Reconstruction and Representation of 3D Objects with Radial Basis Functions

J. C. Carr[1,2] R. K. Beatson[2] J. B. Cherrie[1] T. J. Mitchell[1,2] W. R. Fright[1] B. C. McCallum[1]
T. R. Evans[1]

[1] Applied Research Associates NZ Ltd*
[2] University of Canterbury†

(a) (b)

Figure 1: (a) Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud. (b) Automatic mesh repair using the biharmonic RBF.

Abstract

We use polyharmonic Radial Basis Functions (RBFs) to reconstruct smooth, manifold surfaces from point-cloud data and to repair incomplete meshes. An object's surface is defined implicitly as the zero set of an RBF fitted to the given surface data. Fast methods for fitting and evaluating RBFs allow us to model large data sets, consisting of millions of surface points, by a single RBF — previously an impossible task. A greedy algorithm in the fitting process reduces the number of RBF centers required to represent a surface and results in significant compression and further computational advantages. The energy-minimisation characterisation of polyharmonic splines result in a "smoothest" interpolant. This scale-independent characterisation is well-suited to reconstructing surfaces from non-uniformly sampled data. Holes are smoothly filled and surfaces smoothly extrapolated. We use a non-interpolating approximation when the data is noisy. The functional representation is in effect a solid model, which means that gradients and surface normals can be determined analytically. This helps generate uniform meshes and we show that the RBF representation has advantages for mesh simplification and remeshing applications. Results are presented for real-world rangefinder data.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations;

Keywords: Variational implicit surfaces, Radial Basis Function, RBF, mesh repair, point-cloud surfacing, surface reconstruction, geometry compression, solid modeling.

*Applied Research Associates NZ Ltd, PO Box 3894, Christchurch, New Zealand. Email: [j.carr,j.cherrie,r.fright,b.mccallum]@aranz.com Web: www.aranz.com

†Dept. Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand, Email: r.beatson@math.canterbury.ac.nz

1 Introduction

Interpolating incomplete meshes (hole-filling) and reconstructing surfaces from point-clouds derived from 3D range scanners are ubiquitous problems in computer graphics and Computer Aided Design (CAD). Smoothly blending between surfaces and ensuring surfaces are manifold, and therefore manufacturable, are related problems in CAD. Similarly, smoothing and remeshing existing noisy surfaces are important problems in both CAD and computer graphics. These problems have mostly been considered independent from one another and received much attention in the literature (see [8] and the references therein). In this paper we propose that the implicit representation of object surfaces with Radial Basis Functions (RBFs) simplifies many of these problems and offers a unified framework that is simple and elegant. An RBF offers a compact functional description of a set of surface data. Interpolation and extrapolation are inherent in the functional representation. The RBF associated with a surface can be evaluated anywhere to produce a mesh at the desired resolution. Gradients and higher derivatives are determined analytically and are continuous and smooth, depending on the choice of basic function. Surface normals are therefore reliably calculated and iso-surfaces extracted from the implicit RBF model are manifold (*i.e.*, they do not self-intersect).

ACM SIGGRAPH 2001, 12-17 August 2001, Los Angeles, CA, USA

The benefits of modeling surfaces with RBFs have been recognised by Savchenko [19], Carr *et al.* [10] and by Turk & O'Brien [23, 22]. However, this work was restricted to small problems by the $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^3)$ arithmetic operations of direct methods. For example, direct fitting of the dragon in Fig. 1(a) would have required 3,000GB just to store the corresponding matrix. Consequently, fitting RBFs to real-world scan data has not been regarded as computationally feasible for large data sets. The fast fitting and evaluation methods introduced in this paper mean modeling surface data from large data sets and complicated objects is now feasible. In this paper we:

- describe the computational advantages offered by new fast methods for fitting and evaluating RBFs.
- introduce RBF center reduction for implicit surface modeling. This results in significant data compression and speed improvements.
- demonstrate that modeling very large data sets and topologically complicated objects is possible.
- introduce RBF approximation for the problem of reconstructing smooth surfaces from noisy data.
- apply RBFs to the problem of mesh repair (hole-filling) and point-cloud surface fitting.

1.1 Implicit surfaces

The surface representation or reconstruction problem can be expressed as

Problem 1.1. *Given n distinct points $\{(x_i, y_i, z_i)\}_{i=1}^n$ on a surface M in $\mathbb{R}^3$, find a surface M' that is a reasonable approximation to M.*

Our approach is to model the surface implicitly with a function $f(x, y, z)$. If a surface M consists of all the points (x, y, z) that satisfy the equation

$$f(x, y, z) = 0, \tag{1}$$

then we say that f implicitly defines M. Describing surfaces implicitly with various functions is a well-known technique [9].

In Constructive Solid Geometry (CSG) an implicit model is formed from simple primitive functions through a combination of Boolean operations (union, intersection etc) and blending functions. CSG techniques are more suited to the design of objects in CAD rather than their reconstruction from sampled data. Piecewise low-order algebraic surfaces, sometimes referred to as implicit patches or semi-algebraic sets, have also been used to define surfaces implicitly. These are analogous to piecewise parametric splines except that the surface is implicitly defined by low degree polynomials over a piecewise tetrahedral domain. An introduction to these techniques and further references can be found in [9].

The distinction between our approach and these well-known techniques is that we wish to model the entire object with a single function which is continuous and differentiable. A single functional description has a number of advantages over piecewise parametric surfaces and implicit patches. It can be evaluated anywhere to produce a particular mesh, *i.e.,* a faceted surface representation can be computed at the desired resolution when required. Sparse, non-uniformly sampled surfaces can be described in a straightforward manner and the surface parameterization problem, associated with piecewise fitting of cubic spline patches, is avoided.

The representation of objects with single functions has previously been restricted to modeling "blobby" objects such as molecules [9] and has generally been believed to be infeasible for real-world objects acquired by 3D scanners. Turk & O'Brien [23] have tried modeling laser scan data with RBFs. However they have been restricted to blobby approximations derived from small subsets of data consisting of a few hundred to a thousand surface points. More recent work [25] has focussed on simplifying larger data sets to make them computationally manageable. However, the ability to represent complicated objects of arbitrary topology and model the detail obtainable with modern laser scanners, is compromised. Carr *et al.* [10] used RBFs to reconstruct cranial bone surfaces from 3D CT scans. Data surrounding large irregular holes in the skull were interpolated using the thin-plate spline RBF. Titanium plate was then molded into the shape of the fitted surface to form a cranial prosthesis. That paper exploited the interpolation and extrapolation characteristics of RBFs as well as the underlying physical properties of the thin-plate spline basic function. However, the approach is restricted to modeling surfaces which can be expressed explicitly as a function of two variables. In this paper we demonstrate that by using new fast methods, RBFs can be fitted to 3D data sets consisting of millions of points without restrictions on surface topology — the kinds of data sets typical of industrial applications.

Our method involves three steps:

- Constructing a signed-distance function.
- Fitting an RBF to the resulting distance function.
- Iso-surfacing the fitted RBF.

Section 2 of this paper describes how we formulate the surface fitting problem as a scattered data interpolation problem. Section 3 introduces RBFs and Section 4 describes the use of new fast methods which overcome problems that have prevented the use of RBFs in the past. Section 5 introduces RBF center reduction which results in a compact surface representation as well as faster fitting and evaluation times. Section 6 introduces RBF approximation as a means of smoothing noisy surface data and demonstrates this by reconstructing smooth surfaces from noisy LIDAR (laser rangefinder) data. Section 7 describes iso-surface extraction. Section 8 demonstrates the abilities of RBFs to reconstruct non-trivial surfaces from point-clouds and interpolate incomplete meshes. Section 9 draws conclusions and discusses future work.

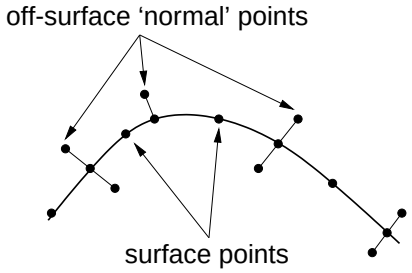


Figure 2: A signed-distance function is constructed from the surface data by specifying off-surface points along surface normals. These points may be specified on either or both sides of the surface, or not at all.

2 Fitting an implicit function to a surface

We wish to find a function f which implicitly defines a surface M' and satisfies the equation

$$f(x_i, y_i, z_i) = 0, \qquad i = 1, \ldots, n,$$

(a) (b) (c)

Figure 3: Reconstruction of a hand from a cloud of points with and without validation of normal lengths.

where $\{(x_i, y_i, z_i)\}_{i=1}^{n}$ are points lying on the surface. In order to avoid the trivial solution that f is zero everywhere, off-surface points are appended to the input data and are given non-zero values. This gives a more useful interpolation problem: Find f such that

$$f(x_i, y_i, z_i) = 0, \qquad i = 1, \ldots, n \qquad (\textit{on}\text{-surface points}),$$
$$f(x_i, y_i, z_i) = d_i \neq 0, \quad i = n+1, \ldots, N \qquad (\textit{off}\text{-surface points}).$$

This still leaves the problem of generating the off-surface points $\{(x_i, y_i, z_i)\}_{i=n+1}^{N}$ and the corresponding values d_i.

An obvious choice for f is a *signed-distance function*, where the d_i are chosen to be the distance to the closest on-surface point. Points outside the object are assigned positive values, while points inside are assigned negative values. Similar to Turk & O'Brien [23], these off-surface points are generated by projecting along surface normals. Off-surface points may be assigned either side of the surface as illustrated in Fig. 2.

Experience has shown that it is better to augment a data point with two off-surface points, one either side of the surface. In Fig. 3(a) surface points from a laser scan of a hand are shown in green. Off-surface points are color coded according to their distance from their associated on-surface point. Hot colors (red) represent positive points outside the surface while cold colors (blue) lie inside. There are two problems to solve; estimating surface normals and determining the appropriate projection distance.

If we have a partial mesh, then it is straightforward to define off-surface points since normals are implied by the mesh connectivity at each vertex. In the case of unorganised point-cloud data, normals may be estimated from a local neighbourhood of points. This requires estimating both the normal direction and determining the sense of the normal. We locally approximate the point-cloud data with a plane to estimate the normal direction and use consistency and/or additional information such as scanner position to resolve the sense of the normal. In general, it is difficult to robustly estimate normals everywhere. However, unlike other methods [16] which also rely on forming a signed-distance function, it is not critical to estimate normals everywhere. If normal direction or sense is ambiguous at a particular point then we do not fit to a normal at that point. Instead, we let the fact that the data point is a zero-point (lies on the surface) tie down the function in that region.

Given a set of surface normals, care must be taken when projecting off-surface points along the normals to ensure that they do not intersect other parts of the surface. The projected point is constructed so that the closest surface point is the surface point that generated it. Provided this constraint is satisfied, the reconstructed surface is relatively insensitive to the projection distance $|d_i|$. Fig. 3(c) illustrates the effect of projecting off-surface points inappropriate distances along normals. Off-surface points have been chosen to lie a fixed distance from the surface. The resulting surface, where f is zero, is distorted in the vicinity of the fingers where opposing normal vectors have intersected and generated off-surface points with incorrect distance-to-surface values, both in sign and magnitude. In Fig. 3(a) & (b) validation of off-surface distances and dynamic projection has ensured that off-surface points produce a distance field consistent with the surface data. Fig. 4 is a cross section through the fingers of the hand. The figure illustrates how the RBF function approximates a distance function near the object's surface. The approximately equally spaced iso-contours at $+1$, 0 and -1 in the top of the figure and the corresponding function profile below, illustrate how the off-surface points have generated a function with a gradient magnitude close to 1 near the surface (which corresponds to the zero-crossings in the profile shown).

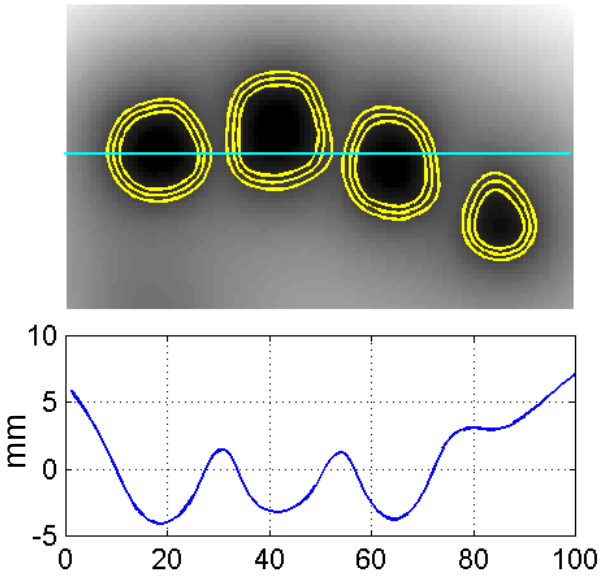


Figure 4: Cross section through the fingers of a hand reconstructed from the point-cloud in Fig. 3. The iso-contours corresponding to +1, 0 and -1 are shown (top) along with a cross sectional profile of the RBF (bottom) along the line shown.

3 Radial Basis Function interpolation

Given a set of zero-valued surface points and non-zero off-surface points we now have a scattered data interpolation problem: we want to approximate the signed-distance function $f(\boldsymbol{x})$ by an interpolant $s(\boldsymbol{x})$. The problem can be stated formally as follows,

Problem 3.1. *Given a set of distinct nodes* $X = \{\boldsymbol{x}_i\}_{i=1}^{N} \subset \mathbb{R}^3$ *and a set of function values* $\{f_i\}_{i=1}^{N} \subset \mathbb{R}$, *find an* interpolant

$s : \mathbb{R}^3 \to \mathbb{R}$ *such that*

$$s(\boldsymbol{x}_i) = f_i, \qquad i = 1, \ldots, N. \tag{2}$$

Note that we use the notation $\boldsymbol{x} = (x, y, z)$ for points $\boldsymbol{x} \in \mathbb{R}^3$.

The interpolant will be chosen from $\mathrm{BL}^{(2)}(\mathbb{R}^3)$, the Beppo-Levi space of distributions on $\mathbb{R}^3$ with square integrable second derivatives. This space is sufficiently large to have many solutions to Problem 3.1, and therefore we can define the affine space of interpolants:

$$S = \{s \in \mathrm{BL}^{(2)}(\mathbb{R}^3) : s(\boldsymbol{x}_i) = f_i, \quad i = 1, \ldots, N\}. \tag{3}$$

The space $\mathrm{BL}^{(2)}(\mathbb{R}^3)$ is equipped with the rotation invariant semi-norm defined by

$$\|s\|^2 = \int_{\mathbb{R}^3} \left(\frac{\partial^2 s(\boldsymbol{x})}{\partial x^2}\right)^2 + \left(\frac{\partial^2 s(\boldsymbol{x})}{\partial y^2}\right)^2 + \left(\frac{\partial^2 s(\boldsymbol{x})}{\partial z^2}\right)^2 + 2\left(\frac{\partial^2 s(\boldsymbol{x})}{\partial x \partial y}\right)^2 + 2\left(\frac{\partial^2 s(\boldsymbol{x})}{\partial x \partial z}\right)^2 + 2\left(\frac{\partial^2 s(\boldsymbol{x})}{\partial y \partial z}\right)^2 d\boldsymbol{x}. \tag{4}$$

This semi-norm is a measure of the energy or "smoothness" of functions: functions with a small semi-norm are smoother than those with a large semi-norm. Duchon [12] showed that the smoothest interpolant, *i.e.,*

$$s^\star = \arg\min_{s \in S} \|s\|,$$

has the simple form

$$s^\star(\boldsymbol{x}) = p(\boldsymbol{x}) + \sum_{i=1}^{N} \lambda_i |\boldsymbol{x} - \boldsymbol{x}_i|, \tag{5}$$

where p is a linear polynomial, the coefficients λ_i are real numbers and $|\cdot|$ is the Euclidean norm on $\mathbb{R}^3$.

This function is a particular example of a *radial basis function* (RBF). In general, an RBF is a function of the form

$$s(\boldsymbol{x}) = p(\boldsymbol{x}) + \sum_{i=1}^{N} \lambda_i \phi(|\boldsymbol{x} - \boldsymbol{x}_i|), \tag{6}$$

where p is a polynomial of low degree and the *basic function* ϕ is a real valued function on $[0, \infty)$, usually unbounded and of non-compact support (see, *e.g.,* Cheney & Light [11]). In this context the points $\boldsymbol{x}_i$ are referred to as the *centers* of the RBF.

Popular choices for the basic function ϕ include the thin-plate spline $\phi(r) = r^2 \log(r)$ (for fitting smooth functions of two variables), the Gaussian $\phi(r) = \exp(-cr^2)$ (mainly for neural networks), and the multiquadric $\phi(r) = \sqrt{r^2 + c^2}$ (for various applications, in particular fitting to topographical data). For fitting functions of three variables, good choices include the biharmonic ($\phi(r) = r$, *i.e.,* Equation (5)) and triharmonic ($\phi(r) = r^3$) splines.

RBFs are popular for interpolating scattered data as the associated system of linear equations is guaranteed to be invertible under very mild conditions on the locations of the data points [11, 18]. For example, the biharmonic spline of Equation (5) only requires that the data points are not co-planar, while the Gaussian and multiquadric place no restrictions on the locations of the points. In particular, RBFs do not require that the data lie on any sort of regular grid.

An arbitrary choice of coefficients λ_i in Equation (5) will yield a function $s^\star$ that is not a member of $\mathrm{BL}^{(2)}(\mathbb{R}^3)$. The requirement that $s^\star \in \mathrm{BL}^{(2)}(\mathbb{R}^3)$ implies the orthogonality or side conditions

$$\sum_{i=1}^{N} \lambda_i = \sum_{i=1}^{N} \lambda_i x_i = \sum_{i=1}^{N} \lambda_i y_i = \sum_{i=1}^{N} \lambda_i z_i = 0.$$

More generally, if the polynomial in Equation (6) is of degree m then the side conditions imposed on the coefficients are

$$\sum_{i=1}^{N} \lambda_i q(\boldsymbol{x}_i) = 0, \text{ for all polynomials } q \text{ of degree at most } m. \tag{7}$$

These side conditions along with the interpolation conditions of Equation (2) lead to a linear system to solve for the coefficients that specify the RBF.

Let $\{p_1, \ldots, p_\ell\}$ be a basis for polynomials of degree at most m and let $\boldsymbol{c} = (c_1, \ldots, c_\ell)$ be the coefficients that give p in terms of this basis. Then Equations (2) and (7) may be written in matrix form as

$$\begin{pmatrix} A & P \\ P^\mathsf{T} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} = B \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}, \tag{8}$$

where

$$A_{i,j} = \phi(|\boldsymbol{x}_i - \boldsymbol{x}_j|), \qquad i, j = 1, \ldots, N,$$
$$P_{i,j} = p_j(\boldsymbol{x}_i), \qquad i = 1, \ldots, N, \quad j = 1, \ldots, \ell.$$

In the specific case of the biharmonic spline in 3D, if it is assumed that the polynomial part of the RBF in Equation (5) has the form $p(\boldsymbol{x}) = c_1 + c_2 x + c_3 y + c_4 z$, then

$$A_{i,j} = |\boldsymbol{x}_i - \boldsymbol{x}_j|, \qquad i, j = 1, \ldots, N,$$

P is the matrix with ith row $(1, x_i, y_i, z_i)$, $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_N)^\mathsf{T}$ and $\boldsymbol{c} = (c_1, c_2, c_3, c_4)^\mathsf{T}$.

Solving the linear system (8) determines $\boldsymbol{\lambda}$ and $\boldsymbol{c}$, and hence $s(\boldsymbol{x})$. However, the matrix B in Equation (8) typically has poor conditioning as the number of data points N gets larger. This means that substantial errors will easily creep into any standard numerical solution.

On initial inspection, the essentially local nature of the Gaussian, inverse multiquadric ($\phi(r) = (r^2 + c^2)^{-1/2}$) and compactly supported basic functions appear to lead to more desirable properties in the RBF. For example, the matrix B now has special structure (sparsity) which can be exploited by well-known methods and evaluation of Equation (6) only requires that the sum be over nearby centers instead of all N centers. However, non-compactly supported basic functions are better suited to extrapolation and interpolation of irregular, non-uniformly sampled data. Indeed, numerical experiments using Gaussian and compactly supported piecewise polynomials for fitting surfaces to point-clouds have shown that these basic functions yield surfaces with many undesirable artifacts in addition to the lack of extrapolation across holes.

The energy minimisation properties of biharmonic splines make them well suited to the representation of 3D objects. Since the corresponding basic function $\phi(r) = r$ is not compactly supported and grows arbitrarily large as r tends to infinity, the corresponding matrix B of Equation (8) is not sparse and, except for symmetry, has no obvious structure that can be exploited in solving the system. Storing the lower triangle of matrix B requires space for $N(N+1)/2$ real numbers. Solution via a symmetric solver will require $N^3/6 + \mathcal{O}(N^2)$ flops. For a problem with $20,000$ data points this is a requirement for approximately 1.6×10^9 bytes (1.5GB) of core memory, and 10^{13} flops, which is impractical. Furthermore, ill-conditioning of the matrix B is likely to make any solution one gets from such a direct computation highly unreliable. Thus, it is clear that direct methods are inappropriate for problems with $N \gtrsim 2,000$. Moreover, a single direct evaluation of Equation (6) requires $\mathcal{O}(N)$ operations. These factors have led many authors to conclude that, although RBFs are often the interpolant of choice, they are only suitable for problems with at most a few thousand points [13, 14, 20]. The fast methods described in the following section demonstrate that this is no longer the case.

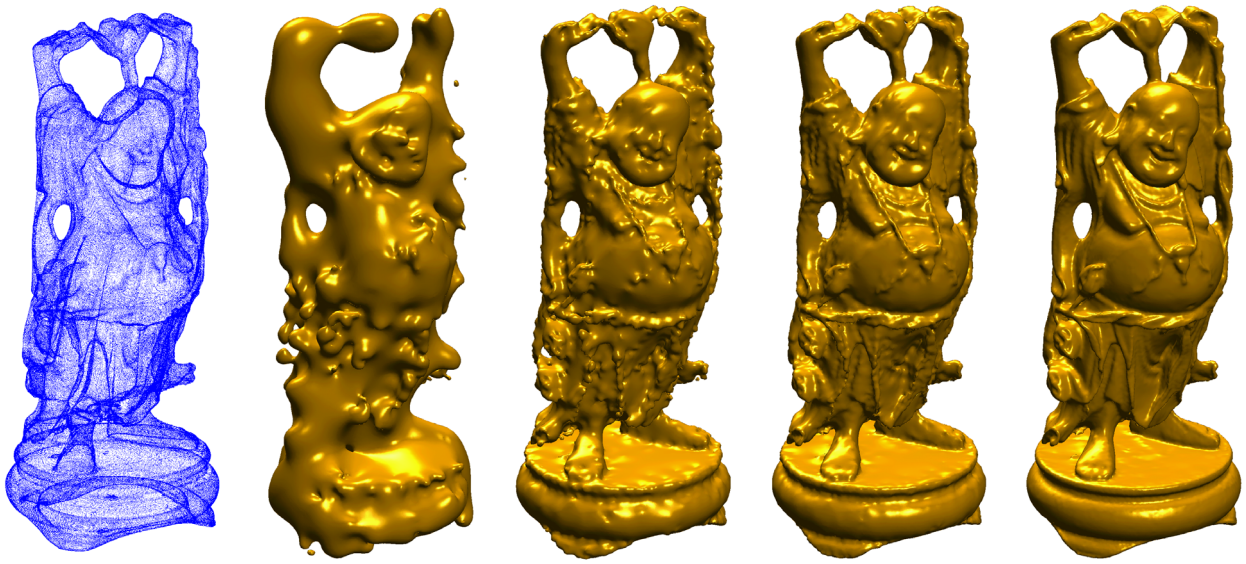


Figure 6: A greedy algorithm iteratively fits an RBF to a point cloud resulting in fewer centers in the final function. In this case the 544,000 point cloud is represented by 80,000 centres to a relative accuracy of 5×10^{-4} in the final frame.

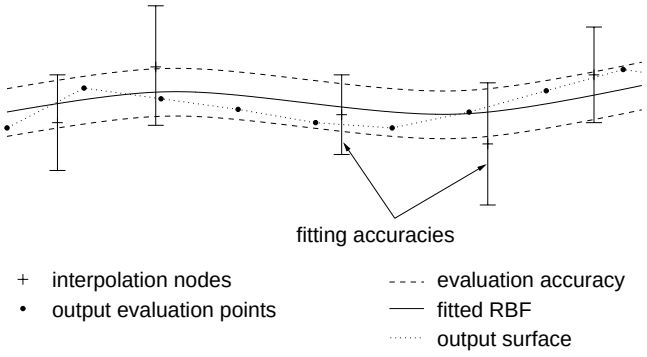


Figure 5: Illustration of the fast fitting and evaluation parameters.

	Direct methods	Fast methods
Fitting		
storage required	$N(N+1)/2$	$\mathcal{O}(N)$
flops to solve system	$N^3/6 + \mathcal{O}(N^2)$	$\mathcal{O}(N \log N)$
Evaluation		
flops per evaluation	$\mathcal{O}(N)$	$\mathcal{O}(1)$ after $\mathcal{O}(N \log N)$ setup

Table 1: Comparison of direct and fast methods.

4 Fast methods

Fast evaluation of RBF's is performed via the Fast Multipole Method (FMM) of Greengard & Rokhlin [15]. The FMM was designed for the fast evaluation of potentials (harmonic RBF's) in two and three dimensions. However Beatson *et al.* [7] have adapted the expansion and translation theory for the potential to higher order polyharmonic RBFs. Note that polyharmonic RBFs include the biharmonic spline of Equation (5). The FMM may also be used with polyharmonic splines in two [5] and four dimensions [3].

A full description of the FMM is beyond the scope of this paper and the interested reader is directed towards the introductory short course [4] for an expository treatment of the FMM. However, we give a brief outline of the method.

The FMM makes use of the simple fact that when computations are performed, infinite precision is neither required nor expected. Once this is realised, the use of approximations is allowed. For the evaluation of an RBF, the approximations of choice are far- and near-field expansions. With the centers clustered in a hierarchical manner, far- and near-field expansions are used to generate an approximation to that part of the RBF due to the centers in a particular cluster. A judicious use of approximate evaluation for clusters "far" from an evaluation point and direct evaluation for clusters "near" to an evaluation point allows the RBF to be computed to any predetermined accuracy and with a significant decrease in computation time compared with direct evaluation.

These fast evaluation methods, when used together with fitting methods particular to RBFs [2, 6], greatly reduce the storage and computational costs of using RBFs. They reduce the cost of evaluating $s(\boldsymbol{x})$ at M points from $\mathcal{O}(MN)$ to $\mathcal{O}(M + N \log N)$ operations. The cost of simultaneously computing the gradient $\nabla s(\boldsymbol{x})$ with $s(\boldsymbol{x})$ is approximately twice that of computing $s(\boldsymbol{x})$ alone. Table 1 summarises the gains of fast methods over direct methods.

Fig. 5 illustrates two parameters introduced by fast methods: a *fitting accuracy* and an *evaluation accuracy*. The fitting accuracy specifies the maximum allowed deviation of the fitted RBF value from the specified value at the interpolation nodes. If desired, a different fitting accuracy may be specified at each data point, as illustrated by the varying error bars in Fig. 5. The evaluation accuracy specifies the precision with which the fitted RBF is then evaluated. Fig. 5 shows an RBF evaluated at regular intervals lying between the dashed evaluation accuracy bands either side of the actual function. It is sensible to choose the evaluation accuracy to be numerically smaller than the fitting accuracy. The resulting dotted curve is an approximation to the smooth and continuous RBF.

5 RBF center reduction

Conventionally, an RBF approximation uses all the input data points (the $\boldsymbol{x}_i$'s in Equation (2)) as nodes of interpolation, and as centers of the RBF. However, the same input data may be able to be approximated to the desired accuracy using significantly fewer

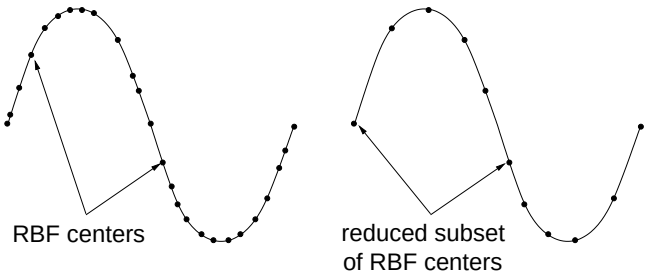


Figure 7: Illustration of center reduction.

centers, as illustrated in Fig. 7. A greedy algorithm can therefore be used to iteratively fit an RBF to within the desired *fitting accuracy*.

A simple greedy algorithm consists of the following steps:

1. Choose a subset from the interpolation nodes $\boldsymbol{x}_i$ and fit an RBF only to these.
2. Evaluate the residual, $\epsilon_i = f_i - s(\boldsymbol{x}_i)$, at all nodes.
3. If $\max\{|\epsilon_i|\} <$ *fitting accuracy* then stop.
4. Else append new centers where ϵ_i is large.
5. Re-fit RBF and goto 2.

If a different accuracy δ_i is specified at each point, then the condition in step 3 may be replaced by $|\epsilon_i| < \delta_i$.

Center reduction is not essential when using the fast methods described in Section 4. For example, no reduction was used when fitting to the LIDAR example of Fig. 8. However, reducing the number of RBF centers results in smaller memory requirements and faster evaluation times, without a loss in accuracy. Fig. 6 illustrates the fitting process with center reduction. As more centers are added to the RBF, the zero-surface more closely approximates the entire set of data points. In this case, a laser scan of a Buddha figurine, consisting of 544,000 points, has been approximated by an RBF with 80,000 centers to a relative accuracy of 1.4×10^{-4} (achieved at all the data points).

The greedy algorithm often results in a net faster fitting time, even with a moderate reduction in the number of centers. This is due to the efficiencies associated with solving and evaluating a similar system at each iteration and the fact that initial iterations involve solving much smaller problems. The results presented in Section 8 are typical of our experience and show that dense laser scan data can be represented by significantly fewer centers than the total number of data points.

6 RBF approximation of noisy data

In Section 3 we looked for an interpolant that minimized a measure of smoothness. However, if there is noise in the data, the interpolation conditions of Equation (2) are too strict and we would prefer to place more emphasis on finding a smooth function, where smoothness is measured by Equation (4). Thus, consider the problem

$$\min_{s \in \mathrm{BL}^{(2)}(\mathbb{R}^3)} \rho \|s\|^2 + \frac{1}{N} \sum_{i=1}^{N} \big(s(\boldsymbol{x}_i) - f_i\big)^2, \tag{9}$$

where $\rho \geq 0$ and $\|\cdot\|$ is defined in Equation (4). The parameter ρ balances smoothness against fidelity to the data. It can be shown [24] that the solution $s^\star$ to this problem also has the form of Equation (5), but now the coefficient vector $(\boldsymbol{\lambda}^\mathsf{T}, \boldsymbol{c}^\mathsf{T})^\mathsf{T}$ is the solution to

$$\begin{pmatrix} A - 8N\pi\rho I & P \\ P^\mathsf{T} & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}, \tag{10}$$

where the matrices A and P are as in Equation (8). The parameter ρ can be thought of as the stiffness of the RBF $s(\boldsymbol{x})$. The system (10) can also be solved using fast methods.

In Fig. 8 we illustrate RBF approximation (also known as spline smoothing) in the context of reconstructing a surface from 3D LIDAR data. LIDAR data is often noisy and irregular due to limited range resolution and misregistration between scans taken at different scanner positions. Restricted physical viewing angles for the scanner mean increased susceptibility to occlusion and therefore regions of incomplete data. Elsewhere, due to overlapping scans, the data may contain redundancy. Consequently, LIDAR scans represent one of the more difficult surface fitting problems. In this example a smooth surface has been automatically fitted to LIDAR scans of a fountain in Santa Barbara using an RBF approximation. The statue is approximately 2m×5m and was scanned using a CYRAX 2400 scanner. The data set consists of 350,000 points imaged from several viewpoints in front of the statue. Three spheres apparent on top of and beside the statue correspond to landmarks added to the scene in order to align the scans. Fig. 8(a) is the point-cloud taken from several scanner positions in front of the statue. Note the large occluded regions where no data has been recorded. Fig. 8(b) is the automatically fitted RBF surface. Fig. 8(c) is a detailed view of the fitted surface which illustrates how the smoothness constraint inherent in the biharmonic RBF has correctly preserved the gap between the arm and the rest of the statue, despite having no data in these regions. Along the cut-away planes in this figure we show the magnitude of the RBF. Lighter colored points have smaller magnitudes, *i.e.*, they are "closer" to the zero-surface than darker points.

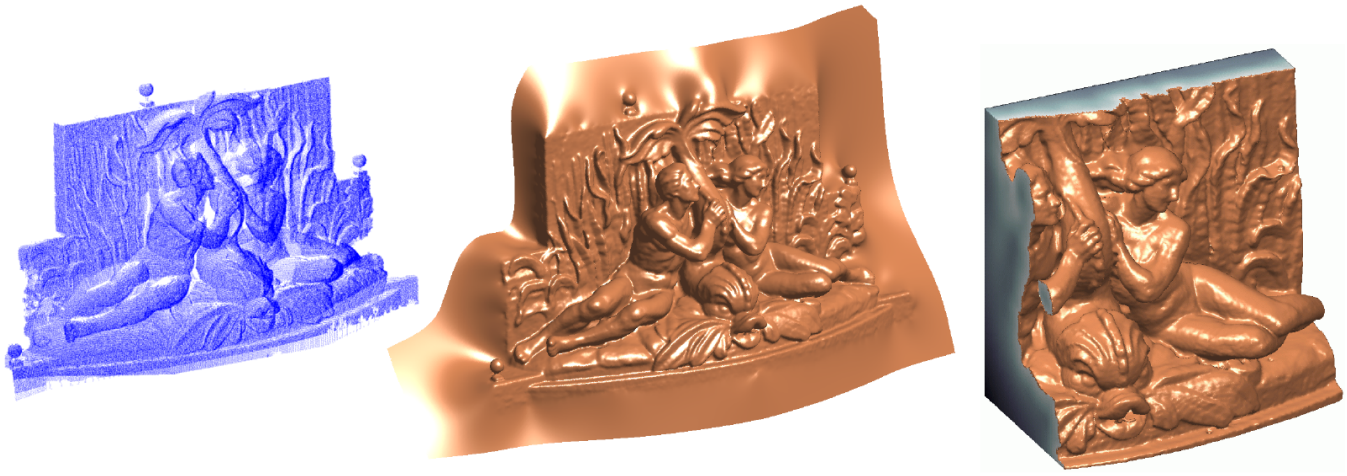
Fig. 9 illustrates the affect of varying ρ on a detailed portion of the surface in Fig. 8. Fig. 9(a) is an exact fit to the raw data ($\rho = 0$), Fig. 9(b) is the value of ρ corresponding to Fig. 8(b) and Fig. 9(c) illustrates increased smoothing with a larger value for ρ.

In this example a global value for ρ has been chosen, however, by dividing Equation (9) by ρ it is possible to specify a particular smoothing parameter ρ_i for each data point or group of points.

7 Surface evaluation

An RBF fitted to a set of surface data forms a *solid* model of an object. The surface of the object is the locus of points where the RBF is zero. This surface can be visualised directly using an implicit ray-tracer [9], or an intermediate explicit representation, such as a mesh of polygons, can be extracted. In the latter case, well-known iso-surfacing algorithms such as Marching Cubes [17] can be used to polygonize the surface. However, conventional implementations are optimised for visualising a complete volume of data sampled on a regular voxel grid. The cost associated with evaluating an RBF means that an efficient surface-following algorithm is desirable.

In this paper a marching tetrahedra variant, optimised for surface following, has been used to polygonize surfaces. A mesh optimisation scheme adapted from [21] results in fewer triangles with better aspect ratios, *i.e.*, thin and elongated triangles are avoided. A typical output mesh from this algorithm is illustrated in Fig. 10(b). Wavefronts of facets spread out from seed points across the surface until they meet or intersect the bounding box. For clarity, a wavefront from a single seed is shown in red in Fig. 10(a) spreading over the surface of the Buddha figurine during iso-surfacing. Surface-following is initiated from seed points corresponding to RBF centers. By design, many centers will lie on, or very near, the surface. In the case of off-surface centers, the RBF gradient is used to search



(a) (b) (c)

Figure 8: RBF approximation of noisy LIDAR data. (a) 350,000 point-cloud, (b) the smooth RBF surface approximates the original point-cloud data, (c) cut-away view illustrating the RBF distance field and the preservation of the gap between the arm and the torso.

(a) (b) (c)

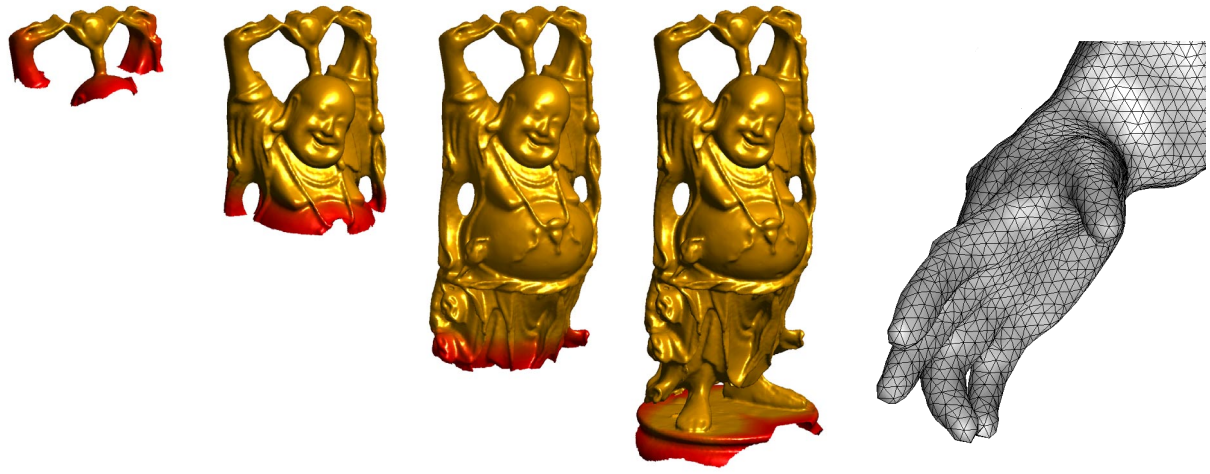
Figure 9: (a) Exact fit, (b) medium amount of smoothing applied (the RBF approximates at data points), (c) increased smoothing.

for the nearest zero-crossing. Convergence is rapid because the gradient is approximately constant near the surface. Local minima have not been observed in our experience, but this is not surprising since we deliberately constructed the function in Section 2 to have these properties. In any case, only a small subset of centers is required to seed the surface, one for each distinct surface section. The surface-following strategy avoids the conventional requirement for a 3D array of sample points and therefore minimises the number of RBF evaluations. Consequently, the computational cost increases with the square of the resolution, rather than the cube, as it would if a complete volume were sampled. Memory overhead is also minimised since it is only necessary to retain the sample vertices associated with the advancing wavefronts. Quadrilateral meshes can also be produced in this manner. Surfacing ambiguities can be resolved easily due to the ability to analytically evaluate the gradient of the RBF.

8 Results

Table 2 quantifies the fitting and evaluation times for the figures presented in this paper. In all cases the biharmonic spline was fitted. Two off-surface points were generated for every second point in the original surface data, hence the number of interpolation nodes to which an RBF is fitted is approximately twice the number of surface points. Center reduction was used throughout, except in the LIDAR example, where the number of RBF centers consequently equals the number of interpolation nodes. Fitting and evaluation was performed on a 550MHz PIII Pentium with 512MB RAM. Figs. 1(a), 3, 6, 12, 13 and 14 illustrate fitting surfaces to point-clouds while Figs. 1(b) and 11 illustrate fitting to partial meshes. Fig. 8 demonstrates approximation with an RBF in the context of fitting a smooth surface to noisy LIDAR data.

The dragon in Fig. 1(a), the Buddha figurine (Fig. 6) and the skeleton hand (Fig. 12) demonstrate the ability of fast methods to model large complicated data sets to high accuracy and the compact nature of the RBF representation. The dragon data was derived from a mesh consisting of 438,000 vertices and 871,000 faces, the Buddha from 544,000 vertices and 1,087,000 faces. Normals were computed at vertices from the adjacent faces. Direct methods would inevitably fail on these problems. For example, a direct approach to fitting the Buddha data would require 4,700GB of storage just to store the matrix of the interpolation system (8). The peak core memory requirements of the new fast methods in Table 2 are par-



(a) (b)

Figure 10: Iso-surfacing an RBF. (a) Surface-following from a single seed, (b) example of an optimised mesh.

Figure	Number of surface points	Number of inter--polation nodes	Number of RBF centers	Peak RAM (MB)	Fitting time	Surfacing time	Relative accuracy
Face	14,806	29,074	3,564	29	68s	27s	7×10^{-4}
Hand	13,348	26,696	4,299	29	97s	32s	1×10^{-3}
Dragon	437,645	872,487	72,461	306	2:51:09	0:04:40	8×10^{-4}
Buddha	543,652	1,086,194	80,518	291	4:03:26	0:04:07	5×10^{-4}
Cherub statue	331,135	662,269	83,293	187	3:09:06	0:06:41	4×10^{-4}
Skeleton hand	327,323	654,645	85,468	188	3:08:44	0:04:04	3×10^{-4}
LIDAR statue	345,910	518,864	518,864	390	3:08:21	0:25:39	6×10^{-3}

Table 2: Comparison of RBF fitting and evaluation times on a 550MHz PIII with 512MB RAM.

ticularly noteworthy in this respect.

Fig. 1(b) and Fig. 11 illustrate the application of RBFs to mesh repair. In Fig. 1(b) an RBF is fitted to a partial mesh obtained from a laser scanner. This figure demonstrates the ability of the biharmonic spline to smoothly interpolate across large irregular holes, for example under the chin, and smoothly extrapolate a surface far from the data. In Fig. 11 an RBF has been fitted to a large statue data set. Although carefully scanned, the statue contains many small holes and larger holes corresponding to occluded regions between the embracing figures. The fitted RBF has automatically filled all the holes and generated a water-tight model of the statue without the user having to specify any parameters other than the desired fitting accuracy. Note how the fragment of shoulder data on the right figurine has been extrapolated to reconstruct the missing chest data.

Fig. 13 illustrates the reconstruction of the asteroid Eros from scattered range data. This is a good example of non-uniformly distributed data, often difficult to reconstruct using other methods. Furthermore, in this example the accuracy associated with the range measurements varies at each point.

Table 3 compares the size of the original meshes in the dragon, Buddha and skeletal hand data sets with the size of the equivalent RBF representation. The uncompressed mesh sizes were derived by assigning 3 floats (12 bytes) to each vertex and 3 integers (12 bytes) to each triangular facet. The uncompressed RBF file size corresponds to representing each center with 3 floats (12 bytes) and each coefficient (λ_i) with a double precision number (8 bytes). It is likely that single precision would be adequate, which would result in more compression, but we have not yet quantified the effect of coefficient precision on the evaluated surface. This table demonstrates the significant compression of both point-cloud and mesh data that an RBF representation offers. We have also listed the size of the meshes derived from evaluating the RBF at a comparable resolution to the original data. It appears from initial results that a promising application of RBFs is the remeshing of existing meshes. Fitting an RBF not only fills holes, but a more uniform (and hence more compact) mesh can be derived from it.

Finally, for the doubting reader, we demonstrate in Figure 14 that a single RBF can model an extremely complicated surface with a high genus. In this case 594,000 centers were required to model the turbine blade to 10^{-4} accuracy. The fast evaluation methods described in Section 4 are essential for working with RBFs of this size. The RBF has reproduced intricate internal structure along with surface detail present in the original data. The point-cloud data were only available as a Marching Cubes (MC) mesh derived from X-ray CT data. However, if the CT data were available, then we would fit an RBF directly to the CT values. Specifically, we would fit to a subset of the data with values in the vicinity of the threshold used by Marching Cubes to iso-surface the data. This would result in a smaller fitting problem since the MC mesh contains extraneous vertices and the construction of a signed-distance function, which requires adding off-surface points, would be avoided. Smooth RBF interpolation of the CT data would also be preferable to the linear interpolation used in the MC algorithm, particularly between CT slices, which are usually spaced further apart relative to the pixel resolution within a slice. An approximating spline could also be used to reduce noise in the CT data.

9 Summary and future work

Fast methods make it computationally possible to represent complicated objects of arbitrary topology by RBFs. The scale-

Figure	Original mesh # vertices	# facets	Storage	New mesh # vertices	# facets	Storage	RBF representation RBF centers	Storage
Dragon	437,645	847,414	15.4MB	126,998	254,016	4.5MB	72,461	1.4MB
Buddha	543,652	1,086,798	19.6MB	96,766	193,604	3.5MB	80,518	1.6MB
Skeleton hand	327,323	654,666	11.8MB	81,829	163,698	2.9MB	85,468	1.7MB

Table 3: Comparison of storage requirements for RBF representations and derived meshes.

Figure 11: An RBF has automatically filled small holes and extrapolated across occluded regions in the scan data (left), to produce a closed, water-tight model (right). The complex topology of the statue has been preserved.

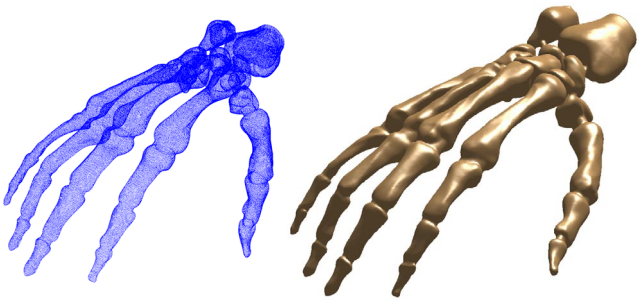


Figure 12: With sufficient sampling, complicated objects can be represented with RBFs.

independent, "smoothest interpolator" characterisation of polyharmonic splines make RBFs particularly suited to fitting surfaces to non-uniformly sampled point-clouds and partial meshes that contain large irregular holes. The smoothest surface, most consistent with the input data, is produced. Details are resolved, provided they are adequately sampled. The functional nature of the RBF representation offers new possibilities for surface registration algorithms, mesh simplification, compression and smoothing algorithms.

Within a Constructive Solid Geometry (CSG) framework the RBF offers a new way of modeling real-world (scanned) objects since it is inherently a solid model. It can be manipulated through a series of Boolean unions and intersections with other objects in a manner similar to how simpler geometric primitives are currently used to construct more complicated objects.

RBFs are also relevant to the problem of visualising volume data

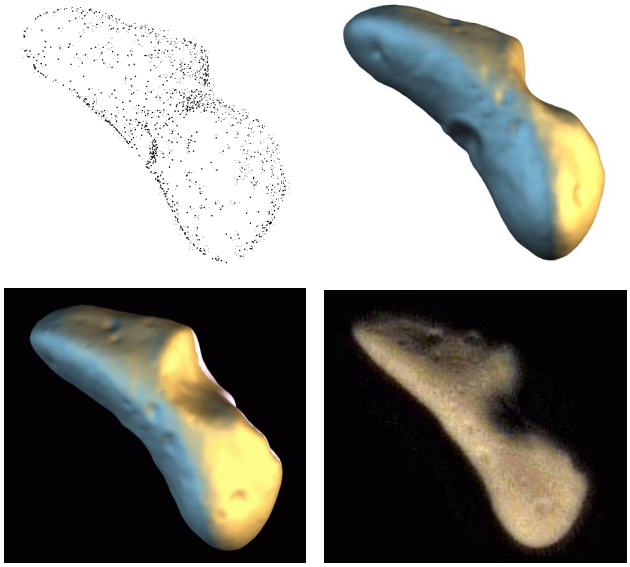


Figure 13: RBF reconstruction of the asteroid Eros from non-uniformly distributed range data (top). Photograph and model from a similar view (bottom).

acquired on an irregular grid, as can occur in medical imaging and geophysical data. Here an RBF can be fitted to the sampled data in order to approximate the underlying scalar distribution. Vector fields can be modeled easily if the components are independent. In that case, an RBF can be fitted to each field component. This does not require much extra computation time since the matrix in Equation (8) depends only on the location of the interpolation nodes and is therefore common to each component RBF.

Planned future work includes improving the performance of the reduction algorithm presented in Section 5. We have already achieved better reductions in the number of centers than those given in Table 2 but at the cost of slower fitting times. For example, the dragon has been modeled by as few as 32,000 centers and the Buddha by 42,000 centers, to the same fitting accuracy. This improvement was achieved by simply adding fewer centers to the RBF at each iteration. The global nature of the RBF representation implies RBFs inherit the drawbacks of any global model when manipulation of part of that model or ray-tracing is required. However, we believe that it should be straightforward to decompose a global RBF description into a piecewise mesh of implicit surface patches in the spirit of [1] to facilitate local manipulation and ray-tracing. Future speed improvements in computing and evaluating RBFs are likely to come from the inherent suitability of the algorithms to parallel processing and by tailoring to particular applications. In particular, we believe that significant speed improvements are likely if the data lie on a grid, albeit an incompletely sampled grid.

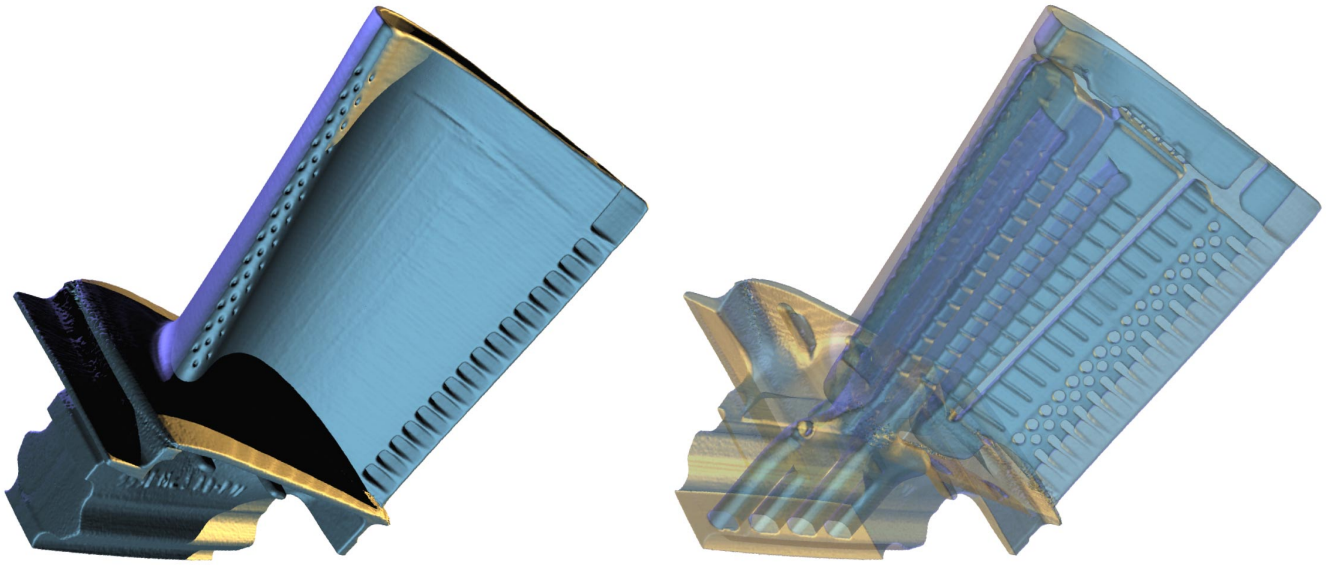


Figure 14: Solid and semi-transparent renderings of an RBF model of a turbine blade containing intricate internal structure. The RBF has 594,000 centers.

10 Acknowledgements

This research has been supported by FRST contracts DRFX-9902, DRFX-0001 and ARA-801. We are grateful to Stanford Computer Graphics Laboratory for the Buddha and Dragon data, to Cornell University and NASA for the Eros data, to Georgia Institute of Technology and the Large Geometric Models Archive for the turbine and skeleton data, to Allen Instruments and Supplies, USA, for the LIDAR data and Polhemus *FastSCAN* for all other data.

References

[1] C. Bajaj, J. Chen, and G. Xu. Modeling with cubic a-patches. *ACM Transactions on Computer Graphics*, 14(2):103–133, 1995.

[2] R. K. Beatson, J. B. Cherrie, and C. T. Mouat. Fast fitting of radial basis functions: Methods based on preconditioned GMRES iteration. *Advances in Computational Mathematics*, 11:253–270, 1999.

[3] R. K. Beatson, J. B. Cherrie, and D. L. Ragozin. Fast evaluation of radial basis functions: Methods for four-dimensional polyharmonic splines. *SIAM J. Math. Anal.*, 32(6):1272–1310, 2001.

[4] R. K. Beatson and L. Greengard. A short course on fast multipole methods. In M. Ainsworth, J. Levesley, W.A. Light, and M. Marletta, editors, *Wavelets, Multilevel Methods and Elliptic PDEs*, pages 1–37. Oxford University Press, 1997.

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

[6] R. K. Beatson, W. A. Light, and S. Billings. Fast solution of the radial basis function interpolation equations: Domain decomposition methods. *SIAM J. Sci. Comput.*, 22(5):1717–1740, 2000.

[7] R. K. Beatson, A. M. Tan, and M. J. D. Powell. Fast evaluation of radial basis functions: Methods for 3-dimensional polyharmonic splines. In preparation.

[8] F. Bernardini, C. L. Bajaj, J. Chen, and D. R. Schikore. Automatic reconstruction of 3D CAD models from digital scans. *Int. J. on Comp. Geom. and Appl.*, 9(4–5):327, Aug & Oct 1999.

[9] J. Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, San Francisco, California, 1997.

[10] J. C. Carr, W. R. Fright, and R. K. Beatson. Surface interpolation with radial basis functions for medical imaging. *IEEE Trans. Medical Imaging*, 16(1):96–107, February 1997.

[11] E. W. Cheney and W. A. Light. *A Course in Approximation Theory*. Brooks Cole, Pacific Grove, 1999.

[12] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In W. Schempp and K. Zeller, editors, *Constructive Theory of Functions of Several Variables*, number 571 in Lecture Notes in Mathematics, pages 85–100, Berlin, 1977. Springer-Verlag.

[13] N. Dyn, D. Levin, and S. Rippa. Numerical procedures for surface fitting of scattered data by radial functions. *SIAM J. Sci. Stat. Comput.*, 7(2):639–659, 1986.

[14] J. Flusser. An adaptive method for image registration. *Pattern Recognition*, 25(1):45–54, 1992.

[15] L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. *J. Comput. Phys*, 73:325–348, 1987.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstuction from unorganized points. *Computer Graphics (SIGGRAPH'92 proceedings)*, 26(2):71–78, July 1992.

[17] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987.

[18] C. A. Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constr. Approx.*, 2:11–22, 1986.

[19] 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.

[20] R. Sibson and G. Stone. Computation of thin-plate splines. *SIAM J. Sci. Stat. Comput.*, 12(6):1304–1313, 1991.

[21] G. M. Treece, R. W. Prager, and A. H. Gee. Regularised marching tetrahedra: improved iso-surface extraction. *Computers and Graphics*, 23(4):583–598, 1999.

[22] G. Turk and J. F. O'Brien. Shape transformation using variational implicit surfaces. In *SIGGRAPH'99*, pages 335–342, Aug 1999.

[23] G. Turk and J. F. O'Brien. Variational implicit surfaces. Technical Report GIT-GVU-99-15, Georgia Institute of Technology, May 1999.

[24] G. Wahba. *Spline Models for Observational Data*. Number 59 in CBMS-NSF Regional Conference Series in Applied Math. SIAM, 1990.

[25] G. Yngve and G. Turk. Creating smooth implicit surfaces from polygonal meshes. Technical Report GIT-GVU-99-42, Georgia Institute of Technology, 1999.