the constraints of available processing power and the limited complexity of the models that can be created. As CPU speeds, available memory, and computing costs have evolved, more complex models and techniques have become possible, spurring general interest in implicit surfaces [3, 6]. However, managing complex models remains difficult. The core research in modeling with implicit surfaces centers around primitives that do not always facilitate the control of the exact surface. While alternatives to surface control through simple primitives exist, they also have drawbacks. In their recent work on adaptive particle sampling and control of implicit surfaces, Heckbert and Witkin [18] report that particle control of such surfaces is slippery and elusive.

Recent growth in level set techniques [9, 12] and variational methods [7] have created new interest in understanding and manipulating complex surface models directly from the surface representation. Whitaker and Breen [17] have shown how level set techniques can be used to model and manipulate computer graphic shapes, effectively morphing from one to another. They characterize the level set as an implicit representation where the primitives are distributed throughout an active volumetric cloud layer near the surface boundary. They utilize Sethian's notion of the active set, to reduce the size of the volume representation to a sparse collar of primitives. However, even with this reduction, the representations are cumbersome, and the bookkeeping to support these methods are intricate and difficult to maintain.

Other recent work has developed techniques for interpolating an implicit surface directly from surface point data [11, 14]. This work provides some insight into how to

Proceedings of the International Conference on Shape Modeling and Applications (SMI '01)
0-7695-0853-7/01 $10.00 © 2001 IEEE

manage and employ a collection of implicit primitives while simultaneously directly controlling the surface parameters. This method allows direct specification of a complex surface from sparse, irregular surface samples. The method is quite flexible and has been extended to higher dimensions to support shape interpolation [15]. However, because of computational and storage complexity, the technique as described cannot be used to model surfaces where large numbers of surface points are included, making it unsuitable for applications where range data or tomographic reconstruction often lead to data described by hundreds of thousands of surface points.

This paper primarily addresses the topic of computational complexity. We explore an adaptation of the methods in [11, 14], applying compactly supported radial basis functions [16] to create an efficient algorithm for computing interpolated surfaces. Our technique produces significant improvements in memory utilization and computational efficiency. We discuss both the advantages of our technique as well as the consequences or prerequisite requirements imposed by our methods in later sections.

This more efficient approach is creating opportunities to explore complex shapes through implicit modeling methods. In particular, the thin-plate-spline radial basis function and the Green's function solutions with their order O(n^2) solutions are impractical when the number of constraints exceeds a few thousand points. By shifting to a compactly supported radial basis function, we can create differentiable analytic representations of large complex models. These efficient solutions make possible techniques for studying the deep structure of solid shapes. In the discussion section of this paper, we present early applications of these implicit surfaces to problems such as surface shape analysis.

2 Background / Problem

The key idea in both [11] and [14] is that one may produce an implicit surface from known surface points by interpolating the embedding function within which the surface is implicitly defined. While we primarily follow here the presentation found in [14], we also encourage the interested reader to see an alternative and earlier formulation in [11].

2.1 Interpolating Surfaces by Interpolating Embedding Functions

An *implicit surface* is defined by $\{\overline{x} : f(\overline{x}) = 0\}$ for some embedding function $f : \mathbf{R}^n \rightarrow \mathbf{R}$. The key idea behind interpolated implicit surfaces is to find a smooth embedding function f such that $f(\overline{x}_i) = 0$ for each known surface point $\overline{x}_i$, and $f(\overline{y}_i) = 1$ for one or more points $\overline{y}_i$ known to be inside the shape. (Alternatively, constraints of the form $f(\overline{y}_i) = -1$ may be added for points outside the shape.) Turk and O'Brien select these interior points using normals at the surface points as shown in Figure 1.

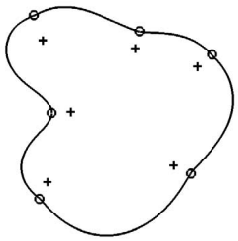


Figure 1. Implicit curve interpolated using zero-constraints along the curve and positive constraints just inside these points in the direction opposite the known (or desired) normals. (Figure courtesy of Greg Turk.)

This may be generalized to a *scattered-data interpolation* problem as follows. Given a set of positions $\overline{c}_i$ and corresponding values h_i, solve for an embedding function f such that $f(\overline{c}_i) = h_i$. Thus, surface interpolation may be turned into higher-dimensional scattered-data interpolation, a well-studied field.

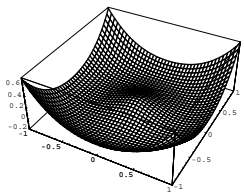
Turk and O'Brien chose to use *thin-plate splines*, which minimize the bending energy of the embedding function: $E = \int_\Omega f_{xx}^2(\overline{x}) + f_{xy}^2(\overline{x}) + f_{yy}^2(\overline{x})\ d\overline{x}$. They called their method *variational implicit surfaces*, because they formulate the problem as one of variational interpolation (minimizing an energy functional subject to interpolative constraints). They did not, however, solve for the embedding function using an iterative minimization approach but instead solve for the known closed-form solution using radial basis functions as described in Section 2.2. (Savchenko, et al. use similar splines based on the use of Green's function.)

2.2 Radial Basis Functions

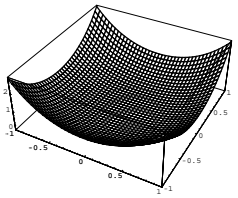
Scattered data interpolation can be achieved using *radial basis functions* centered at the constraints. Radial basis functions are circularly-symmetric functions centered at a particular point.

Duchon [5] has shown that solving for thin-plate splines through known points in two dimensions is equivalent to interpolating these points using the biharmonic radial basis function $\phi(r) = r^2 \log |r|$ (Figure 2a). In three dimensions, the thin-plate solution is equivalent to interpolating these points using the radial basis function $\phi(r) = |r|^3$ (Figure 2b).

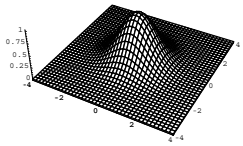
Radial basis functions may be used to interpolate a function with n points by using n radial basis functions centered at these points. The resulting interpolated function thus be-



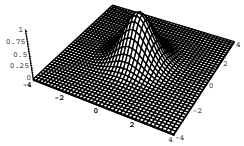
a) Thin-plate (2-d)
$\phi(r) = r^2 \log r$



b) Thin-plate (3-d)
$\phi(r) = r^3$



c) Gaussian
$\phi(r) = e^{r^2\sigma^2}$



d) Compactly Supported
$\phi(r) = (1-r)^4_+(4r+1)$

Figure 2. Comparison of different radial basis functions

comes

$$f(\overline{x}) = \sum_{i=1}^{n} d_i \phi(\|\overline{x} - \overline{c}_i\|) \tag{1}$$

where $\overline{c}_i$ is the position of the known values, d_i is the weight of the radial basis function positioned at that point. In some cases (including the thin-plate spline solution), it is necessary to add a first-degree polynomial P to account for the linear and constant portions of f and ensure positive-definiteness of the solution:

$$f(\overline{x}) = \sum_{i=1}^{n} d_i \phi(\|\overline{x} - \overline{c}_i\|) + P(\overline{x}) \tag{2}$$

To solve for the set of weights d_i that satisfy the known constraints $f(\overline{c}_i) = h_i$, we substitute each $\overline{c}_i$ into Eq. 2:

$$f(\overline{c}_i) = \sum_{j=1}^{n} d_j \phi(\|\overline{c}_i - \overline{c}_j\|) = h_i \tag{3}$$

or, if a polynomial is required:

$$f(\overline{c}_i) = \sum_{j=1}^{n} d_j \phi(\|\overline{c}_i - \overline{c}_j\|) + P(\overline{x}) = h_i \tag{4}$$

Solving for the weights d_j using Eq. 3 and denoting $\phi_{ij} = \phi(\|\overline{c}_i - \overline{c}_j\|)$ produces the following system:

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} \\ \phi_{21} & \phi_{22} & \dots & \phi_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \end{bmatrix} \tag{5}$$

If a polynomial is required, Eq. 4 similarly becomes

$$\begin{bmatrix} \phi_{11} & \phi_{12} & \dots & \phi_{1n} & c_1^x & c_1^y & c_1^z & 1 \\ \phi_{21} & \phi_{22} & \dots & \phi_{2n} & c_2^x & c_2^y & c_2^z & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_{n1} & \phi_{n2} & \dots & \phi_{nn} & c_n^x & c_n^y & c_n^z & 1 \\ c_1^x & c_2^x & \dots & c_n^x & 0 & 0 & 0 & 0 \\ c_1^y & c_2^y & \dots & c_n^y & 0 & 0 & 0 & 0 \\ c_1^z & c_2^z & \dots & c_n^z & 0 & 0 & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ p^x \\ p^y \\ p^z \\ 1 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_n \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

In both Eqs. 5 and 6 the matrix is obviously real symmetric, and with proper selection of basis functions it can be made positive-definite. Thus, a solution always exists to these systems.

2.3 Algorithmic Complexity

Calculating and using implicit surfaces that interpolate may be analyzed in three parts:

1. Constructing the system of equations,
2. Solving the system of equations, and
3. Evaluating the interpolating function (as required).

2.3.1 Constructing the System of Equations

A significant portion of the computational cost involved in calculating these implicit surfaces is the cost required to construct the matrix (or submatrix) $\phi_{ij} = \phi(\|\overline{c}_i - \overline{c}_j\|)$. Recall that the thin-plate radial basis function is $\phi(r) = r^2 \log r$ (two dimensions) or $\phi(r) = r^3$ (three dimensions). This means that *the matrix is entirely non-zero except along the diagonal*, requiring the calculation of all inter-point distances within the set $\{\overline{c}_i\}$. Although the symmetry of the matrix cuts the computational cost in half, the computational complexity is still $O(n^2)$. Furthermore, storage of such a matrix requires $O(n^2)$ floating-point values—a potentially more prohibitive factor than the computational complexity.

2.3.2 Solving the System of Equations

Although Turk and O'Brien use LU factorization (an $O(n^3)$ algorithm) to solve Eq. 5, they correctly point out that it is possible to solve this system in $O(n^2)$ by iterative means. Thus, while solution of the system may appear to be the limiting step, it need only be as computationally expensive as constructing the system.

2.3.3 Evaluating the Function

For nearly all applications it is not enough to simply solve for the weights of the respective radial basis functions.

Rather, it is necessary to evaluate this embedding function at potentially many points in order to extract the isosurface, calculate normals or other derivative quantities, etc. Because the terms $\phi(\|\overline{x} - \overline{c}_i\|)$ in Eq. 1 are all non-zero for the thin-plate solution (except for one zero term when $\overline{x} \in \{\overline{c}_i\}$), all of the terms must be used in calculating any one point. Thus, each evaluation of the interpolated function is $O(n)$.

2.4 Problems with the Thin-Plate Solution

While the thin-plate spline embedding function does indeed minimize bending energy, it has the following drawbacks in computation and usefulness for user interaction:

1. $O(n^2)$ computation is required to build the system of equations.
2. $O(n^2)$ storage is required (for the nearly-full matrix) to represent the system.
3. $O(n^2)$ computation is required to solve the system of equations.
4. $O(n)$ computation is required per evaluation
5. Because every known point affects the result, a small change in even one constraint is felt throughout the entire resulting interpolated surface, an undesirable property for shape modelling.

3 Using Compactly-Supported Radial Basis Functions

Wendland [16] has recently solved for the minimum-degree polynomial solution for compact, locally-supported radial basis functions that guarantee positive-definiteness of the matrix (Figure 2d). All of the solutions have the form

$$\phi(r) = \begin{cases} (1-r)^p P(r) & \text{if } r < 1 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

For various degrees of desired continuity (C^k) and dimensionality (d) of the interpolated function, he has derived the following radial basis functions:

$d = 1$	$(1-r)_+$	C^0
	$(1-r)_+^3(3r+1)$	C^2
	$(1-r)_+^5(8r^2+5r+1)$	C^4
$d = 3$	$(1-r)_+^2$	C^0
	$(1-r)_+^4(4r+1)$	C^2
	$(1-r)_+^6(35r^2+18r+3)$	C^6
	$(1-r)_+^8(32r^{3'}+25r^2+8r+1)$	C^6
$d = 5$	$(1-r)_+^3$	C^0
	$(1-r)_+^5(5r+1)$	C^2
	$(1-r)_+^7(16r^2+7r+1)$	C^4

These functions have radius of support equal to 1. Scaling of the basis functions (i.e., $\phi(r/\alpha)$) allows any desired radius of support α.

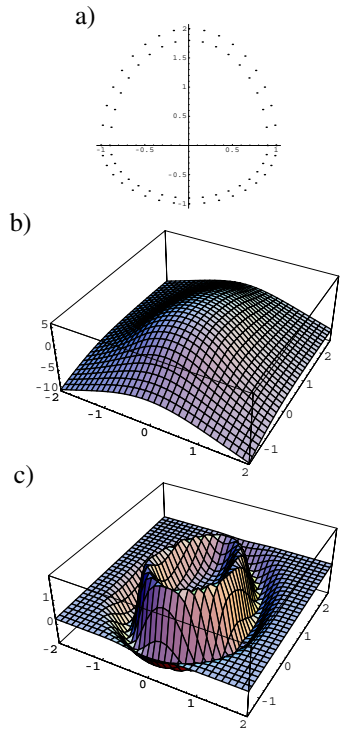


Figure 3. A simple 36-point ovoid (a) interpolated using thin-plate (b) and compactly-supported (c) radial basis functions.

3.1 Example

Figure 3 illustrates using both thin-plate and compactly-supported radial basis functions to compute embedding functions. The constraint points consist of 36 points in an ovoid shape with 36 normal (positive valued) constraints placed just inside (3a). A thin-plate radial basis function produces a globally-smooth embedding function (3b). A compactly-supported radial basis function produces an embedding function that does not have global smoothness but is as smooth as the thin-plate spline interpolation in a narrow band surrounding the shape both inside and out.

3.2 Algorithm

Using compactly-supported radial basis functions provides advantages in all three phases of the implicit-surface interpolation algorithm.

3.2.1 Constructing the system

Because the radial basis functions have finite support, $\phi(\|\overline{c}_i - \overline{c}_j\|) = 0$ for all $(\overline{c}_i, \overline{c}_j)$ farther apart than the radius of support. By using a k-d tree [1], the set of all points

within distance r of a particular point $\overline{c}_i$ can be determined in $O(\log n)$ time.

A k-d tree is a multidimensional binary tree with the following sorting property for a tree with point $\overline{x}$ at the root and subtrees T_{left} and T_{right}.

$$\forall \overline{y} \in T_{\text{left}} : \overline{y}^d \leq \overline{x}^d$$

$$\forall \overline{y} \in T_{\text{right}} : \overline{y}^d > \overline{x}^d$$

where the sorting dimension d changes at each level of the tree.

k-d trees can be used to find all points within distance r of a particular constraint in $O(n \log n)$ time using the following algorithm (C pseudocode):

```
void Search (KDtree T, Point P, int radius)
{
  if (T->point[T->dim]) < P[dim] + radius)
    Search(T->right);
  if (T->point[T->dim]) > P[dim] - radius)
    Search(T->left);
  Test(T->point,P,radius);
}
```

While a number of points must still be tested explicitly, the multidimensional sorting nature of the k-d tree allows a large number of points to be rejected at each level of the tree.

The resulting matrix is extremely sparse, as shown in Figure 4. Using a sparse-matrix representation (we use the Hartwell-Boeing format), only $O(n)$ storage is required.

3.2.2 Solving the system

If the average number of points within the radius of support of each constraint $\overline{c}_i$ is less than some constant k, the number of non-zero entries in the matrix is $O(n)$. We use a direct (LU) sparse matrix solver [4] to find the solution to the system of equations. The computational complexity of such a solver depends on the amount of matrix "fill in" that occurs during the solution, but some authors have reported behavior in the range $O(n^{1.2})$ to $O(n^{1.5})$ [10]. Our own experience (Section 4) agrees with this.

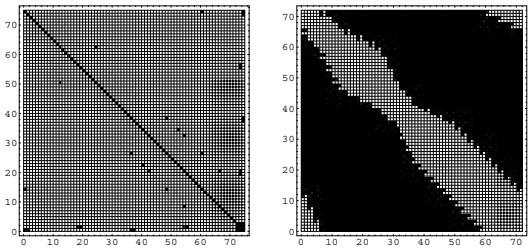


Figure 4. Structure of the matrices produced by thin-plate and compactly-supported radial basis functions (Figure 3). The compactly-supported basis function produce a matrix that is sparse (black), while the thin-plate basis functions produce a matrix that is nearly full (white).

3.2.3 Evaluating the interpolating function

We can exploit the spatial locality of the compactly-supported radial basis functions during evaluation of the embedding function f by recognizing that only a fraction of the terms of Eq. 1 are non-zero for a given $\overline{x}$: $\phi(\overline{c}_i - \overline{c}_j) \neq 0$ iff $\|\overline{c}_i - \overline{c}_j\| < 1$. By again using a k-d tree to organize the constraints spatially, each evaluation of the interpolating function requires only $O(\log n)$ operations to determine these non-zero terms.

3.3 Thin-Plate vs. Compactly-Supported Radial Basis Functions

Using compactly-supported radial basis functions directly addresses each of the five problems identified previously with the thin-plate spline basis functions (Section 2.4) as follows:

	Thin-plate	Compact
Computation to build	$O(n^2)$	$O(n \log n)$
Computation to solve	$O(n^2)$	$O(n^{1.5})$
Storage to build/solve	$O(n^2)$	$O(n)$
Computation to evaluate	$O(n)$	$O(\log n)$
Effect of a single point	Global	Local

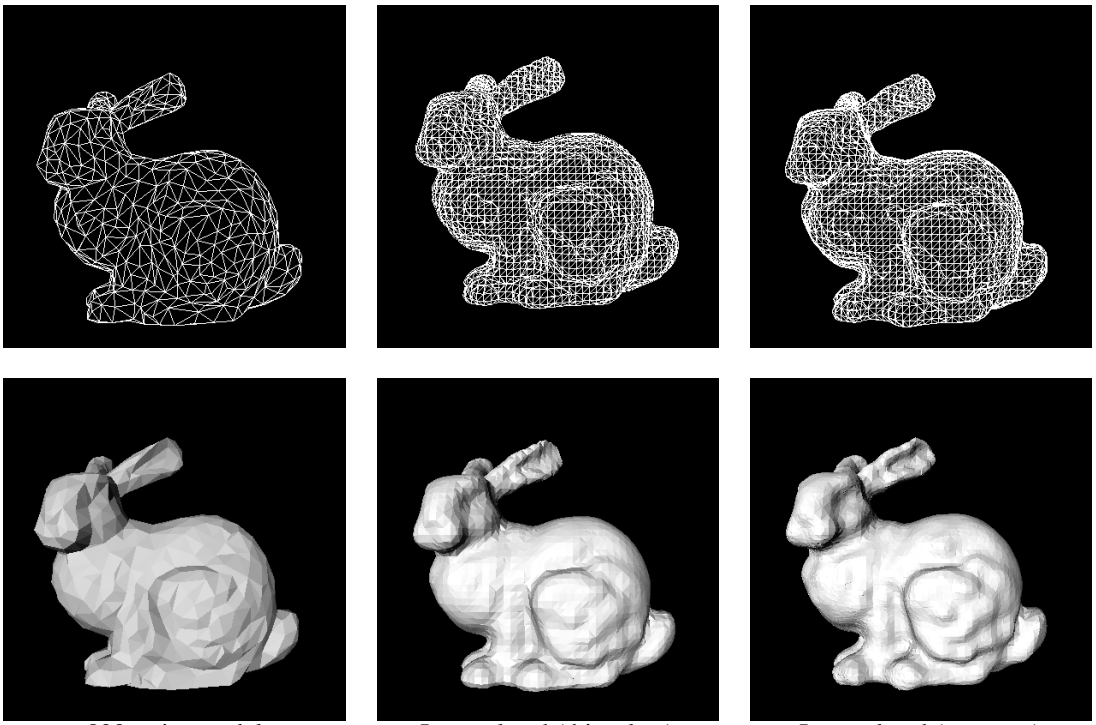
4 Results

Figure 5 compares the results of interpolating an 800-point model of the Stanford bunny using both thin-plate and compactly-supported radial basis functions. The results of of the two methods are qualitatively identical.

Interpolation of larger models is possible using compactly-supported radial basis functions. The 8000-point model in Figure 6 was interpolated using compactly-supported radial basis functions, and the resulting isosurface was tesselated to 41,864 points.[1] The interpolated image shows sharpness and detail hidden by the flat polyhedra of the original model.

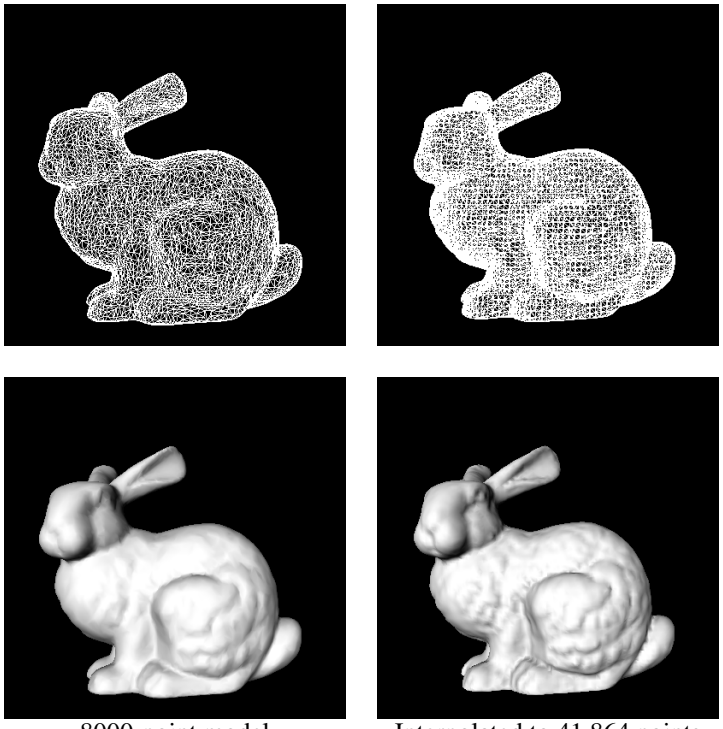
Table 1 summarizes the time required to calculate various differing-resolution models of the same figure (Stanford bunny). The radius of support used for each model was selected so as to keep the number of points in the radius of

[1] A similar thin-plate interpolation would require almost 2 GB of storage for the matrix alone and could not be calculated on our system.



800-point model | Interpolated (thin-plate) | Interpolated (compact)

Figure 5. Comparison of original model to implicit surfaces extracted from embedding functions calculated using both thin-plate and compactly-supported basis functions. The results of the interpolations are qualitatively identical.



8000-point model | Interpolated to 41,864 points

Figure 6. Interpolation of an 8000-point model of the Stanford bunny

Points	Constraints	Radius	Non-zero	Per Row	% Full	Build k-d	Build Matrix	Solve Matrix	Total
2922	5848	0.200	476173	81.4	1.39%	0.02	1.05	5.93	7.00
5839	11682	0.150	973423	83.3	0.71%	0.04	2.09	17.29	19.42
11831	23666	0.100	1663733	70.3	0.30%	0.08	4.00	45.44	49.52
35947	71898	0.004	5061542	70.4	0.10%	0.93	31.51*	284.01*	316.45*

* some virtual memory swapping

Table 1. Execution times in seconds for various phases of computation for interpolated implicit surfaces using compactly-supported radial basis functions.

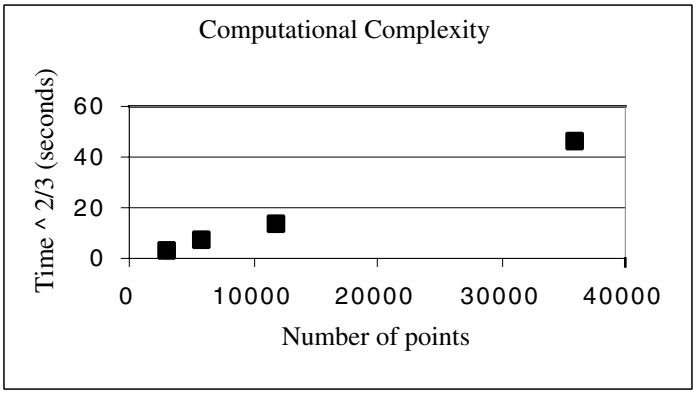


Figure 7. Timing results for four different-resolution models of the same figure (Table 1). The linear graph when plotted on a $time^{\frac{2}{3}}$ vertical axis indicates $O(n^{1.5})$ complexity.

Thin-Plate			Compactly-Supported		
Points	Build	Solve	Points	Build	Solve
800	0.58 sec	2.56 min	800	0.57 sec	1.53 sec
2000	6.3 sec	2:05 hrs	2922	1.07 sec	5.93 sec
4000	25.0 sec	16:11 hrs*	5839	2.13 sec	17.29 sec
8000	n/a (1)	n/a	11831	4.08 sec	97.53 sec
35947	n/a (2)	n/a	35947	31.44 sec	284.01 sec

* significant virtual memory swapping

Table 2. Comparison of execution times required to calculate embedding functions using thin-plate vs. compactly-supported radial basis functions

Thin-Plate		Compactly-Supported	
Points	Memory (MB)	Points	Memory (MB)
800	19.5	800	1.0
2000	122.1	2922	3.6
4000	488.3	5839	7.4
8000	1,953.1	11831	12.7
35947	39,434.4	35947	38.6

Table 3. Comparison of memory requirements (matrix only, double-precision floating-point values) required to calculate embedding functions using thin-plate vs. compactly-supported radial basis functions

support (approximately) constant, thus allowing comparison of the results. The dominant term in the required computation is the time to solve the system of equations, which seems to demonstrate $O(n^{1.5})$ complexity (Figure 7).

Table 2 compares these running times to the time required to calculate comparable thin-plate interpolations (using same-size or smaller models). The $O(n^2)$ time required to compute these models using thin-plate radial basis functions quickly becomes prohibitive.

Similarly, Table 3 compares the memory required to represent the matrix for the models described in Table 2. As with the computational complexity, the $O(n^2)$ storage required to compute these models using thin-plate radial basis functions also quickly becomes prohibitive.

5 Considerations

The finite nature of the compactly-supported radial basis functions introduces two factors that must be considered when using them to interpolate embedding functions.

5.1 Selecting the Radius of Support

The finite radius of support introduces an additional parameter that doesn't exist in the thin-plate implementation. Proper selection of the radius of support is critical to achieving optimal efficiency of computation and results. Too small a radius can produce basis functions that are unable to span the inter-constraint gaps. Too large a radius does not adversely affect the results but reduces the sparseness of the matrix, thus increasing the computation required. It is thus necessary to select a radius of support that is both large enough to produce effective results and not so large that the computation becomes impractical.

5.2 Isosurface Extraction

Because of the finite extent of the compactly-supported radial basis functions, only those points within the radius of support of one of the original positions have non-zero values. For all points outside this band, all of the terms of Eq. 1 are zero. (Figure 3c illustrates this.) In this way, these embedding functions are not the same as those normally used for implicit surfaces—the implicit surface represented is not the only set of zero-valued points in the space. However, the implicit surface does form a unique contiguous locus of zero-valued points passing through the constraints. In this sense, the method presented here is somewhat similar to the narrow-band active set approach of Sethian [12] or Whitaker and Breen [17].

An isosurface extractor may be used to extract this surface by seeding it with any one of the initial constraints. However, care must be taken so that the step size of the extractor does not cause it to jump outside the band of non-zero points. It is, however, rather easy to explicitly recognize when no non-zero terms are found in Eq. 1 (none of the constraint points lie within the radius of support of the point being evaluated).

Because the surface of interest is not the only zero set of the embedding function, the resulting embedding function has limited application in CSG, interpolation [15], or similar applications. However, we are experimenting with a hybrid approach that interpolates a subset of the points using radial basis functions with infinite support and the difference using basis functions with compact support.

6 Discussion: Analytic Approaches Enabled by Efficient Algorithms

An efficient framework for finding embedding functions that define implicit surfaces from scattered data points increases the practicality of studying complex shape models represented by large numbers of such points. Models captured from physical phenomena usually contain large numbers of surface points, polygons, or other surface primitives that are not easily reduced. For instance, polygonal representations of medical data often begin with models containing millions of triangles, which can later be simplified to hundreds of thousands of polygons.

While the polygonal representations of these models can be rendered using current graphics hardware, the discrete sampling introduced by the process of producing the polyons raises barriers to deeper studies of the geometry of the surfaces themselves. Implicit methods solve this difficulty by creating analytic functions that smoothly reconstruct a surface from a constellation of points. In addition, the implicit surface constructed using the method presented here is differentiable. Local surface geometry now becomes approachable since numerically stable solutions can be found to sample higher order derivatives of the implicit surface.

6.1 Differential Geometry of Shape

Differential geometry is the study of multilocal surface behavior, employing the normal vector and tangent plane at each surface point as a reference environment. Curvature and other geometric features relative to the surface tangent are measured using second order derivatives [13]. Volumetric approaches can compute approximations to these measurements based on discretely sampled grids [12]. However, the accuracy and behavior of the sampled derivatives is subject to aliasing and sampling issues, exacerbated by the noise-amplifying effects of higher-order functions.

Implicit surfaces allow us to reconstruct smooth surface representations from a set of oriented points and sample

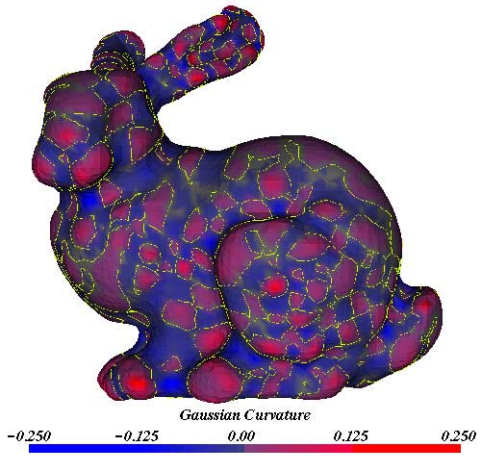


Figure 8. Gaussian curvature computed over an analytically-defined implicit surface calculated from scattered surface points

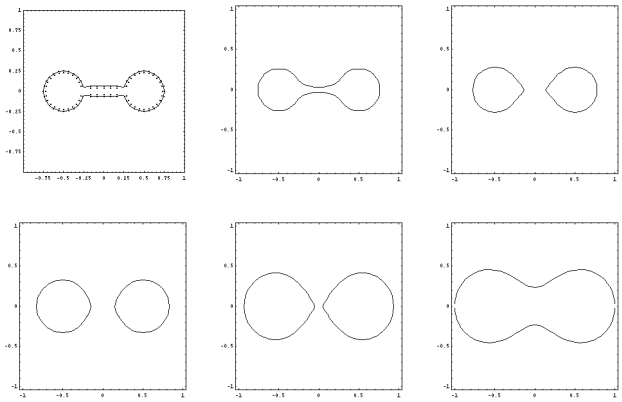


Figure 9. A dumbell figure reconstructed from sample points and represented at successively larger scales by convolving the embedding function with successively larger Gaussian kernels

higher order derivatives with instantaneous precision. Analytic representations of not only surface shape, but also of the embedding function, simplifies the computation of the normal vector, the tangent plane, and the principal curvatures of the implicit surface at any arbitrary location.

Figure 8 shows the Gaussian curvature (the product of the principal curvatures) of an array of implicit surfaces calculated using the method presented here. Blue areas represent positive Gaussian curvature (elliptical regions) and red areas represent negative Gaussian curvature (hyperbolic or saddle-shaped regions). The yellow contour lines indicate the places where the Gaussian curvature is zero, separating the red and blue surface regions. The yellow contours denote parabolic curves where specular and diffuse highlights fuse or reproduce under changing lighting conditions.

Note: the polygonalization of the surface and the interpolation errors are artifacts of the visualization technique. The implicit surface itself can be sampled with arbitrary precision to generate smooth models with correspondingly smooth representations of curvature.

6.2 Future Directions: Scale Space, Ray Tracing, and Other Topics

Linear scale space filtering was introduced by Witkin [19] as a means of measuring the saliency of features within an image. Further work established the general field of scale-space theory in computer vision [8]. The fundamental notion of such analysis is that significant details of an image (or surface representation) persist as the scale or the measurement aperture is increased. In the pursuit of multiscale image descriptions, a Gaussian kernel is usually used as the measurement aperture function.

From this perspective, a scale space representation of the implicit models presented in this paper can be constructed as a convolution of a Gaussian kernel with the embedding function, $f(\bar{x}) \otimes g(\bar{x}, \sigma)$. Since convolution is both commutative and distributive with respect to addition, this is equivalent to convolving each of the radial basis interpolants with a Gaussian. This process can be approximated by solving for the implicit surface with a particular radius of support, and later dilating the compactly supported radial basis function during the evaluation to create a scale space level set representation of the basic function.

Figure 9 shows a 2-D implicit figure represented at a range of scales. The original model has been reconstructed from oriented points as an implicit surface. In the subsequent representations, the embedding function has been convolved with an approximation to a Gaussian kernel, and the implicit surface reconstructed. Fine details such as the corners and discontinuities associated with the cross member in the figure are suppressed at moderate scales. Eventually at the largest scales, the figure is viewed as a single topologically simple object. This approach to representing object shape may have applications in modeling and computer graphics in the representation of objects at multiple levels of detail.

Beyond the application of scale-space image-analysis techniques to implicit surface representations, there remain interesting problems of rendering these models. Implicit surfaces based on signed distance functions and other embedding functions with similar properties are easily rendered through ray tracing. Because of the multiple zero level sets created by the compactly supported radial basis function approach, a basic ray tracing method is insufficient for rendering these models. However, in the visual-

ization of discrete volume data, complex transfer functions that include geometric information such as gradient magnitude and isosurface curvature are able to capture surfaces based on features other than simple isovalues. Future work will include the development of transfer functions for raycast rendering of these models.

7 Conclusion

Given a set of points $C = \{\overline{c}_1, \overline{c}_2, \ldots \overline{c}_n\}$ and associated normals, the method presented here interpolates an implicit surface through those points by finding a scalar embedding function $f(\overline{x})$ whose zero level set, $\{\overline{x} : f(\overline{x}) = 0\}$, passes through all points in C. Following Turk and O'Brien [14] (see also [11]), we employ radial basis interpolating functions in a linear system of $2n$ equations and $2n$ unknowns, expressed as an $2n \times 2n$ matrix. Unlike the original use of $\phi(r) = |r|^3$ as the underlying interpolant, we use a family of radial basis functions with the necessary continuity but also with compact local support. The result is a sparse system whose solution can be accelerated. The result is a single, accelerated, closed form analytic representation of the desired surface.

The shift from a radial basis function of infinite extent to a compactly supported one creates dramatic gains in memory utilization and computational complexity. Previous work described solutions for systems of equations of order $O(n^2)$ complexity. The shift to finite interpolants and sparse matrices has shifted the complexity of the matrix solution to order $O(n^{1.5})$, the loading of the matrix data structures to $O(n \log n)$, and the memory requirements to order $O(n)$. Evaluation of the interpolated embedding function is similarly reduced to $O(\log n)$.

These improvements in efficiency make possible a variety of applications that were previously impractical with an infinite radial basis function. We have briefly surveyed our first probes into the differential geometry of surface shape and explorations in scale space analysis of complex models using implicit surfaces interpolated from scattered surface data points.

Acknowledgments

This work was performed in large part at the National Library of Medicine under a visiting faculty program supporting both Dr. Morse and Dr. Subramanian. Dr. Rheingans was supported in part by NSF CAREER Grant #9996043. We would like to thank Greg Turk for his useful conversations and for making his code available to us, upon which our implementation is based. We would also like to thank Dr. Michael Ackerman and the staff of NLM's Office of High Performance Computing and Communications for their help and support. Finally, we would like to thank the reviewers and program committee for SMI2001 for their many helpful suggestions.

References

[1] J. L. Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9):509–517, 1975.

[2] J. Blinn. A generalization of algebraic surface drawing. *IEEE Transactions on Graphics*, 1(3):235–246, 1982.

[3] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan-Kaufman, 1997.

[4] J. J. Dongarra, J. D. Croz, S. Hammarling, and I. Duff. A set of level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, Mar. 1990.

[5] J. Duchon. Sur l'erruer d'interpolation des fonctions de plusieurs variables par les d^m splines. *R.A.I.R.O Analyse numerique*, 12(4):325–334, 1978.

[6] J. Hart and e. D. Ebert. *New Fontiers in Modeling and Texturing*. Siggraph 97 Course Notes, 1997.

[7] B. Kimia, A. Tannenbaum, and S. Zucker. On optimal control methods in computer vision and image processing. In B. t.H. Romeny, editor, *Geometry Driven Diffusion in Computer Vision*, pages 307–338. Kluwer, 1994.

[8] T. Lindeberg. *Scale-space theory in computer vision*. Kluwer Academic Publishers, 1994.

[9] S. Osher and J. A. Sethian. Fronts propogating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation. *J. Comput. Phys.*, 79:12–49, 1988.

[10] R. A. Saleh, K. A. Gallivan, M. Chang, I. N. Hajj, D. Smart, and T. N. Patrick. Parallel circuit simulation on supercomputers. *Proceedings of the IEEE*, 77(12):1915–1930, 1989.

[11] V. V. Savchenko, A. A. Pasko, O. G. Okunev, and T. L. Kunii. Function representation of solids reconstructed from scattered surface points and contours. *Computer Graphics Forum*, 14(4):181–188, 1995.

[12] J. A. Sethian. *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Sciences*. Cambridge University Press, 1996.

[13] J. Thirion. New feature points based on geometric invariants for 3d image registration. Technical Report INRIA-RR-1901, INRIA, 1993.

[14] G. Turk and J. F. O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, 1998.

[15] G. Turk and J. F. O'Brien. Shape transformation using variational implicit surfaces. In *Computer Graphics Proceedings, Annual Conference Series*, 1999.

[16] H. Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *AICM*, 4:389–396, 1995.

[17] R. Whitaker and D. Breen. Level-set models for the deformation of solid objects. In *The Third International Workshop on Implicit Surfaces*, pages 19–35. Eurographics, 1998.

[18] A. Witkin and P. Heckbert. Using particles to sample and control implicit surfaces. In A. Glassner, editor, *SIGGRAPH '94 Proceedings*, Computer Graphics Proceedings, Annual Conference Series, pages 269–278. ACM SIGGRAPH, ACM Press, July 1994.

[19] A. P. Witkin and J. M. Tenenbaum. On the role of structure in vision. In J. Beck, B. Hope, and A. Rosenfeld, editors, *Human and Machine Vision*, pages 481–543. Academic Press, New York, NY, 1983.