Volume xx (*200y*), *Number z, pp. 1–12*

Surface Reconstruction Based on Hierarchical Floating Radial Basis Functions

Jochen Süßmuth and Quirin Meyer and Günther Greiner

Computer Graphics Group, University Erlangen-Nuremberg, Germany

Abstract

In this paper we address the problem of optimal center placement for scattered data approximation using radial basis functions (RBF) by introducing the concept of floating centers. Given an initial least-squares solution, we optimize the positions and the weights of the RBF centers by minimizing a nonlinear error function. By optimizing the center positions, we obtain better approximations with a lower number of centers, which improves the numerical stability of the fitting procedure. We combine the nonlinear RBF fitting with a hierarchical domain decomposition technique. This provides a powerful tool for surface reconstruction from oriented point samples. By directly incorporating point normal vectors into the optimization process, we avoid the use of off-surface points which results in less computational overhead and reduces undesired surface artifacts. We demonstrate that the proposed surface reconstruction technique is as robust as recent methods which compute the indicator function of the solid described by the point samples. In contrast to indicator function based methods, our method computes a global distance field which can directly be used for shape registration.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling—

1. Introduction

Many applications in engineering and science build on accurate digital models of real-world objects. Typical examples include the digitalization of manufactured parts for quality control, statues and artifacts in archeology, or human bodies for movies and video games. Using modern 3D scanners, it is possible to acquire point clouds containing millions of points sampled from an object. As scanners are imperfect devices, scanned data usually contains noise. Occlusion leads to areas which are not visible to the scanner and thus to holes in the point cloud.

The process of building a geometric model (e.g. a polygonal mesh) from such point clouds is usually referred to as *surface reconstruction*. A good surface reconstruction algorithm should thereby adapt to varying sampling density, be able to deal with noise and smoothly interpolate holes in the point set.

In this paper we describe such an algorithm which is based on hierarchical nonlinear *Radial Basis Function (RBF)* fitting. Our main contributions are:

- We introduce the concept of floating centers for RBF scattered data approximation.
- We propose an efficient GPU implementation for nonlinear center optimization.
- We present a hierarchical approximation framework for surface reconstruction from oriented point samples.

Using our hierarchical approximation algorithm, we are able to quickly compute a distance field for large point clouds. We demonstrate the value of our method for surface reconstructions in three and four dimensions. Furthermore, we outline the prospective use of our algorithm for deformable shape registration.

1.1. Related Work

The problem of reconstructing a surface from unorganized points has long been discussed in the computer graphics and the computer vision communities. The algorithms developed in this process can be roughly classified in Delaunay based methods and implicit methods.

Delaunay based methods work on the Voronoi diagram

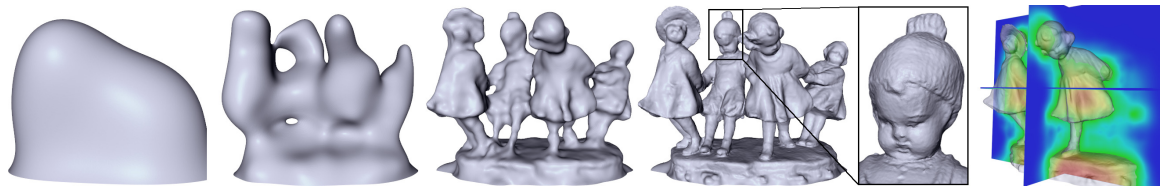

Figure 1: *Hierarchical reconstruction of the dancing children data set from 620.744 points with normal vectors:* (from left to right) *reconstructions at level 1, 3, 6 and 9 and the computed distance field.*

or its dual, the Delaunay triangulation, of the input points (see [CG06] for a detailed survey on Delaunay based methods). The benefit of many Delaunay based methods is that they are supported by rigorous theoretical guarantees which means that the output of the algorithms is proven to be correct under certain sampling conditions. On the other hand, Delauney based methods usually seek to interpolate the input data which makes them susceptible to noise.

Implicit methods seek to approximate the input points by the zero set of an implicit function f. A surface is then generated by extracting the zero set of the implicit function using marching cubes [LC87] or similar algorithms [Blo94]. The main advantage of implicit methods is that they decouple the complexity of the reconstructed shape from the size of the input point cloud which enables the reconstruction of data sets containing millions of points. Aside from that, they are more resilient to noise and guaranteed to produce closed manifold surfaces. Implicit methods mainly differ in the way how they represent and compute the function f. In their pioneering work, Hoppe et al. [HDD*92] define f to be a piecewise linear function which is obtained by computing the signed distance to the tangent plane of the closest point. Ohtake et al. [OBA*03] approximate the points locally by quadratic polynomials and blend these local reconstructions to produce a smooth global function. Another powerful class of implicit surface reconstruction methods uses radial basis functions (*RBF*) to describe the implicit function f: Carr et al. [CBC*01] use globally supported basis functions which they combine with a greedy algorithm for center reduction and a fast approximate method for fitting and evaluating of the RBF. In Walder et al. [WSC06], the input points and normal vectors are approximated by RBFs with local support. The centers and the support of the basis functions are chosen such that high frequency detail can be reconstructed in high curvature regions. Samozino et al. [SAAY06] propose to approximate the point cloud by compactly supported RBFs which are placed on the medial axis of the object. Ohtake et al. [OBS03] follow the idea of Floater and Iske [FI96] and compute a hierarchy $\mathcal{P}^1 \subset \mathcal{P}^2 \subset \ldots \subset \mathcal{P}^M = \mathcal{P}$ of the point set. Each level of the hierarchy is then interpolated by fitting RBFs of decreasing support to the residual of the underlying level.

Closely related to the above methods is another class of reconstruction algorithms which recently gained enormous popularity. Namely those which compute the indicator function of the solid enclosed by the point cloud [Kaz05, KBH06, ACSTD07, JR07, MPS08]. Such methods are usually very fast and resilient to noise. However, since they only compute an indicator function, they may not be used for applications which require a global distance field, e.g. non-rigid surface registration [LSP08, SWG08].

The surface reconstruction method presented in this paper falls into the category of RBF based implicit methods. The difference to previous RBF based methods is that we hierarchically approximate the data in a least-squares sense. The hierarchy is thereby based on locally refining the approximating function, thus combining the advantages of local and global approaches.

1.2. Overview

The remainder of this paper is structured as follows: In the next section, we show how scattered data approximation using radial basis functions can be used for surface reconstruction and introduce the idea of floating centers. In Section 3, we present an efficient GPU implementation for nonlinear RBF approximation. Section 4 suggests the use of a hierarchical approximation scheme to speed up the reconstruction procedure and to facilitate the reconstruction of large point sets. Results for three- and four-dimensional, i.e. time dependent, data sets are presented in Section 5. We conclude our work and outline several future research directions in Section 6.

2. Floating RBF Approximation

We start this section with a short introduction on scattered data approximation using RBFs in general. Later on, we will show how surface reconstruction from oriented points can be formulated as scattered data approximation problem. Finally, we present the concept of floating centers which we use to optimize the fitting result.

The scattered data *interpolation* problem can be formulated as follows:
Given a set of function values $\mathcal{F} = \{f_1, \ldots, f_n\} \in \mathbb{R}$, sampled at piecewise distinct locations $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \in \mathbb{R}^d$,

find the parameters of an analytic function $f : \mathbb{R}^d \rightarrow \mathbb{R}$ such that f minimizes some smoothness measure and interpolates the values f_i at positions $\mathbf{p}_i$, i.e.

$$f(\mathbf{p_i}) = f_i, \quad 1 \leq i \leq n. \tag{1}$$

In case of fitting radial basis functions, the function f is defined as the sum of m scaled and translated radial symmetric basis functions $\phi_{\mathbf{c}_j} : \mathbb{R}^d \rightarrow \mathbb{R}$ and a d-variate polynomial $P(\mathbf{x}) = \sum_{i=0}^{l} \beta_i \, \mathrm{b}_i(\mathbf{x})$ of low degree:

$$f(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j \phi_{\mathbf{c}_j}(\mathbf{x}) + P(\mathbf{x}), \tag{2}$$

where $\mathbf{c}_j$ are the centers/translates and α_j the weights/scales of the basis functions. The choice of the basis function $\phi_{\mathbf{c}}$ determines which smoothness energy is minimized, e.g. the generalized Duchon splines [Duc77] in three dimensions $\phi_{\mathbf{c}_j}(\mathbf{x}) = \left\|\mathbf{c}_j - \mathbf{x}\right\|^3$ minimize the thin plate energy.

RBF interpolation usually assumes that one basis function is located at each sample position, i.e. $m = n$ and $\mathbf{c}_j = \mathbf{p}_j$. The parameters $\alpha = \{\alpha_i\}$ and $\beta = \{\beta_i\}$ of the function f, which is in a sense smooth and meets the interpolation conditions (1) and the side conditions $\mathbf{P}^T \alpha = 0$ (see [CBC*01] for a detailed derivation), may then be computed by solving the linear system

$$\begin{bmatrix} \Phi & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \mathcal{F} \\ 0 \end{bmatrix}, \tag{3}$$

where

$$\begin{aligned} \Phi \;&\in\; \mathbb{R}^{n \times n}; \;\; \Phi_{i,j} = \phi_{\mathbf{p}_j}(\mathbf{p}_i) \;\text{ and} \\ \mathbf{P} \;&\in\; \mathbb{R}^{n \times l}; \;\; \mathbf{P}_{i,j} = \mathrm{b}_j(\mathbf{p}_i). \end{aligned} \tag{4}$$

As real-life data mostly contains noise, it is usually more desirable to *approximate* the data instead of interpolating it. Most papers [CBC*01, WSC06] do this by solving a regularized interpolation problem similar to Tikhonov regularization in statistics, i.e. they modify Equation (3) to be

$$\begin{bmatrix} \Phi + \varepsilon \, \mathbf{Id} & \mathbf{P} \\ \mathbf{P}^T & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \mathcal{F} \\ 0 \end{bmatrix}.$$

Id is the identity matrix and the parameter ε is used to balance smoothness against fitting accuracy: Large values for ε result in a smoother function and small values in a tighter fit.

The approach which we follow in this paper is to reduce the number of centers used for RBF approximation. Assume that we are given m center locations $\hat{\mathbf{c}}_j$, where $m \ll n$. Instead of enforcing the interpolation conditions (1), we seek the parameters $\{\alpha_j\}$ and $\{\beta_j\}$ of f such that the discrete least-squares error

$$E(f) = \sum_{k=1}^{n} \left\| f(\mathbf{p}_k) - f_k \right\|^2 \tag{5}$$

of the fit is minimized. Minimizing the error $E(f)$ corresponds to solving the linear system

$$\mathbf{A}^T \mathbf{A} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \mathbf{A}^T [\mathcal{F}], \tag{6}$$

with $\mathbf{A} = \begin{bmatrix} \hat{\Phi} & \mathbf{P} \end{bmatrix}$, where

$$\hat{\Phi} \in \mathbb{R}^{n \times m}; \;\; \hat{\Phi}_{i,j} = \phi_{\hat{\mathbf{c}}_j}(\mathbf{p}_i)$$

and $\mathbf{P}$ is defined as in Equation (4).

Reducing the number of centers offers two advantages over the regularization approach: Firstly, the size of the system matrix in Equation (6) is now roughly $(m \times m)$ instead of $(n \times n)$ as in Equation (3). Secondly, formulating the RBF fitting problem as an approximation problem allows us to naturally incorporate point normal vectors as we will show in the next section.

2.1. Computing an Initial Fit

To improve the readability, we restrict the discussion in this section to the case of fitting a surface to points in 3-dimensional space. The generalization of the formulas for fitting a d-dimensional hypersurface is straight forward. Let $\mathcal{P}$ be a set of n points, $\mathcal{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_n\}$, which have been sampled from a surface $\mathcal{M}$ by a 3D scanner. We assume that each point $\mathbf{p}_i$ is equipped with a unit normal vector $\mathbf{n}_i \in \mathcal{N} = \{\mathbf{n}_1, \ldots, \mathbf{n}_n\}$ which describes the orientation of $\mathcal{M}$ at $\mathbf{p}_i$. Most scanning devices infer these normal vectors from neighborhood relationships during scanning. If normal vectors are not provided by the scanner, they can be estimated as proposed in [HDD*92]. To make the fitting procedure scale invariant, we initially translate and scale the point cloud $\mathcal{P}$ such that all points are completely contained in the unit cube $[0,1]^3$.

We wish to find an implicit function f such that its zero set $f = 0$ approximates the points from $\mathcal{P}$. Thus, we set the target function values f_i to zero for all points $\mathbf{p}_i$. Next we need to compute the parameters of f such that the error-functional in Equation (5) is minimized.

To avoid the trivial solution that f is zero anywhere, Turk et al. [TO99] propose to add additional non-zero valued constraints inside or outside the surface. These so called *off-surface* points are new points which are created by moving points along the normal vectors and assigning them the signed distance to the surface as function value f_i. The problem with these manufactured off-surface points is that they may penetrate other surface sheets which means that further heuristics have to be employed to test whether an off-surface point is valid. It also increases the memory requirements and the costs for solving the interpolation/approximation problem.

Instead of using artificially generated off-surface points, we reformulate the approximation problem such that it directly incorporates the point normal vectors which have been

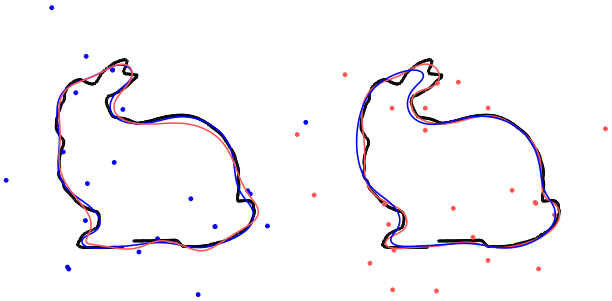

Figure 2: *Two different sets of RBF centers:* (left) *center placement optimized for fitting inverse quadratic basis functions* (blue line) *does not produce an optimal result when used with multiquadric basis functions* (red line) *and vice versa* (right).

recorded by the scanner. The gradient ∇f of an implicit distance function corresponds to the normal vectors of a level-set $f = c$. Thus, we can assume that the gradient of a function f, which is a good approximation to the distance function of $\mathcal{P}$, is also in accordance with the point normal vectors $\mathbf{n}_i$ at $\mathbf{p}_i$. We use this observation to extend the energy functional (5) by adding a term which punishes deviations of ∇f from the point normal vectors:

$$E(f) = \theta \sum_{i=1}^{n} \|f(\mathbf{p}_i) - f_i\|^2 + \bar{\theta} \sum_{i=1}^{n} \|\nabla f(\mathbf{p}_i) - \mathbf{n}_i\|^2, \quad (7)$$

where $\bar{\theta} = (1-\theta)$. The parameter θ can be used to balance between a good fit in the function values versus a good fit in the gradient. Using unit length normal vectors and assuming that the sample points are completely contained in the unit cube, we found that $\theta = 0.95$ is a good compromise and use this value for all data-sets shown in this paper.

The next step is to select a basis function ϕ and a set of centers $\{\hat{\mathbf{c}}_j\}$. As can be seen from Figure 2, the fitting accuracy is influenced by the placement of the centers which is in turn influenced by the choice of the basis function ϕ: Center positions which yield a good fit using a specific basis function do not necessarily produce a good approximation when combined with another basis function. Because of this dilemma, the final choice of center locations is postponed to the next section. While basically any basis function could be used, we choose ϕ to be the *inverse quadratic* (IQ) basis function:

$$\phi_{\mathbf{c}}(\mathbf{x}) = \frac{1}{1 + \|\mathbf{c} - \mathbf{x}\|^2 \delta^2}. \quad (8)$$

The IQ is infinitely smooth and contains a so called *shape parameter* δ. As polynomial P in Equation (2) we use a 3-variate linear polynomial. Thus, the function f which we seek to optimize is defined as

$$f(\mathbf{x}) = \sum_{j=1}^{m} \alpha_j \phi_{\hat{\mathbf{c}}_j}(\mathbf{x}) + \langle \mathbf{n}, \mathbf{x} \rangle + b. \quad (9)$$

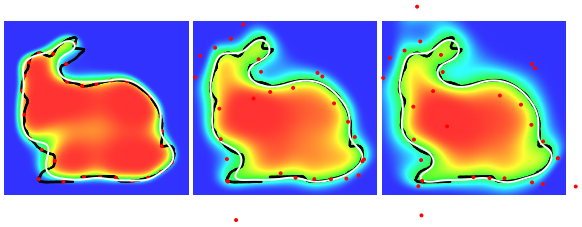

Figure 3: *Fitting a curve to 2D points:* (from left to right) *Initial fit, fit after 10 and 35 nonlinear optimization steps.*

For now, let us assume that we use a sub set of the input points $\mathcal{P}$ as centers: Say we have a budget of m centers, then we randomly select m points of $\mathcal{P}$ and use them as centers $\hat{\mathcal{C}} = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_m\} \in \mathcal{P}$.

Minimizing Equation (7) subject to the unknown parameters of the function f in (9) poses a linear least-squares problem. The solution can therefore be found by solving the normal equation

$$\mathbf{B}\mathbf{t} = \mathbf{c}, \quad (10)$$

where $\mathbf{t} = [\ \alpha_1, \ldots, \alpha_m, \mathbf{n}.x, \mathbf{n}.y, \mathbf{n}.z, b\ ]^T$ is the vector containing the unknowns. A detailed specification of the matrix $\mathbf{B}$ and the vector $\mathbf{c}$ can be found in the supplemental material to this paper.

The system of linear equations in (10) can be solved numerically stable using a singular value decomposition. The implicit function f in (9) is now fully defined and the approximating surface can be extracted by computing the zero set $f = 0$. The leftmost image of Figure 3 shows an example of an approximating curve in 2D. Two 3D examples of RBF approximations using fixed centers can be seen on the left side of the image in Figure 4.

2.2. Optimizing the Center Positions

The solution of Equation (10) gives the parameters of the best fitting function f for a given set of centers $\{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_m\}$. However, as previously mentioned, it is not clear how to select the center locations since their choice strongly depends on the shape of the original object and the used basis functions.

Previous work proposed various heuristics for placing the centers, e.g. on a sub set of the data points [CBC*01] or on a regular grid. Iske [Isk00] demonstrates that choosing maximal uniformly distributed centers is desirable as it reduces the condition number of the resulting linear system. In a more recent contribution, Samozino et al. [SAAY06] place the centers on the medial-axis of the object described by the point cloud. The authors give empirical results that their method generates better fits when used in conjunction with compactly supported Wendland basis functions [Wen95].

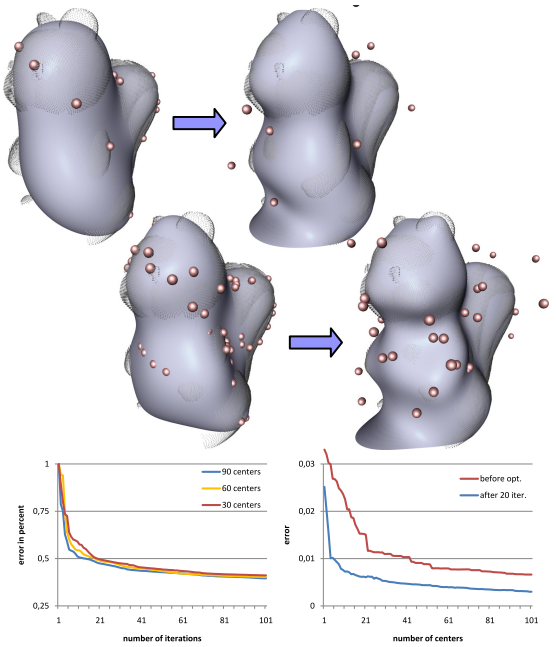

Figure 4: *Convergence behavior of the center optimization: The upper image shows the approximation using 40* (top) *and 90 centers* (bottom) *before and after 50 nonlinear optimization steps. The lower left plot shows the relative error over 100 iterations for 30, 60 and 90 centers, respectively. The lower right plot shows the average least-squares error before* (red line) *and after* (blue line) *20 iterations of center optimization depending on the number of centers.*

Instead of sticking to a certain heuristic for placing the centers, similarly to [FH99, JBL*06], we propose to consider the search for the most suitable center locations as an optimization problem itself. Thus, the least-squares error in Equation (7) is not only minimized subject to the weights and the polynomial P of the RBF, but to the positions of the centers as well. Due to the iterative nature of this minimization, we call the process *floating centers* optimization.

The RBF in (9) is not linear in the locations of the centers $\hat{\mathbf{c}}_j$. Hence, finding the set of centers $\hat{\mathbf{c}}_j$ which minimizes the least-squares error in (7) no longer boils down to solving a linear system. Instead, we use the Levenberg-Marquardt algorithm [Mar63] (*LMA*) to find the center locations and the RBF weights which minimize the least-squares error $E(f)$ in (7). Given a guess for the parameters $\mathbf{t}_0$ of f: $\mathbf{t}_0 = \left(\{\alpha_j\}, \{\hat{\mathbf{c}}_j\}, \mathbf{n}.x, \mathbf{n}.y, \mathbf{n}.z, b \right)^T$, an LMA iteration computes a solution $(\mathbf{t}_0 + \delta)$, which reduces the least-squares error, i.e. $E(f_{\mathbf{t}_0+\delta}) < E(f_{\mathbf{t}_0})$. Therefore, the system of linear equations

$$(\mathbf{C} + \lambda \operatorname{diag}(\mathbf{C}))\,\delta = \mathbf{g} \qquad (11)$$

is solved using a singular value decomposition again.† The damping factor λ is chosen such that $E(f_{\mathbf{t}_i+\delta})$ is guaranteed to decrease in each step. Then, the current guess is updated as $\mathbf{t}_{i+1} = (\mathbf{t}_i + \delta)$ and the iteration is continued until a local minimum is found or the maximum number of iterations has been reached. As initial guess for the center positions $\hat{\mathbf{c}}_j$, the weights α_j and the linear polynomial P, we use the linear solution to the least-squares problem which was computed as explained in the previous section.

The effect of our floating centers optimization for a simple 2D example can be seen in Figure 3: The leftmost image shows the fitting of an RBF with 30 centers to the Bunny data set and the reconstructed distance field. The red dots mark the positions of the centers and the white line is the zero isoline of the distance field. The center image and the image on the right show the approximation after 10 and 35 nonlinear optimization steps, respectively. The convergence behavior of our floating centers optimization is analyzed in Figure 4: We used our method to fit RBFs with up to 100 centers to the Squirrel data set consisting of 41K points. The average least-squares error after the initial fit as a function of the number of centers is shown as the red line in the right diagram. The error after 20 steps of our center optimization is shown as the blue line. It can be seen that the error is roughly halved after 20 iterations. This observation is confirmed by the left diagram which shows the normalized approximation error depending on the number of iterations for an RBF fit with 30, 60, and 90 centers.

3. Parallel Computation

The computationally most expensive step during the RBF fitting procedure is the assembly of the system matrices in Equations (10) and (11) as each matrix entry requires to sum over all points of the point cloud. Since the system matrix for the nonlinear optimization contains 16 times more elements than the matrix for the initial linear fit, one single floating centers optimization step is roughly ten times slower than the initial fit. In order to make the floating centers optimization applicable for real-life data sets, we present a fast parallel algorithm for computing the system matrices.

With the matrix elements being independent from each other, they can obviously be computed in parallel. However, the number of matrix elements is too small for fully utilizing modern massively parallel architectures like GPUs, that require tens of thousands of computational threads to run virtually in parallel in order to be efficient. Therefore, we propose to parallelize with the respect to the sample points $\mathbf{p}_k$. Note that all matrices and vectors in (10) and (11) adhere to the following computational pattern (see supplemental ma-

† The matrices $\mathbf{C}$ and $\mathbf{g}$ are detailed in the supplemental material.

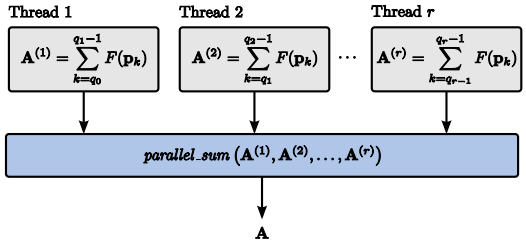

Figure 5: *Parallelized computation of the system matrices.*

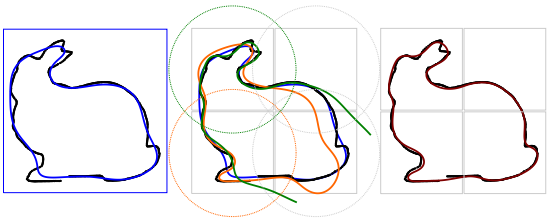

Figure 6: *Hierarchical Approximation:* (left) *level 1 fit;* (middle) *reconstruction of local increments;* (right) *global reconstruction a level 2.*

terial):

$$\mathbf{A} = \sum_{k=1}^{n} F(\mathbf{p}_k).$$

An overview of our parallel approach is depicted in Figure 5: We split up the summation into r independent sums:

$$\mathbf{A} = \sum_{i=0}^{r} \mathbf{A}^{(i)}, \text{ where } \mathbf{A}^{(i)} = \sum_{k=q_i}^{q_{i+1}-1} F(\mathbf{p}_k),$$

with $q_k = \lceil \frac{n}{r} \rceil \cdot k + 1$ and $q_r = n+1$. Each sub-sum $\mathbf{A}^{(i)}$ is computed independently. In a final step we accumulate the sub sums using a standard parallel reduction.

	Bunny		Galaad		Dancing	
	IF	OPT	IF	OPT	IF	OPT
CPU	1,620	232K	6,012	843K	2,565	362K
GPU	11	2,485	32	7,860	14	3,549

Table 1: *Timings (in ms) for fitting a RBF with 16 centers* (IF) *and 20 nonlinear optimization iterations* (OPT) .

We implemented our algorithm in CUDA [NVI09] and ran tests on different data sets. As can be seen from Table 1, our GPU version outperforms the CPU version by two orders of magnitude: Using the GPU implementation, we can compute an initial fit and 14 optimization steps in the time needed for computing the initial fit on the CPU.

4. Hierarchical Approximation

Fitting only a single level as proposed in the previous sections works well only for computing rough approximations of the point cloud. In order to capture small details of the shape, the number of centers would have to be dramatically increased. Increasing the number of centers for RBF fitting is undesirable for two reasons: first, the computation time is roughly quadratic in the number of centers. Second, increasing the number of centers will also increase the condition number of the system matrix in (10) which in turn leads to less reliable fits. One possibility to counter these problems is to use compactly supported basis functions such as Wendlands C^2 function: $\phi(r) = (1-r)_+^4(4r+1)$. Unfortunately, using compactly supported basis functions brings up other issues: They define the distance function only in a neighborhood of the input points and the nice property of extrapolation across holes will be lost.

To overcome the limitations mentioned above, we use an adaptive octree subdivision, where at each tree level we refine the previous approximation by fitting local offsetting functions [VMG09]. Let Ω_l^k be an octree cell at level k. Let $\mathbf{c}$ be the center of Ω_l^k and d the length of its diagonal. We define a region of influence for the cell as the sphere Θ_l^k with radius $(\alpha_{\text{radius}} \cdot d)$ centered at $\mathbf{c}$ (we use $\alpha_{\text{radius}} = 0.65$). Let $f_{k,i}^{res}$ be the residual (i.e. the fitting error) of point $\mathbf{p}_i$ at level k and $\mathbf{n}_{k,i}^{res}$ the residual of the gradient fit respectively.

For each octree cell Ω_l^k at level k, we fit a function consisting of 16 RBFs which approximates the residuals of the points in Θ_l^k. Thereby, we scale the shape parameter δ of the basis functions in (8) such that the $\phi(d)$ is constant, i.e. we double δ as we halve the cell size. This implies that small cells reconstruct high frequency detail while large cells in low octree levels reconstruct the base shape.

After all cells of an octree level k have been processed, we blend the local reconstructions together using the partition of unity approach of Ohtake et al. [OBA*03]. Thus, we obtain a global function f^k which approximates the residuals $f_{k,i}^{res}$ and $\mathbf{n}_{k,i}^{res}$. The residuals for the next level $(k+1)$ are now updated as

$$\begin{aligned} f_{(k+1),i}^{res} &= f_{k,i}^{res} - f^k(\mathbf{p}_i), \\ \mathbf{n}_{(k+1),i}^{res} &= \mathbf{n}_{k,i}^{res} - \nabla f^k(\mathbf{p}_i). \end{aligned}$$

Cells which contain a point $\mathbf{p}_i$ for which the error in the value or gradient fit is larger than a predefined threshold

$$\left(f_{(k+1),i}^{res} > \varepsilon_{\text{val}}\right) \vee \left(\left\|\mathbf{n}_{(k+1),i}^{res}\right\| > \varepsilon_{\text{grad}}\right)$$

are further subdivided. This recursion is repeated until no more cells are subdivided or a maximum tree depth is reached. The approximating function f_{tree} can now be evaluated by summing up over the level fits f^k:

$$f_{\text{tree}}(\mathbf{x}) = \sum_{k=1}^{\text{depth}} f^k(\mathbf{x})$$

We initialize the recursion with a single node containing the entire point set and the residuals are initially set to the target values, i.e.

$$\begin{aligned} \Omega_1^1 &= [0,1]^3, \\ f_{1,i}^{res} &= f_i = 0, \\ \mathbf{n}_{1,i}^{res} &= \mathbf{n}_i. \end{aligned}$$

A two-dimensional example of our hierarchical fitting algorithm is depicted in Figure 6: The leftmost image shows the fit at level 1. Local reconstructions can be seen in the middle image: The orange curve approximates the lower left cell well and the green curve provides a good fit for the upper left cell. The image on the right shows the fit at level two after the local reconstructions have been blended together. Reconstructions of a 3D point cloud at various octree depths are shown in Figure 1.

Our hierarchical approximation is conceptually similar to the *Multi-level Partition of Unity* (MPU) approach [OBA*03]. However, we would like to stress that we incrementally reconstruct the approximating function: While the MPU method discards the information from low level fits, our approach refines previous fits.

5. Results and Discussion

In this section we discuss the application of our method for surface reconstruction, reconstruction of time-varying point data and surface registration. The results presented here were computed using our hierarchical reconstruction, where in the first two levels we used the GPU implementation to fit 16 centers and to perform 10 floating centers optimization steps. For levels above 2 we used the CPU version to fit 6 centers for each cell and no nonlinear optimization was performed. All computations were performed on a 2.93 GHz Intel Core I7 workstation with 6GB of RAM and a Nvidia GeForce GTX 260 graphics card with 896 MB of RAM.

Figure 7: *Reconstructions of the* Neptune *statue and a miniature plastic figure* (Galaad).

5.1. Surface-Reconstruction

The first and most important application for our algorithm is the reconstruction of 3D models from registered range scans. Reconstruction from raw point data is especially challenging as the data usually contains noise, holes and misalignment artifacts.

The Neptune and the Galaad models in Figure 7 demonstrate the ability of our algorithm to reconstruct large, complicated models. The Neptun model was reconstructed from a point cloud containing 3,183,064 points which were taken from registered range scans. Some parts of the statue (e.g. the face) have been scanned at a higher resolution. Thus, the input point cloud contains areas of sharply varying sample density. The model was reconstructed at an octree depth 9 in 123 seconds at 1,135 MB peek memory and the surface was extracted in 8 seconds using the octree polygonizer from [KKDH07]. The Galaad model was acquired using a Kreon laser scanner. The Kreon scanner samples the object line-wise. Therefore, the resolution along the laser line is much finer than the resolution orthogonal to the laser line. The reconstruction using our method at tree depth 9 took roughly 67 seconds and 613 MB peek memory including isosurfacing.

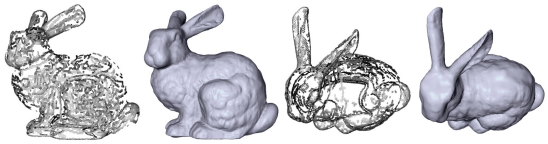

Figure 8: *Reconstruction from incomplete data.*

The ability of our algorithm to handle incomplete data is shown in Figure 8: Only the points which lie close to a crest line were used to reconstruct the Stanford Bunny model. The missing parts were smoothly interpolated by our method. In Figure 9, we demonstrate the effect of floating centers when used together with the hierarchical approximation: Without floating centers, we need to fit one or two additional levels in order to obtain the same fitting accuracy.

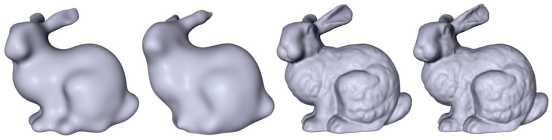

Figure 9: *Impact of floating centers* (*FC*). (from left to right)*: Reconstructions at level 3 with and without FC, fit at level 5 using FC and reconstruction at level 6 without FC.*

We compare the results obtained by our method to the results of the MPU method [OBA*03], the Poisson reconstruction [KBH06] and the Wavelet reconstruction [MPS08]

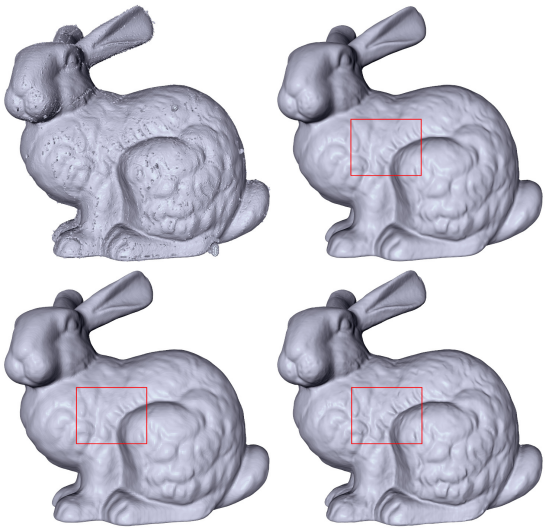

Figure 10: *Comparison with other methods:* (top left) *MPU,* (top right) *Poisson,* (bottom left) *Wavelet and* (bottom right) *our method.*

Method	Fit	Poly.	Peak Mem.	Grid
MPU	42s		411MB	500^3
Poisson	43s	16s	323MB	512^3
Wavelet	19s		122MB	512^3
Our method	36s	9s	135MB	512^3

Table 2: *The computation times and the memory usage of the methods compared in Figure 10.*

in Figure 10 and Table 2. We used the original implementations of these methods to reconstruct the Stanford Bunny from 391,769 raw points, thereby analyzing running times, memory consumption and the visual quality of the reconstructed model.

The reconstruction using MPU Implicits contains spurious surface sheets which originate from the completely local nature of the fitting method and, as mentioned in a follow-up paper by the same group, the fitting primitive. The Poisson reconstruction and the reconstruction using D4 wavelets, both at octree level 9, produce results which are comparable to our reconstruction. A closer look at the area marked by the red rectangle however unveils that both methods tend to oversmooth the result, whereas our method shows more contrast, hence better captures sharp features. The memory consumption and the running time for reconstructing the Bunny model are approximately the same for all methods. However, in contrast to the Poisson and the Wavelet reconstruction, our method does additionally compute a global distance field and can therefore be used for surface registration.

As it is difficult to assess the quality of a surface reconstruction algorithm when only raw points are available as input, we performed some artificial tests to compare our algorithm to Poisson surface reconstruction. Therefore, we sampled points and normal vectors directly from known polygon models (cf. Figure 11) and then reconstructed polygonal surfaces using our method and Poisson surface reconstruction. We then used Metro [CRS98] to compute for each vertex of the reconstructed mesh the distance to the original mesh and vice versa. The average errors for the tested models at various reconstruction depths are summarized in Table 3. It can been seen that, at comparable computation times, the fitting error of our method is usually only half of that when using Poisson surface reconstruction. Only in the presence of strong noise, Poisson reconstruction achieves better results.

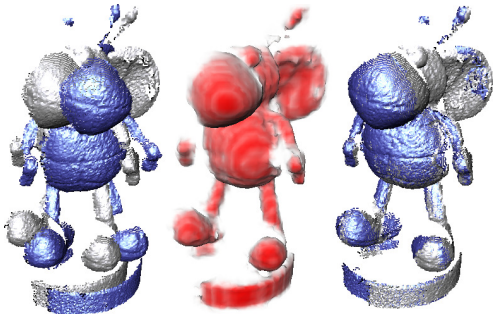

Figure 12: *Registering two scans of the bee data set:* (left) *Initial alignment,* (middle) *zero set of the computed distance function color-coded and transparency by confidence value, and* (right) *result of the rigid registration.*

5.2. Point Cloud Registration

To demonstrate the prospective use of our method for non-rigid surface registration, we have implemented a simple ICP based registration algorithm which uses the calculated distance field to iteratively align two overlapping point sets. Instead of searching for correspondences, we can directly minimize the distance between the point sets given by the distance function f. The distance is thereby weighted by a confidence value which is a smooth function of the point density. The confidence value is computed using the same hierarchical scheme as for computing the function value, except that we blend per-cell density estimates instead of function values. Our distance function could be directly integrated into the non-rigid registration methods of Li [LSP08] or Süßmuth [SWG08]. We illustrate the registration in Figure 12, where two range scans of a stuffed toy bee are aligned. The image on the left shows the initial configuration, the middle image shows the zero set of the reconstructed distance function for the first data set, which is colored according to the confidence value. The rightmost image shows the final alignment after 7 iterations.

Figure 11: *Artificial data sets that were used to evaluate the fitting accuracy of the proposed algorithms. Reconstructions are shown in the supplemental material.*

Data-Set				Poisson		Our approach	
Name	#Pts.	Noise	Tree Depth	Time	Avg. Error	Time	Avg. Error
Dancing Children	1M	0%	7	5.6 s	0.289992	33.8 s	0.125806
Dancing Children	1M	0%	8	18.8 s	0.104706	42.8 s	0.045044
Dancing Children	1M	0%	9	78.7 s	0.040073	60.8 s	0.019344
Dancing Children	1M	0.1%	9	91.3 s	0.051764	66.2 s	0.046558
Dancing Children	1M	0.2%	9	122.7 s	0.073707	72.7 s	0.117307
Gargoyle	2M	0%	9	102.5 s	0.025558	113.2 s	0.012143
Gargoyle	2M	0.1%	9	117.7 s	0.029181	126.5 s	0.026991
Gargoyle	2M	0.2%	9	166.2 s	0.047055	139.7 s	0.078216
Filigree	4M	0%	8	12.3 s	0.000688	74.7 s	0.000326
Filigree	4M	0%	9	75.3 s	0.000256	99.8 s	0.000109
Filigree	4M	0%	10	205.2 s	0.000117	153.6 s	0.000070
Livingstone Elephant	8M	0%	9	85.6 s	0.035024	327.9 s	0.0197755
Livingstone Elephant	8M	0%	10	204.3 s	0.018907	384.4 s	0.0059160

Table 3: *Statistics for the data sets used to assess the fitting accuracy of the proposed algorithm. The first columns show data set statistics and fitting parameters, the remaining columns show the fitting times and the average distances between the original mesh and the reconstructed mesh for Poisson surface reconstruction and our method. The average error has been computed at the mesh vertices using Metro [CRS98]. The reconstructed meshes are shown in the supplemental material.*

5.3. Reconstruction of time-varying data

Another interesting application for our hierarchical surface reconstruction is the reconstruction of time-varying data as proposed in [WSC06, SWG08], where the time at which a data point has been recorded is treated as fourth coordinate. We have implemented a CPU prototype for the 4D reconstruction which fits a 4D implicit function f to a time-space point cloud. Then we extract time slices from f and polygonize them using marching cubes. The reconstruction of the implicit function from 4.5 million 4D points at tree depth 8 took roughly 13 minutes and 2.2 GB peek memory using the proposed method. To obtain the "cut-out" look of Süßmuth et al. [SWG08], we restrict the isosurface extraction to areas where the confidence is larger than a given threshold. Results of the 4D reconstruction can be seen in Figure 13 and the accompanying video.

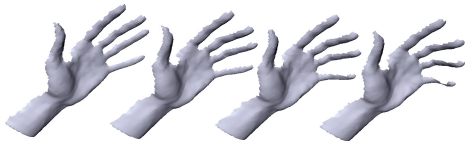

Figure 13: *Meshes reconstructed from the four-dimensional time-varying hand data set.*

6. Conclusion

We have introduced the concept of floating centers for scattered data approximation using radial basis functions. Our method optimizes the locations of the RBF centers such that the approximation error is minimized, thus generating better fits than conventional methods for RBF approximation. We presented a fast GPU based implementation for the floating centers optimization. The GPU implementation allows us to use our algorithm at little additional costs compared to a CPU implementation.

We have further presented a hierarchical framework

which is based on locally offsetting the approximating function. The hierarchical approximation is robust and computes a global distance field, thus it has applications in surface reconstruction and non-rigid surface registration.

In future, we plan to optimize not only the center position in the nonlinear optimization step but the shape parameters δ and possibly anisotropic basis functions as well. Further avenues for future work include the integration of sample confidence values into the fitting process and the realization of a streaming out-of-core implementation to reconstruct huge point clouds.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. We are very grateful to Yutaka Ohtake, Michael Kazhdan and Josiah Manson for providing their surface reconstruction algorithms. We would further like to thank Michael Kazhdan for providing the implementation for the unconstrained isosurface extraction on arbitrary octrees. The Squirrel data set is courtesy of Yutaka Ohtake. The Bee range scans and the hand sequence have been provided by Thibaut Weise. The Bunny data set is from the Stanford 3D scanning repository and the Neptune and the dancing children data sets are courtesy of Marco Attene from the Aim@Shape Project. The Galaad point cloud is provided courtesy of Laurent Saboret by the AIM@SHAPE Project.

References

[ACSTD07] ALLIEZ P., COHEN-STEINER D., TONG Y., DESBRUN M.: Voronoi-based variational reconstruction of unoriented point sets. In *Proc. SGP'07* (2007), pp. 39–48.

[Blo94] BLOOMENTHAL J.: An implicit surface polygonizer. *Graphics Gems IV* (1994), 324–349.

[CBC*01] CARR J. C., BEATSON R. K., CHERRIE J. B., MITCHELL T. J., FRIGHT W. R., MCCALLUM B. C., EVANS T. R.: Reconstruction and representation of 3D objects with radial basis functions. In *SIGGRAPH '01: ACM SIGGRAPH 2001 Papers* (2001), pp. 67–76.

[CG06] CAZALS F., GIESEN J.: Delaunay triangulation based surface reconstruction. *Effective Computational Geometry for Curves and Surfaces* (2006), 231–276.

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[Duc77] DUCHON J.: Splines minimizing rotation-invariant seminorms in Sobolev spaces. *Constructive Theory of Functions of Several Variables* (1977), 85–100.

[FH99] FRANKE R., HAGEN H.: Least squares surface approximation using multiquadrics and parametric domain distortion. *Comput. Aided Geom. Des. 16*, 3 (1999), 177–196.

[FI96] FLOATER M. S., ISKE A.: Multistep scattered data interpolation using compactly supported radial basis functions. *J. Comput. Applied Math. 73*, 1–3 (1996), 65–78.

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. In *SIGGRAPH '92: ACM SIGGRAPH 1992 Papers* (1992), pp. 71–78.

[Isk00] ISKE A.: *Optimal distribution of centers for radial basis function methods*. Tech. rep., TUM-M0004, TU München, 2000.

[JBL*06] JANG Y., BOTCHEN R. P., LAUSER A., EBERT D. S., GAITHER K. P., , ERTL T.: Enhancing the interactive visualization of procedurally encoded multifield data with ellipsoidal basis functions. *Computer Graphics Forum (Proc. Eurographics'06) 25*, 3 (2006), 587–596.

[JR07] JALBA A., ROERDINK J.: Efficient surface reconstruction using generalized coulomb potentials. *IEEE TVCG 13*, 6 (Nov.-Dec. 2007), 1512–1519.

[Kaz05] KAZHDAN M.: Reconstruction of solid models from oriented point sets. In *Proc. SGP'05* (2005), pp. 73–82.

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. In *Proc. SGP'06* (2006), pp. 61–70.

[KKDH07] KAZHDAN M., KLEIN A., DALAL K., HOPPE H.: Unconstrained isosurface extraction on arbitrary octrees. In *Proc. SGP'07* (2007), pp. 125–133.

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *SIGGRAPH '87: ACM SIGGRAPH 1987 Papers* (1987), pp. 163–169.

[LSP08] LI H., SUMNER R. W., PAULY M.: Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum (Proc. SGP'08) 27*, 5 (2008), 1421–1430.

[Mar63] MARQUARDT D. W.: An algorithm for least-squares estimation of nonlinear parameters. *J. of the Soc. for Industrial and Applied Mathematics 11*, 2 (1963), 431–441.

[MPS08] MANSON J., PETROVA G., SCHAEFER S.: Streaming surface reconstruction using wavelets. *Computer Graphics Forum (Proc. SGP'08) 27*, 5 (2008), 1411–1420.

[NVI09] NVIDIA CORPORATION: *NVIDIA CUDA Programming Guide Version 2.3*, January 2009.

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (2003), pp. 463–470.

[OBS03] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: A multi-scale approach to 3D scattered data interpolation with compactly supported basis functions. In *Proc. SMI'03* (2003), pp. 153–161.

[SAAY06] SAMOZINO M., ALEXA M., ALLIEZ P., YVINEC M.: Reconstruction with Voronoi centered radial basis functions. In *Proc. SGP'06* (2006), pp. 51–60.

[SWG08] SÜSSMUTH J., WINTER M., GREINER G.: Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum (Proc. SGP'08) 27*, 5 (2008), 1469–1476.

[TO99] TURK G., O'BRIEN J. F.: *Variational implicit surfaces*. Tech. rep., Georgia Institute of Technology, May 1999.

[VMG09] VUÇINI E., MÖLLER T., GRÖLLER M. E.: On visualization and reconstruction from non-uniform point sets using B-Splines. *Computer Graphics Forum (Proc. EuroVis'09) 28*, 3 (2009), 1007–1014.

[Wen95] WENDLAND H.: Piecewise polynomial, positive definite and compactly supported radial basis functions of minimal degree. *Advances in Computational Mathematics 4 4* (1995), 389–396.

[WSC06] WALDER C., SCHÖLKOPF B., CHAPELLE O.: Implicit surface modelling with a globally regularised basis of compact support. *Computer Graphics Forum (Proc. Eurographics'06) 25*, 3 (2006), 635–644.