Fast Surface Reconstruction Using the Level Set Method

Hong-Kai Zhao[*] Stanley Osher[†] Ronald Fedkiw[‡]

Abstract

In this paper we describe new formulations and develop fast algorithms for implicit surface reconstruction based on variational and partial differential equation (PDE) methods. In particular we use the level set method and fast sweeping and tagging methods to reconstruct surfaces from scattered data set. The data set might consist of points, curves and/or surface patches. A weighted minimal surface-like model is constructed and its variational level set formulation is implemented with optimal efficiency. The reconstructed surface is smoother than piecewise linear and has a natural scaling in the regularization that allows varying flexibility according to the local sampling density. As is usual with the level set method we can handle complicated topology and deformations, as well as noisy or highly non-uniform data sets easily. The method is based on a simple rectangular grid, although adaptive and triangular grids are also possible. Some consequences, such as hole filling capability, are demonstrated, as well as the viability and convergence of our new fast tagging algorithm.

Keywords: implicit surface, partial differential equations, variational formulation, convection, minimal surface.

1 Introduction

Surface reconstruction from unorganized data set is very challenging in three and higher dimensions. The problem is ill-posed, i.e, there is no unique solution. Furthermore the ordering or connectivity of data set and the topology of the real surface can be very complicated in three and higher dimensions. A desirable reconstruction procedure should be able to deal with complicated topology and geometry as well as noise and non-uniformity of the data to construct a surface that is a good approximation of the data set and has some smoothness (regularity). Moreover, the reconstructed surface should have a representation and data structure that is not only good for static rendering but also good for deformation, animation and other dynamic operation on surfaces. None of the present approaches possess all of these properties. In general there are two kinds of surface representations, explicit or implicit. Explicit surfaces prescribe the precise location of a surface while implicit surfaces represent a surface as a particular isocontour of a scalar function. Popular explicit representations include parametric surfaces and triangulated surfaces. For examples, for parametric surfaces such as NURBS [26, 27], the reconstructed surface is smooth and the data set can be non-uniform. However this requires one to parametrize the data set in a nice way such that the reconstructed surface is a graph in the parameter space. The parametrization and patching can be very difficult for surface reconstruction from an arbitrary data set in three and higher dimensions. Also noise in the data set is difficult to deal with. Another popular approach in computer graphics is to reconstruct a triangulated surfaces using Delaunay triangulations and Voronoi diagrams. The reconstructed surface is typically a subset of the faces of the Delaunay triangulations. A lot of work has been done along these lines [3, 4, 7, 16, 17] and efficient algorithms are available to compute Delaunay triangulations and Voronoi diagrams. Although this approach is more versatile in that it can deal with more general data sets, the constructed surface is only piecewise linear and it is difficult to handle non-uniform and noisy data. Furthermore the tracking of large deformations and topological changes is usually quite difficult using explicit surfaces.

Recently, implicit surfaces or volumetric representations have attracted a lot of attention. The traditional approach [6, 20, 30] uses a combination of smooth basis functions (primitives), such as blobs, to find a scalar function such that all data points are close to an isocontour of that scalar function. This isocontour represents the constructed implicit surface. However computation costs are very high for large data sets, since the construction is global which results in solving a large linear system, i.e. the basis functions are coupled together and a single data point change can result in globally different coefficients. This makes human interaction, incremental updates and deformation difficult. The second approach uses the data set to define a signed distance function on rectangular grids and denotes the zero isocontour of the signed distance function as the reconstructed implicit surface [5, 8, 19]. The construction of the signed distance function uses a discrete approach and needs an estimation of local tangent planes or normals for the orientation, i.e. a distinction needs to be made between inside and outside. Similar ideas have been applied to shape reconstruction from range data and image fusion [13, 18] where partial connections are available on each piece of data. Special "fusion" or "tiling" algorithm is need on overlapping patches. In [28] an interesting volume spline is used for reconstruction of implicit surfaces. But the construction depends on the choice of a "carrier" solid and can deal with data sets of moderate size. The main advantages of implicit surfaces include topological flexibility, a simple data structure, depth/volumetric information and memory storage efficiency. Using the signed distance representation, many surface operations such as Boolean operations, ray tracing and offset become quite simple [24, 31]. Efficient algorithms, see e.g. [21, 32], are available to turn an implicit surface into a triangulated surface. In [15] implicit surfaces are used for animation and the level set method is used for surface reconstruction from range data in [10]. In fact the level set method [23] provides a general framework for the deformation of implicit surfaces according to arbitrary physical and/or geometric rules.

We approach the fundamental problem of surface reconstruction on the continuous level by constructing continuous models using differential geometry and partial differential equations. We also develop efficient and robust numerical algorithms for our continuous formulations. Moreover we combine the level set method and implicit surfaces to provide a general framework for surface modeling, analysis, deformation and many other applications. In our previous work [34] we proposed a new "weighted" minimal surface model based on variational formulations and PDE methods. Only the unsigned distance function to the data set was used in our formulation. Our reconstructed surface is smoother than piecewise linear. In ad-

[*]Department of Mathematics, University of California, Irvine, CA 92697-3875, Research supported by NSF DMS 9706566. zhao@math.uci.edu

[†]Department of Mathematics, University of California, Los Angeles, CA 90095-1555, Research supported by ONR N00014-97-1-0027 and NSF DMS 9706827. sjo@math.ucla.edu

[‡]Stanford University, Gates Computer Science Bldg., Stanford, CA 94305-9020, Research supported by ONR N00014-97-1-0027. fedkiw@cs.stanford.edu

dition, in our formulation there is a regularization that is adaptive to the local sampling density which can keep sharp features if a local sampling condition is satisfied. The formulation handles noisy as well as non-uniform data and works in any number of dimensions. We use the level set method as the numerical technique to deform the implicit surface continuously following the gradient descent of the energy functional for the final reconstruction. Instead of tracking a parametrized explicit surface we solve a PDE on a simple rectangular grid and handle topological changes easily. In this paper we develop a simple physically motivated convection model and a fast tagging algorithm to construct a good initial approximation for our minimal surface reconstruction. This will speed up our previous reconstruction by an order of magnitude.

In the next section we briefly review the variational formulation for the weighted minimal surface model in introduced in [34]. A physically motivated simple convection model is developed in section 3. In section 4 we introduce the level set method for our problems. We explain the details of the numerical algorithms in section 5 and show results in section 6.

2 A Weighted Minimal Surface Model

Let $\mathcal{S}$ denote a general data set which can include data points, curves or pieces of surfaces. Define $d(\boldsymbol{x}) = dist(\boldsymbol{x}, \mathcal{S})$ to be the distance function to $\mathcal{S}$. (We shall use bold faced characters to denote vectors.) In [34] the following surface energy is defined for the variational formulation:

$$E(\boldsymbol{\Gamma}) = \left[\int_{\boldsymbol{\Gamma}} d^p(\boldsymbol{x})ds\right]^{\frac{1}{p}}. \quad 1 \le p \le \infty, \tag{1}$$

where $\boldsymbol{\Gamma}$ is an arbitrary surface and ds is the surface area. The energy functional is independent of parametrization and is invariant under rotation and translation. When $p = \infty$, $E(\boldsymbol{\Gamma})$ is the value of the distance of the point $\boldsymbol{x}$ on $\boldsymbol{\Gamma}$ furthest from $\mathcal{S}$. For $p < \infty$, The surface energy $E(\boldsymbol{\Gamma})$ is equivalent to $\int_{\boldsymbol{\Gamma}} d^p(\boldsymbol{x})ds$, the surface area weighted by some power of the distance function. We take the local minimizer of our energy functional, which mimics a weighted minimal surface or an elastic membrane attached to the data set, to be the reconstructed surface.

As derived in [34] the gradient flow of the energy functional (1) is

$$\frac{d\boldsymbol{\Gamma}}{dt} = -\left[\int_{\boldsymbol{\Gamma}} d^p(\boldsymbol{x})ds\right]^{\frac{1}{p}-1} d^{p-1}(\boldsymbol{x})\left[\nabla d(\boldsymbol{x})\cdot\boldsymbol{n} + \frac{1}{p}d(\boldsymbol{x})\kappa\right]\boldsymbol{n}, \tag{2}$$

and the minimizer or steady state solution of the gradient flow satisfies the Euler-Lagrange equation

$$d^{p-1}(\boldsymbol{x})\left[\nabla d(\boldsymbol{x})\cdot\boldsymbol{n} + \frac{1}{p}d(\boldsymbol{x})\kappa\right] = 0, \tag{3}$$

where $\boldsymbol{n}$ is the unit outward normal and κ is the mean curvature. We see a balance between the attraction $\nabla d(\boldsymbol{x}) \cdot \boldsymbol{n}$ and the surface tension $d(\boldsymbol{x})\kappa$ in the equations above. Moreover the nonlinear regularization due to surface tension has a desirable scaling $d(\boldsymbol{x})$. Thus the reconstructed surface is more flexible in the region where sampling density is high and is more rigid in the region where the sampling density is low. In the steady state equation(3) above, since $\nabla d \cdot \boldsymbol{n} \le 1$, we have a local sampling density condition similar to the one proposed in [3], which says sampling densities should resolve fine features locally. To construct the minimal surface we used a continuous deformation in [34]. We start with an initial surface that encloses all data and follow the gradient flow (2). The parameter p affects the flexibility of the membrane to some extent. When $p = 1$, the surface energy defined in (1) has the dimension of volume and the gradient flow (2) is scale invariant i.e., dimensionless. In practice we find that $p = 1$ or 2 (similar to a least squares formulation) are good choices. Some more details can be found in [34].

In two dimensions, it was shown in [34] that a polygon which connects adjacent points by straight lines is a local minimum. This result shows a connection between the variational formulation and previous approaches. On the other hand this result is not surprising since a minimal surface passing through two points is a straight line in two dimensions. However in three dimensions the situation becomes much more interesting. The reconstructed minimal surface has no edges and is smoother than a polyhedron.

Remark: The formulation here is similar to active contour models for image segmentation in [12, 11] and minimal surface model in [33]. However the application, motivation and working mechanism are quite different. In image segmentation, the final curves or surfaces are wrapped along some edges, i.e., some continuous high contrast contours on the grid and the contrast is already defined on every grid point. In our application, we have arbitrary discrete points and none of the distance contours to the data points is the final surface.

3 The Convection Model

The evolution equation (2) involves the mean curvature of the surface and is a nonlinear parabolic equation. A time implicit scheme is not currently available. A stable time explicit scheme requires a restrictive time step size, $\Delta t = O(h^2)$, where h is the spatial grid cell size. Thus it is very desirable to have an efficient algorithm to find a good approximation before we start the gradient flow for the minimal surface. We propose the following physically motivated convection model for this purpose.

The convection of a flexible surface $\boldsymbol{\Gamma}$ in a velocity field $\boldsymbol{v}(\boldsymbol{x})$ is described by the differential equation

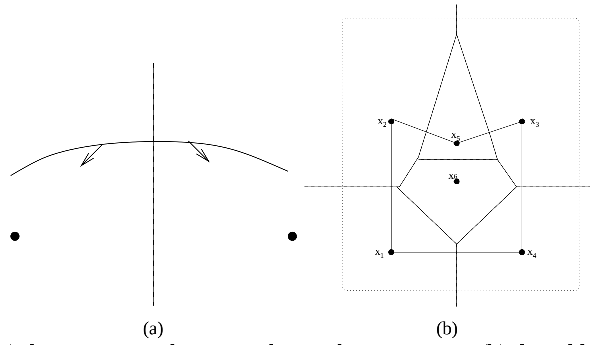
$$\frac{d\boldsymbol{\Gamma}(t)}{dt} = \boldsymbol{v}(\boldsymbol{\Gamma}(t)).$$

If the velocity field is created by a potential field $\mathcal{F}$, then $\boldsymbol{v} = -\nabla\mathcal{F}$. In our convection model the potential field is the distance function $d(\boldsymbol{x})$ to the data set $\mathcal{S}$. This leads to the convection equation

$$\frac{d\boldsymbol{\Gamma}(t)}{dt} = -\nabla d(\boldsymbol{x}). \tag{4}$$

For example, if the data set contains a single point $\boldsymbol{x}_0$, the potential field is $d(\boldsymbol{x})=|\boldsymbol{x}-\boldsymbol{x}_0|$ and the velocity field is $\boldsymbol{v}(\boldsymbol{x}) = -\nabla d(\boldsymbol{x}) = -\frac{\boldsymbol{x}-\boldsymbol{x}_0}{|\boldsymbol{x}-\boldsymbol{x}_0|}$, a unit vector pointing towards $\boldsymbol{x}_0$. Any particle in this potential field will be attracted toward $\boldsymbol{x}_0$ along a straight line with unit speed. For a general data set $\mathcal{S}$, a particle will be attracted to its closest point in $\mathcal{S}$ unless the particle is located an equal distance from two or more data points. The set of equal distance points has measure zero. Similarly, points on a curve or a surface, except those equal distance points, are attracted by their closest points in the data set (see Fig. 1(a)). The ambiguity at those equal distance points is resolved by adding a small surface tension force which automatically exists as numerical viscosity in our finite difference schemes. Those equal distance points on the curve or surface are dragged by their neighbors and the whole curve or surface is attracted to the data set until it reaches a local equilibrium, which is a polygon or polyhedron whose vertices belong to the data set as the viscosity tends to zero (see Fig.1(b)).

Here are some properties of this simple convection model: (1) the normal velocity of the curve or the surface is less than or equal to 1, (2) each point of the curve or surface is attracted by its closest point in the data set.



(a) the attraction of a piece of curve by two points, (b) dotted line is the initial curve, solid line is the final curve, dashed line is the Voronoi diagram.

Figure 1:

Figure 1(b) is an illustration of the convection of a curve. The initial curve (the dotted rectangle) feels the attraction of $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ and closes in. Then it begins to feel $\boldsymbol{x}_5$. The final shape is a pentagon that goes through $\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4$ and $\boldsymbol{x}_5$ while $\boldsymbol{x}_6$ is screened out.

Since the convection equation is a first order linear differential equation, we can solve it using a time step $\Delta t = O(h)$ leading to significant computational savings over typical parabolic $\Delta t = O(h^2)$ time step restrictions. The convection model by itself very often results in a good surface reconstruction. In section 5 we will construct a very fast tagging algorithm that finds a crude approximation of the local equilibrium solution for our convection model.

4 The Level Set Formulation

In general we do not have any á priori knowledge about the topology of the shape to be reconstructed. Topological changes may occur during the continuous deformation process. This makes explicit tracking, which requires consistent parametrization, almost impossible to implement. Here we use the level set method as a powerful numerical technique for the deformation of implicit surfaces. Although implicit surfaces have been used in computer graphics for quite a while, they were mostly used for static modeling and were based on discrete formulations [6]. The *level set method* is based on a continuous formulation using PDEs and allows one to deform an implicit surface, which is usually the zero isocontour of a scalar (level set) function, according to various laws of motion depending on geometry, external forces, or a desired energy minimization. In numerical computations, instead of explicitly tracking a moving surface we implicitly capture it by solving a PDE for the level set function on rectangular grids. The data structure is extremely simple and topological changes are handled easily. The level set formulation works in any number of dimensions and the computation can easily be restricted to a narrow band near the zero level set, see e.g. [1, 25]. We can locate or render the moving surface easily by interpolating the zero isosurface of the level set function. The level set method was originally introduced by Osher and Sethian in [23] to capture moving interfaces and has been used quite successfully in moving interface and free boundary problems as well as in image processing, image segmentation and elsewhere. See [22] for a comprehensive review.

Two key steps for the level set method are: (1) Embed the surface: we represent a surface $\boldsymbol{\Gamma}$ as the zero isocontour of a scalar (level set) function $\phi(\boldsymbol{x})$, i.e. $\boldsymbol{\Gamma} = \{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$. Geometric properties of the surface $\boldsymbol{\Gamma}$ can be easily computed using ϕ. (2) Embed the motion: we derive the time evolution PDE for the level set function such that the zero level set has the same motion law as the moving surface, i.e., $\boldsymbol{\Gamma}(t) = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t) = 0\}$,

$$\frac{d\phi(\boldsymbol{\Gamma}(t), t)}{dt} = \phi_t + \frac{d\boldsymbol{\Gamma}(t)}{dt} \cdot \nabla\phi = 0, \tag{5}$$

where we replace $\frac{d\boldsymbol{\Gamma}(t)}{dt}$ with the velocity of $\boldsymbol{x}$ on $\boldsymbol{\Gamma} = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t) = 0\}$.

For geometric motions, i.e. where the motion law (velocity) depends only on the geometry of the moving surface, the most natural way to define $\boldsymbol{v}$ is to apply the same motion law for all level sets of the level set function, which will result in a morphological PDE [2]. For example, the gradient flow (2) is a geometric motion and we use $p = 1$ for simplicity. After we extend the geometric motion to all level sets, the gradient flow in level set formulation becomes

$$\frac{\partial\phi}{\partial t} = |\nabla\phi|\nabla\cdot\left[d\frac{\nabla\phi}{|\nabla\phi|}\right] = |\nabla\phi|\left[\nabla d\cdot\frac{\nabla\phi}{|\nabla\phi|} + d\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}\right], \tag{6}$$

For the convection model (4), since the velocity field $-\nabla d(\boldsymbol{x})$ is defined everywhere, we can naturally extend the convection to all level sets of $\phi(\boldsymbol{x}, t)$ to obtain

$$\frac{\partial\phi}{\partial t} = \nabla d(\boldsymbol{x}) \cdot \nabla\phi. \tag{7}$$

Although all level set functions are equally good theoretically, in practice the signed distance function is preferred to avoid stiffness and inaccuracy in numerical computations. However even if we start with a signed distance function the level set function will generally not remain a signed distance function. As an example, in the convection model all level sets are attracted to the data set simultaneously and they become more and more packed together. We need a procedure to force them apart while keeping the zero level set intact. We use a numerical procedure called reinitialization, see e.g. [25, 29], to redistance the level set function locally without interfering with the motion of the zero level set. As a result the implicit surface is a signed distance function after the deformation procedure stops.

5 Numerical Implementation

There are three key numerical ingredients in our implicit surface reconstruction. First, we need a fast algorithm to compute the distance function to an arbitrary data set on rectangular grids. Second, we need to find a good initial surface for our gradient flow. Third, we have to solve time dependent PDEs for the level set function.

5.1 Computing the distance function

The distance function $d(\boldsymbol{x})$ to an arbitrary data set $\mathcal{S}$ solves the following Eikonal equation:

$$|\nabla d(\boldsymbol{x})| = 1, \qquad d(\boldsymbol{x}) = 0,\ \boldsymbol{x} \in \mathcal{S}. \tag{8}$$

From the PDE point of view, the characteristics of this Eikonal equation are straight lines which radiate from the data set. This reveals the causality property for the solution of the PDE, i.e., the information propagates along straight lines from the data set, and the solution at a grid point should be determined only by its neighboring grid points that have smaller distance values. We use an algorithm [9, 34] that combines upwind differencing with Gauss-Seidel iterations of different sweeping order to solve (8) on rectangular grids.

From numerical experiments it seems that the total number of iterations is independent of mesh size, i.e. the complexity is $O(M+N)$ for N grid points and M data points. The differences between our algorithm and Danielsson's distance mapping algorithm [14] are (1) our data points are not grid points and that is why we have a complexity $O(M+N)$, (2) our algorithm can be applied to more general equations where $d(\boldsymbol{x})$ is not a distance, e.g., the right hand side of 8 can be an arbitrary function in $\boldsymbol{x}$. Our fast algorithm is versatile, efficient and will be used in later stages of the surface reconstruction.

5.2 Finding a good initial guess

We can use an arbitrary initial surface that contains the data set such as a rectangular bounding box, since we do not have to assume any á priori knowledge for the topology of the reconstructed surface. However, a good initial surface is important for the efficiency of our PDE based method. On a rectangular grid, we view an implicit surface as an interface with some regularity that separates the exterior grid points from the interior grid points. In other words, volumetric rendering requires identifying all exterior (interior) grid points correctly. Based on this idea, we propose a novel, extremely efficient tagging algorithm that tries to identify as many correct exterior grid points as possible and hence provide a good initial implicit surface. As always, we start from any initial exterior region that is a subset of the true exterior region.

All grid points that are not in the initial exterior region are labeled as interior points. Those interior grid points that have at least one exterior neighbor are labeled as temporary boundary points. Now we use the following procedure to march the temporary boundary inward toward the data set. We put all the temporary boundary points in a heapsort binary tree structure sorting according to distance values. Take the temporary boundary point that has the largest distance (which is on the heap top) and check to see if it has an interior neighbor that has a larger or equal distance value. If it does not have such an interior neighbor, turn this temporary boundary point into an exterior point, take this point out of the heap, add all this point's interior neighbors into the heap and re-sort according to distance values. If it does have such an interior neighbor, we turn this temporary boundary point into a final boundary point, take it out of the heap and re-sort the heap. None of its neighbors are added to the heap. We repeat this procedure on the temporary boundary points until the the maximum distance of the temporary boundary points is smaller than some tolerance, e.g. the size of a grid cell, which means all the temporary boundary points in the heap are close enough to the data set. Finally, we turn these temporary boundary points into the final set of boundary points and our tagging procedure is finished. Now we have the final sets of interior, exterior and boundary points. Since we visit each interior grid point at most once, the procedure will be completed in no more than $O(N \log N)$ operations, where $\log N$ comes from the heap sort algorithm. Moreover, since the maximum distance for the boundary heap is strictly decreasing, we can prove that those interior points which have a distance no smaller than the maximum distance of the temporary boundary heap at any time will remain as interior points, i.e. there is a non-empty interior region when the tagging algorithm is finished. We can also show that at least one of the final boundary points is within the tolerance distance to the data set. Similar tagging algorithms can also be applied to finding interior regions and disconnected components of the final shape.

Figure 2 illustrates how our fast tagging algorithm works. Starting from an arbitrary exterior region that is a subset of the final exterior region, the furthest point on the temporary boundary is tangent to a distance contour and does not have an interior point that is farther away. The furthest point will be tagged as an exterior point and the boundary will move inward at that point. Now another point on the temporary boundary becomes the furthest point and hence the whole temporary boundary moves inward. After a while the temporary boundary is close to a distance contour and moves closer and closer to the data set following the distance contours until the distance contours begin to break into spheres (circles in the 2D figure) around data points. We now see that the temporary boundary point at the breaking point of the distance contour, which is equally distant from distinct data points, will have neighboring interior points that have a larger distance. So this temporary boundary point will be tagged as a final boundary point by our procedure and the temporary boundary will stop moving inward at this breaking point. The temporary boundary starts deviating from the distance contours and continues moving closer to the data set until all temporary boundary points either have been tagged as final boundary points or are close to the data points. The final boundary is approximately a a polyhedron (polygon in 2D) with vertices belonging to the data set.

This general tagging algorithm can incorporate human interaction easily by putting any new exterior point(s) or region(s) into our tagged exterior region at any stage in our tagging algorithm. After the tagging algorithm is finished we again use the fast distance algorithm to compute a signed distance to the tagged final boundary.

The marching method (outlined above) requires an initial guess for the exterior region. This can either be the bounding box of our computational rectangular domain or an outer contour of the distance function, $d(\boldsymbol{x}) = \epsilon$. An outer contour of the distance function can be found by starting with the outer boundary of our rectangular box, and expanding the exterior region by repeatedly tagging those grid points which are neighbors of the expanding exterior boundary and have a distance larger than ϵ as exterior points. All remaining untagged grid points are interior points. When the tagging algorithm is finished the boundary of the exterior region is approximately the outer contour of $d(\boldsymbol{x}) = \epsilon$ or roughly an ϵ offset of the real shape. When using this $d(\boldsymbol{x}) = \epsilon$ method, first proposed in [34], one needs to exercise caution in choosing ϵ. For example, if ϵ is too small, we will have isolated spheres surrounding data points. If the sampling density of the data points does not vary too much and is fine enough to resolve all features, then we can find an appropriate ϵ and get a very good initial surface with $O(N+M)$ operations. For non-uniform data points the intersection of a bounding box and a distance contour with moderate ϵ, which is a simple Boolean operation, often gives a good initial surface.

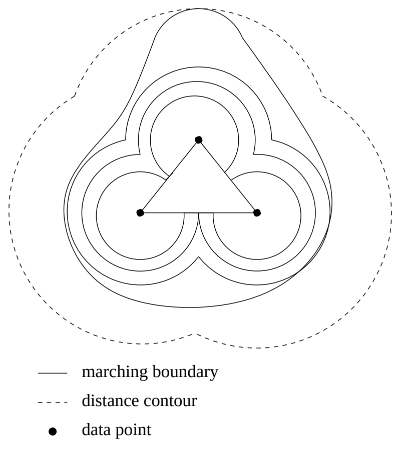


Figure 2:

5.3 Solving the partial differential equation.

After we find the distance function $d(\boldsymbol{x})$ and a good initial implicit surface using the above algorithms, we can start the continuous deformation following either the gradient flow (2) or the convection (4) using the corresponding level set formulation (6) or (7). Our numerical implementations are based on standard algorithms for the level set method. Details can be found in, for example, [25, 33, 34]. The convection model is simple but the reconstructed surface is close to a piecewise linear approximation. In contrast, the energy minimizing gradient flow, which contains a weighted curvature regularization effect, is more computationally expensive but reconstructs a smooth weighted minimal surface. In particular, the gradient flow can be used as a smoothing process for implicit surfaces. In most of our applications, about one hundred time steps in total are enough for our continuous deformation. Since we use a reinitialization procedure during the deformation, we finish with a signed distance function for the reconstructed implicit surface.

5.4 Multiresolution

There are two scales in our surface reconstruction. One is the resolution of the data set. The other is the resolution of the grid. The computational cost generally depends mainly on the grid size. To achieve the best results those two resolutions should be comparable. However our grid resolution can be independent of the sampling density. For example, we can use a low resolution grid when there is noise and redundancy in the data set or when memory and speed are important. From our numerical results, see e.g., figure 9(c) our reconstruction is quite smooth even on a very low resolution grid. We can also use a multiresolution algorithm, i.e., reconstruct the surface first on coarser grids and interpolate the result to a finer resolution grid for further refinement in an hierarchical way.

5.5 Efficient storage

To store or render an implicit surface, we only need to record the values and locations (indices) of those grid points that are next to the surface, i.e., those grid points that have a different sign from at least one of their neighbors. These grid points form a thin grid shell surrounding the implicit surface. No connectivity or other information needs to be stored. We reduce the file size by at least an order of magnitude by using this method. Moreover we can easily reconstruct the signed distance function in $O(N)$ operations for the implicit surface using the following procedure. (1) Use the fast distance finding algorithm to find the distance function using the absolute value of the stored grid shell as an initial condition. (2) Use a tagging algorithm, similar to the one used above to find exterior points outside a distance contour, to identify all exterior points and interior points separated by the stored grid shell and turn the computed distance into the signed distance. For example, if we store the signed distance function for our reconstructed Happy Buddha from almost half a million points on a $146 \times 350 \times 146$ grid in binary form, the file size is about 30MB. If we use the above efficient way of storage the file size is reduced to 2.5MB without using any compression procedure and we can reconstruct the signed distance function in 1 minute using the above algorithm .

6 Results

In this section we present numerical examples that illustrate the efficiency and quality of our surface construction. In particular we show (1) the level set method handles surface deformation and topological change easily, (2) our fast tagging algorithm constructs a good initial guess very quickly, (3) how smooth the reconstructed surfaces are, (4) our algorithm works with non-uniform, noisy or damaged data, and (5) multiresolution works in our formulation. All calculations were done with dual Pentium III, 600Mhz processors. Data points for the drill, dragon and the Buddha

Model	Data points	Grid size	CPU (initial)	CPU (total)
Rat brain	1506	80x77x79	.12	3
Hand	327323	200x141x71	.5	10
Drill	1961	24x250x32	0.1	2
Dragon	437645	300x212x136	4	77
Dragon	100250	300x212x136	3	66
Buddha	543652	146x350x146	3	68
Buddha	543652	63x150x64	.3	7

Figure 3: timing table

were obtained from www-graphics.stanford.edu/data/3Dscanrep and data points for the hand skeleton was obtained from www.cc.gatech.edu/projects/large_models. Only locations of the data points are used in our reconstructions. Timings, number of data points and grid size are shown in table 3. CPU time is measured in minutes. CPU (initial) is the time for the initial reconstruction using the distance contour and the fast tagging algorithm. CPU (total) is the total time used for the reconstruction. Since our PDE based algorithms are iterative procedures, different convergence criterion will give different convergence times.

Figure 4 shows data points for a torus, a few curves (longitudes and latitudes) on a sphere, data points from MRI slices for a rat brain. Figure 5 shows the final surface reconstruction from the above data. We see that the hole in the torus is filled nicely with a minimal surface. For the sphere sphere reconstruction we only provide the unsigned distance function to the curves which can be viewed as an extreme case of non-uniform data. Since the data sets in these three cases are quite non-uniform, we use either a bounding box as the initial surface (in the case of torus and sphere) or the intersection of a bounding box and an outer distance contour with relatively large ϵ. ($\epsilon = 12h$ in the case of the rat brain.)

For data sets that are fairly uniform, such as the drill, hand skeleton, the dragon and the Buddha, we start with an outer distance contour and use the fast tagging algorithm to get an initial reconstruction. The initial reconstruction is extremely fast, as we can see from table 3. After the initial reconstruction, we first use the convection model and then use the gradient flow to finish the final reconstruction. In our reconstruction, the grid resolution is much lower than the data samples and yet we get final results that are comparable to the reconstructions shown at those websites above.

Figure 6 shows the reconstruction of a hand skeleton. Figure 7 is the reconstruction of a drill. It is a quite challenging example for most methods for surface reconstruction as is shown in [13]. Next we show the reconstruction of a dragon on a $300 \times 212 \times 136$ grid using a high resolution data in fig.8(a) and a much lower resolution data in fig.8(b). We can barely see the difference. Figure 9 shows the reconstruction of the Happy Buddha. Figure 9(a) shows the initial reconstruction. We start with an outer distance contour, $d = 3h$, initially and use the fast tagging algorithm. It takes 3 minutes for half a million points on a $146 \times 350 \times 146$ grid. Figure 9(b) is the final reconstruction. Figure 9(c) is the reconstruction on a very coarse $63 \times 150 \times 64$ grid using the same amount of data points. It takes only 7 minutes and the result is quite good.

7 Conclusions

We present a variational and PDE based formulation for surface reconstruction from unorganized data. Our formulation only depends on the (unsigned) distance function to the data and the final reconstruction is smoother than piecewise linear. We use the level set method as a numerical tool to deform and construct implicit surfaces on fixed rectangular grids. We use fast sweeping algorithms for computing the distance function and fast tagging algorithms for initial construction. Our method works for complicated topology and non uniform or noisy data.

References

[1] D. Adalsteinsson and J.A. Sethian. A fast level set method for propagating interfaces. *J. Comp. Phys.*, 118(2):269–277, 1995.

[2] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and fundamental equations of image processing. *Arch. Rat. Mechanics*, 123:199–257, 1993.

[3] N. Amenta, M. Bern, and D. Eppstein. The crust and the β-skeleton: combinatorial curve reconstruction. *Graphical Models and Image Processing*, 60/2(2):125–135, 1998.

[4] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH'98*, pages 415–421, 1998.

[5] C. Bajaj, F. Bernardini, and G. Xu. Automatic reconstruction of surfaces and scalar fields from 3d scans. *SIGGRAPH'95 Proceedings*, pages 193–198, July 1995.

[6] J. Bloomenthal, C. Bajaj, J. Blinn, M.-P Cani-Gascuel, A. Rockwood, B. Wyvill, and G. Wyvill. *Introduction to implicit surfaces*. Morgan Kaufman, Inc., San Francisco, 1997.

[7] J.D. Boissonnat. Geometric structures for three dimensional shape reconstruction. *ACM Trans. Graphics 3*, pages 266–286, 1984.

[8] J.D. Boissonnat and F. Cazals. Smooth shape reconstruction via natural neighbor interpolation of distance functions. *ACM Symposium on Computational Geometry*, 2000.

[9] M. Boué and P. Dupuis. Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. *SIAM J. Numer. Anal.*, 36(3):667–695, 1999.

[10] D.E. Breen, S. Mauch, and R.T. Whitaker. 3d scan conversion of csg models into distance. *1998 Volume Visualization Symposium*, pages 7–14, October 1998.

[11] V. Caselles, R. Kimmel, G. Sapiro, and C. Sbert. Minimal surfaces based object segmentation. *IEEE Trans. Pattern Analysis Machine Intelligence*, 19:4:394–398, 1997.

[12] L.D. Cohen and I. Cohen. Finite-element methods for active contour models and ballons for 2-d and 3-d images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(11):1131–1147, November 1993.

[13] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH'96 Proceedings*, pages 303–312, 1996.

[14] P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[15] M. Desbrun and M.P. Cani-Gasceul. Active implicit surfaces. *Graphics Interface*, 1998.

[16] H. Edelsbrunner. Shape reconstruction with Delaunay complex. In *Proc. of LATIN'98: Theoretical Informatics*, volume 1380 of *Lecture Notes in Computer Science*, pages 119–132. Springer-Verlag, 1998.

[17] H. Edelsbrunner and E. P. Mücke. Three dimensional α shapes. *ACM Trans. Graphics 13*, pages 43–72, 1994.

[18] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface - based geometric fusion. *Comput. Vision and Image Understanding*, 69:273–291, 1998.

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH'92 Proceedings*, pages 71–78, 1992.

[20] S. Muraki. Volumetric shape description of range data using "blobby model". In *Computer Graphics (Proc. SIGGRAPH)*, volume 25, pages 227–235, July 1991.

[21] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.

[22] S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. *J. Comp. Phys.*, 169(2), 2001.

[23] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed, algorithms based on a Hamilton-Jacobi formulation. *J. Comp. Phys.*, 79:12–49, 1988.

[24] A. Pasko, V. Adzhiev, A. Sourin, and V. Savchenko. Function representation in geometric modeling: concepts, implementation and applications. *The Visual Computer*, 11(8):429–446, 1995.

[25] D. Peng, B. Merriman, S. Osher, H.K. Zhao, and M. Kang. A PDE based fast local level set method. *J. Comp. Phys.*, 155:410–438, 1999.

[26] L. Piegl and W. Tiller. *The NURBS book*. Berlin, Germany: Springer-Verlag, 2nd edition edition, 1996.

[27] D.F. Rogers. *An Introduction to NURBS*. Morgan Kaufmann, 2000.

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

[29] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flows. *J. Comp. Phys.*, 119:146–159, 1994.

[30] G. Turk and J. ÒBrien. Shape transformation using variational implicit functions. *SIGGRAPH99*, pages 335–342, August 1999.

[31] B. Wyvill, A. Guy, and E. Galin. Extending the CSG tree, warping, blending and boolean operations in an implicit surface modeling system. *Computer Graphics Forum*, 18(2):149–158, June 1999.

[32] B. Wyvill, C. McPheeters, and G. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[33] H.K. Zhao, T.F. Chan, B. Merriman, and S.Osher. A variational level set approach to multiphase motion. *J. Comp. Phys.*, 127:179–195, 1996.

[34] H.K. Zhao, S. Osher, B. Merriman, and M. Kang. Implicit and non-parametric shape reconstruction from unorganized points using variational level set method. *Computer Vision and Image Understanding*, 80(3):295–319, 2000.

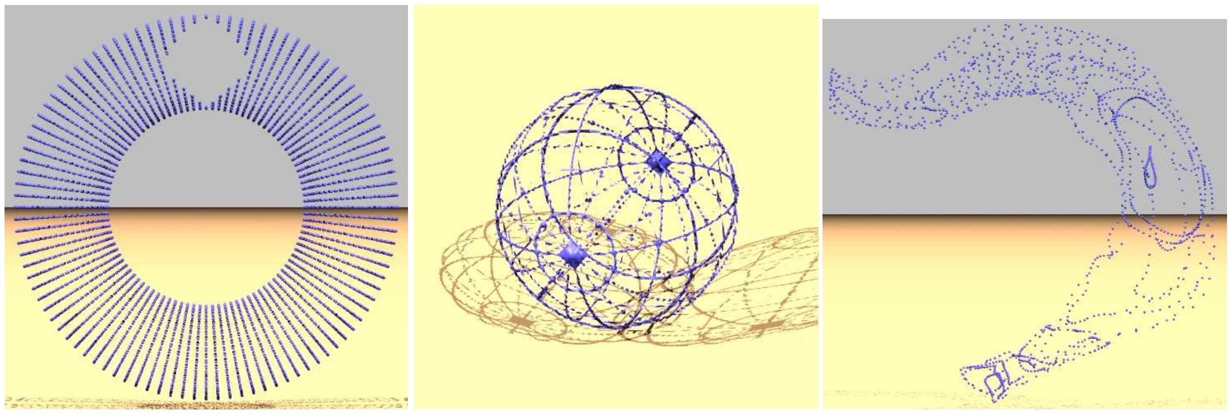
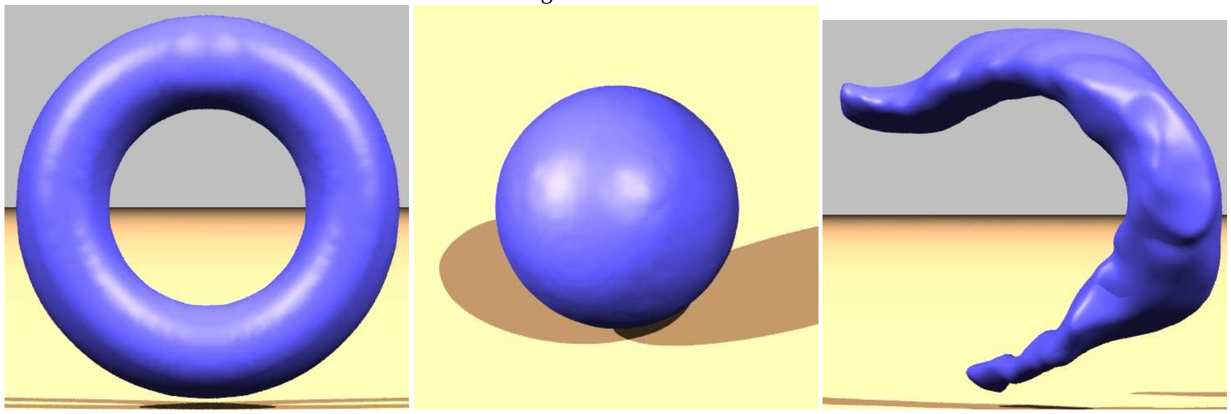


Figure 4: initial data

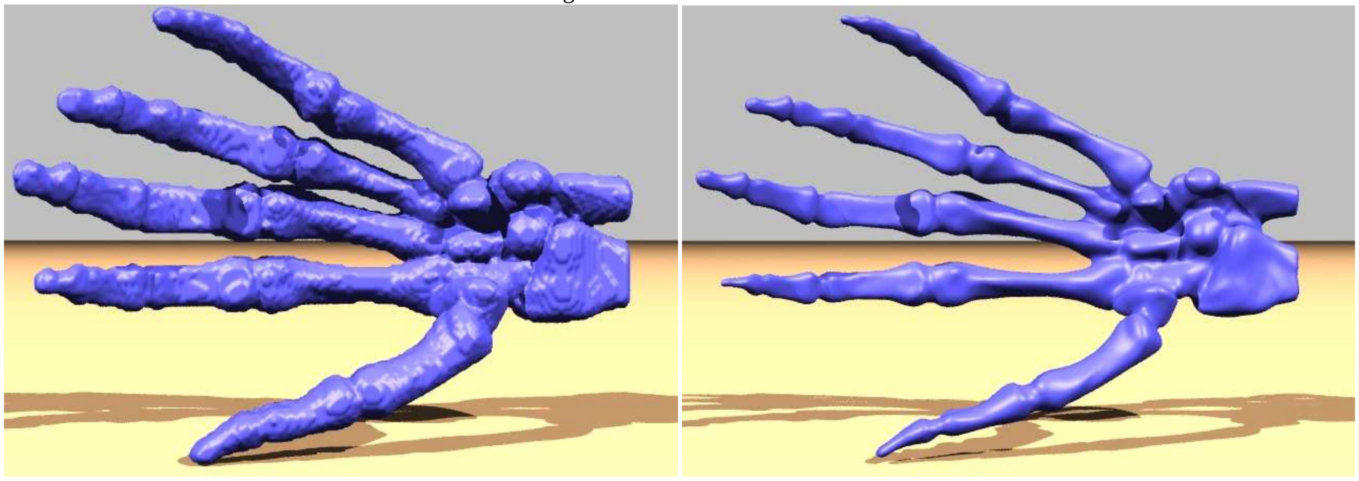


hole filling of a torus

reconstruction of a sphere from curves

reconstruction of a rat brain from MRI slices

Figure 5: final reconstruction



(a) initial reconstruction

(b) final reconstruction

Figure 6: reconstruction of a hand skeleton

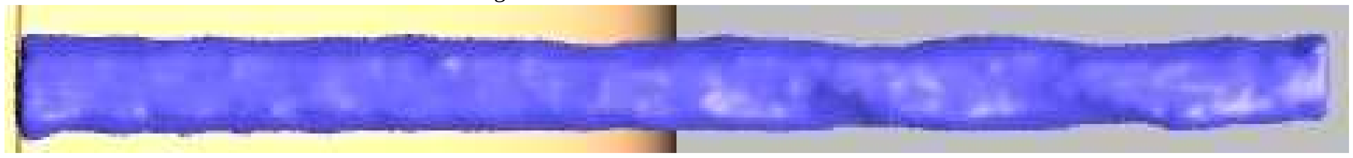
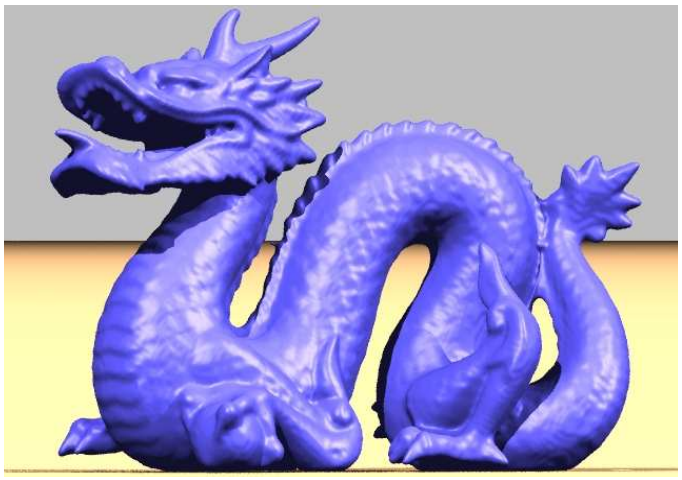


Figure 7: reconstruction of a drill


(c) final reconstruction

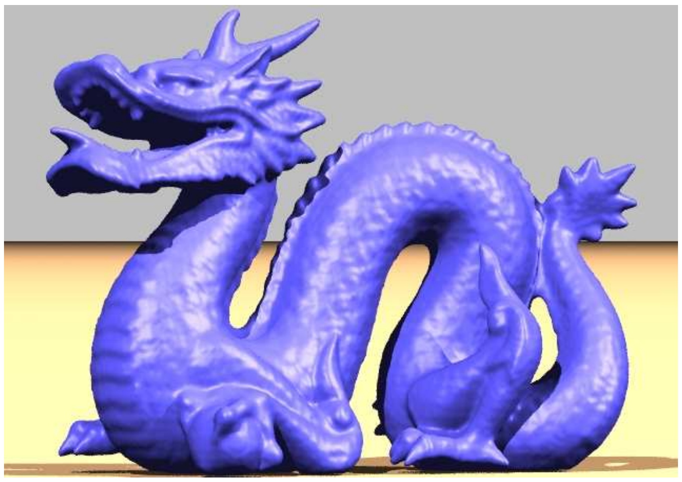
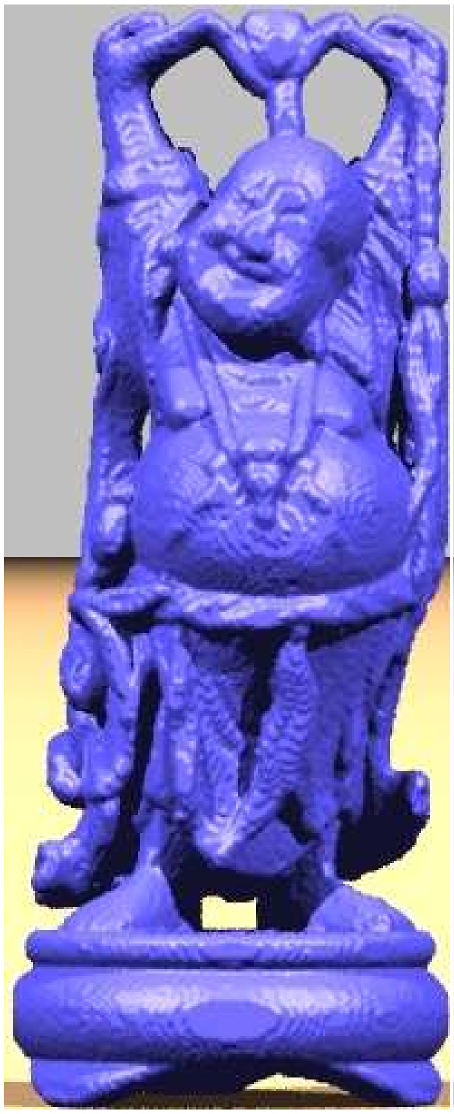
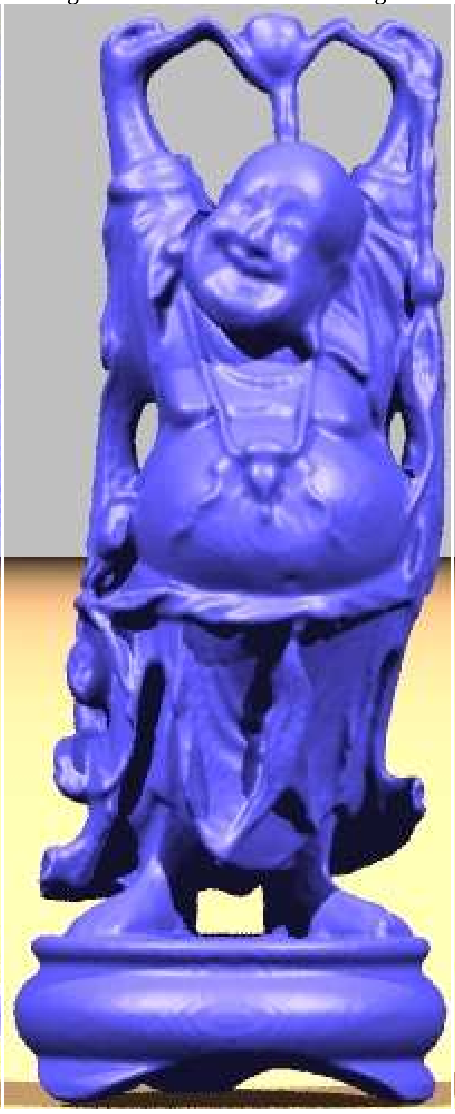

(d) low resolution reconstruction

Figure 8: reconstruction of the dragon


(a) initial reconstruction


(b) final reconstruction

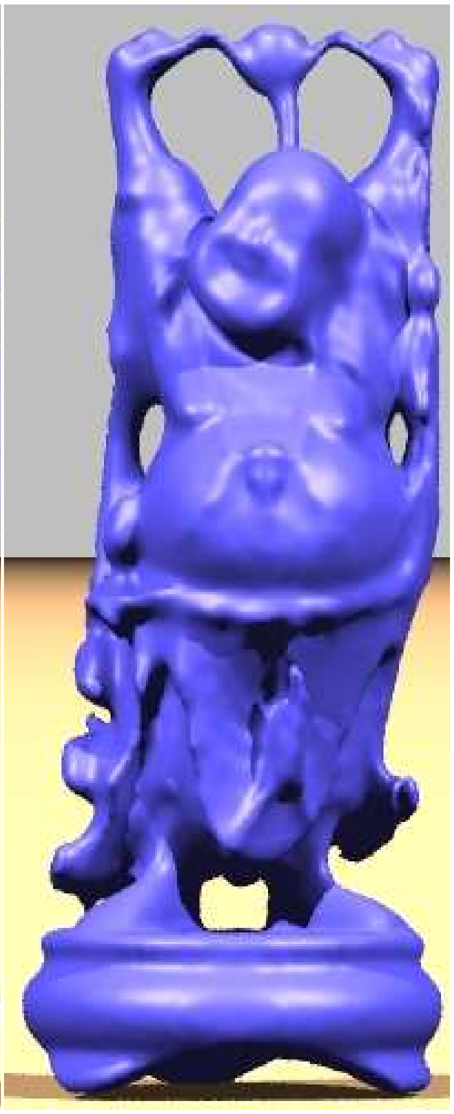

(c) reconstruction on a coarse grid

Figure 9: reconstruction of the Happy Buddha