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 topologies and deformations, as well as noisy or highly non-unifrom 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 a rigorous proof of the viability and convergence of our new fast tagging algorithm.

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

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 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 [22, 23], 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, 5, 8, 12, 13] 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. There are two main approaches for creating and analyzing implicit surfaces. The traditional approach [7, 18, 27, 29] uses a combination of smooth basis functions, 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. Although the implicit surface is usually smooth, the construction is global, i.e. all 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 [6, 9, 16]. 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 [11, 15] where partial connections are available on each piece of data and some "zippering" is needed to patch things together. The 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. Moreover an extremely efficient marching cubes algorithm [17] is available to turn an implicit surface into a triangulated surface.

We approach this fundamental problem 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 [31] 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 addition, in our formulation there is a regularization that is adaptive to the local sampling density which can keep sharp features if the a local sampling condition is satisfied. The formulation handles noisy as well as non-unform 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

*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

energy functional for the final reconstruction. Instead of tracking a parametrized explicit surface we solve an 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. We also introduce a smoothing algorithm similar to [28] as a post process to smooth implicit surfaces or reconstructed implicit surfaces from noisy data.

In the next section we briefly review the variational formulation for the weighted minimal surface model in introduced in [31]. A physically motivated simple convection model is developed in section 3. In section 4 we introduce the level set method for our problems and a simple denoising/smoothing formulation for implicit surfaces . We explain the details of the numerical implementation and fast 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 [31] 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 [31] 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 $\nabla d \cdot \boldsymbol{n} \le 1$, a local sampling density condition similar to the one proposed in [4], which says sampling densities should be proportional to the local curvature of the feature. To construct the minimal surface we used a continuous deformation in [31]. 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 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 [31].

In two dimensions, it was shown in [31] 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.

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

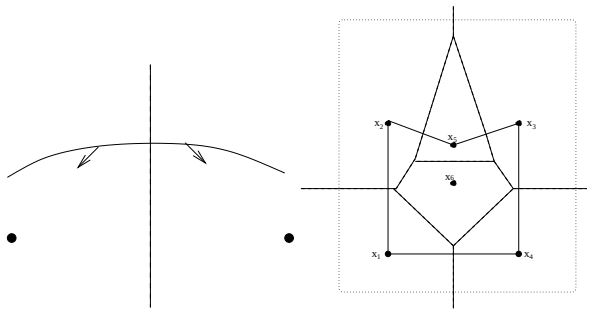
$$\frac{d\boldsymbol{\Gamma}(t)}{dt} = -\nabla d(\boldsymbol{x}). \tag{4}$$

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 (see Fig.1(b)), 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.

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.



(a) (b)

(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:

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 introduce 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 rendering and were based on discrete formulations [7]. 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, 21]. 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 [20] 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 [19] for a comprehensive review.

Two key steps for the level set method are:

- Embed the surface: we initially represent a co-dimension one surface $\mathbf{\Gamma}$ as the zero isocontour of a scalar (level set) function $\phi(\boldsymbol{x})$, i.e. $\mathbf{\Gamma} = \{\boldsymbol{x} : \phi(\boldsymbol{x}) = 0\}$. $\phi(\boldsymbol{x})$ is negative inside $\mathbf{\Gamma}$ and positive outside $\mathbf{\Gamma}$. Geometric properties of the surface $\mathbf{\Gamma}$, such as the normal, surface area, volume, mean and Gaussian curvature can be easily computed using ϕ. For example, the outward unit normal $\boldsymbol{n}$ is simply $\frac{\nabla\phi}{|\nabla\phi|}$ and the mean curvature κ is $\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}$.

- 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. the moving surface coincides with the zero level set for all time. Since $\mathbf{\Gamma}(t) = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t) = 0\}$,

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

where we replace $\frac{d\mathbf{\Gamma}(t)}{dt}$ with a velocity field $\boldsymbol{v}(\boldsymbol{x})$ defined for all $\boldsymbol{x}$ and equal to $\frac{d\mathbf{\Gamma}(t)}{dt}$ for $\boldsymbol{x}$ on $\mathbf{\Gamma} = \{\boldsymbol{x} : \phi(\boldsymbol{x}, t) = 0\}$.

To develop the time evolution PDE for the level set function, one needs to extend the velocity at the zero level set, which is given by the motion law of the original surface, to other level sets in a natural way. 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. Using the fact (see, e.g., [25, 31])

$$\int_{\Gamma} d^p(\boldsymbol{x}) ds = \int d^p(\boldsymbol{x}) \delta(\phi(\boldsymbol{x})) |\nabla\phi(\boldsymbol{x})| d\boldsymbol{x},$$

where $\phi(\boldsymbol{x}, t)$ is the level set function whose zero level set is $\mathbf{\Gamma}(t)$ and $\delta(x)$ is the one dimensional delta function, and extending the motion (normal velocity) to all level sets we have the level set formulation for the gradient flow (2)

$$\frac{\partial\phi}{\partial t} = \frac{1}{p}|\nabla\phi| \left[\int d^p(\boldsymbol{x})\delta(\phi)|\nabla\phi| d\boldsymbol{x}\right]^{\frac{1}{p}-1} \nabla\cdot\left[d^p(\boldsymbol{x})\frac{\nabla\phi}{|\nabla\phi|}\right], \tag{6}$$

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 for 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. [21, 25], to redistance the level set function locally without interfering with the motion of the zero level set. The reinitialization process will also provide us with a signed distance function for rendering the implicit surface after the deformation procedure stops.

If the data set contains noise, we derive a post-smoothing process similar to that of [28] for our reconstructed implicit surfaces using the variational level set formulation. Let ϕ_0 denote the initial level set function whose zero level set is the surface we would like to denoise or smooth. We define the denoised or smoothed implicit surface as the zero level set of ϕ that minimizes the following functional

$$\frac{1}{2}\int (H(\phi) - H(\phi_0))^2 d\boldsymbol{x} + \epsilon \int \delta(\phi)|\nabla\phi| d\boldsymbol{x}, \tag{8}$$

where $H(x)$ is the one dimensional Heaviside function. The first term in the above energy functional is a fidelity term that measures the symmetric volume difference between two closed surfaces. The second integral in the above functional is the surface area of the zero level set of ϕ, which is a regularization term that minimizes the surface area of the denoised or smoothed surface. The constant ϵ is a parameter that controls the balance between the fidelity and the regularization. We again find the minimizer by following the gradient flow of (8), whose level set formulation is:

$$\phi_t = |\nabla\phi|[\epsilon\kappa - (H(\phi) - H(\phi_0))]$$

To some extent this variational formulation is also related to Total Variation (TV) denoising for images proposed in [24]. In fact it is exactly TV denoising applied to $H(\phi)$, since the total variation of a function can be represented as the integration of the parameter length of all level sets of the function by co-area formula [14].

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:

$$|\nabla d(\boldsymbol{x})| = 1, \qquad d(\boldsymbol{x}) = 0,\ \boldsymbol{x} \in \mathcal{S}. \tag{9}$$

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 [10, 31] that combines upwind differencing with Gauss-Seidel iterations of different sweeping order to solve (9) 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.

Suppose we have a set of data points and a rectangular grid. We use an initialization procedure, of complexity $O(M+N)$ to assign initial values for N grid points and M data points. Those gird points that belong to the data set are assigned zero. Those grid points that are neighbors (i.e., vertices of grid cells that contain data points,) are assigned the exact distance values. These grid points are our boundary points and their distance values will not change in later computations. We assign a large positive number to all other grid points. These values will be updated in later computations. We can deal with more general data set as long as the distance values on grids neighboring to the set are provided initially. In one dimension, the following upwind differencing is used to discretize the Eikonal equation (9) at ith grid point that are not boundary points,

$$[(d_i - d_{i-1})^+]^2 + [(d_i - d_{i+1})^+]^2 = h^2, \ \ i = 1, 2, \ldots, I \tag{10}$$

where h is the grid size, I is the total number of grids and $(x)^+ = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases}$. We use two different sweeps of Gauss-Seidel iterations successively, i.e., for $i = 1 : I$ and $i = I : 1$, to solve this system of equations. At the ith grid, using the current values of d_{i-1} and d_i, there exists at least one solution for equation (10) $[min(d_{i-1}, d_{i+1}) + \frac{h}{\sqrt{2}}, max(d_{i-1}, d_{i+1}) + \frac{h}{\sqrt{2}}]$, which only depends on neighbors with smaller values. We take d_i to be the smaller one if there are two solutions. It can be shown that these two sweeps will get the exact solution of the discrete system (10), which is of first order $O(h)$ accuracy to the real distance function. In two dimensions, a slightly more complicated system,

$$\begin{aligned}&[(d_{i,j} - d_{i-1,j})^+]^2 + [(d_{i,j} - d_{i+1,j})^+]^2 \\ &+[(d_{i,j} - d_{i,j-1})^+]^2 + [(d_{i,j} - d_{i,j+1})^+]^2 = h^2,\end{aligned}$$

$i = 1, \ldots, I,\ j = 1, \ldots, J$, has to be solved using sweeps of Gauss-Seidel iterations of four different orders,

$$\begin{aligned}&(1)\ i = 1 : I, j = 1 : J \quad (2)\ i = 1 : I, j = J : 1 \\ &(3)\ i = I : 1, j = 1 : J \quad (4)\ i = I : 1, j = J : 1\end{aligned}$$

In most numerical computations, a total of five or six sweeps is enough in two dimensions. Similarly a three dimensional extension is straight forward. This distance 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 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 can 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. Here is the description of our fast tagging algorithm and the proof of its viability. For simplicity of exposition only, we shall consider a uniform grid

$$(x_i, y_j) = \boldsymbol{x}_{ij} = (i\Delta, j\Delta), \ \ i, j = 0, \pm 1, \pm 2, -$$

in 2 dimensions, where Δ is the grid size. The results work in any number of dimensions and for more general grid structure.

Let $d_{ij} = d(\boldsymbol{x}_{ij})$ be the unsigned distance of $\boldsymbol{x}_{ij}$ to the data set $\mathcal{S}$ (i.e. to the closest point on $\mathcal{S}$). We say $\boldsymbol{x}_{ij} < \boldsymbol{x}_{kl}$, or $\boldsymbol{x}_{ij}$ is closer than $\boldsymbol{x}_{kl}$ or $\boldsymbol{x}_{ij}$ is smaller than $\boldsymbol{x}_{kl}$ if $d_{ij} < d_{kl}$.

We define $\mathcal{S}^\Delta$ to be the set of grid nodes $\boldsymbol{x}_{ij}$ for which $d_{ij} < \Delta$. (Note: if every data point lies on a grid node, then $\mathcal{S}^\Delta = \mathcal{S}$).

We wish to obtain a set Ω for which $\mathcal{S}^\Delta \subset \Omega$ and for which the boundary $\partial\Omega$ serves as a very good initial guess for our final reconstructed surface. From our convection model, we need to rapidly find a crude approximate solution to the steady state equation

$$\nabla d(x) \cdot \boldsymbol{n} = 0,$$

where $\boldsymbol{n}$ is the unit outward normal of the boundary $\partial\Omega$. This equation can be written (in 2D) as

$$d_x \phi_x + d_y \phi_y = 0, \tag{11}$$

where ϕ is the level set function whose zero level set is $\partial\Omega$ that surrounds $\mathcal{S}^\Delta$. We wish to march quickly in a manner reminiscent of the fast algorithm of [26], but for a very different problem – this is not the eikonal equation, and steady states generally depend on the initial guess. There are some similarities in that a heap sort algorithm is used as is the Cartesian structure of the grid.

For a point $\boldsymbol{x}_{ij}$ we define its neighbors as the four points $\boldsymbol{x}_{i\pm1,j}, \boldsymbol{x}_{i,j\pm1}$. A boundary point of a set Ω is defined to be the set of $\boldsymbol{x}_{ij}$ in Ω for which at least one of its four neighbors are in the exterior of Ω. The boundary of Ω is denoted by $\partial\Omega$.

Given Ω_0 for which $\mathcal{S}^\Delta$ we shall march quickly towards $\Omega \subset \Omega_0$ whose boundary $\partial\Omega$ will act as our initial level surface for the convection and convection-diffusion algorithms defined below. This is our fast tagging algorithm.

We begin by considering $\partial\Omega_0$ which we order in a nondecreasing sequence via a heap sort algorithm. We denote $\partial\Omega_0$ as a temporary boundary set at stage 0. We shall inclusively create Ω_n and a temporary boundary set $\partial\Omega_{n,t} \subset \partial\Omega_n$ so that, after a finite number of steps the largest point in $\partial\Omega_{n,t}$ is also in $\mathcal{S}^\Delta$, at which point the algorithm terminates and Ω_n is the final Ω. At each marching step, we either tag the largest (furthest) temporary boundary point into the final boundary or turn the largest (furthest) temporary boundary

point into an exterior point. This fast tagging algorithm is of complexity $O(N \log N)$; the log N term appears because of the sorting step.

The tagging algorithm is as follows: Consider the largest $\boldsymbol{x}_{ij}^{(n)} \in \partial\Omega_{n,t}$.

(a) If there is at least one interior neighbor of $\boldsymbol{x}_{ij}^{(n)}$ which is not closer to $\mathcal{S}$, put $\boldsymbol{x}_{ij}^{(n)}$ into the tagged boundary set and define $\partial\Omega_{n+1,t} = \partial\Omega_{n,t} - \{\boldsymbol{x}_{ij}^{(n)}\}, \Omega_{n+1} = \Omega_n$.

(b) If all interior neighbors of $\boldsymbol{x}_{ij}^{(n)}$ are closer to $\mathcal{S}$, put $\boldsymbol{x}_{ij}^{(n)}$ into the exterior and include its interior neighbors into the new temporary boundary, i.e., define $\Omega_{n+1} = \Omega_n - \{\boldsymbol{x}_{ij}^{n}\}$ and $\partial\Omega_{n+1,t} = \partial\Omega_{n+1} - \{\partial\Omega_n - \partial\Omega_{n,t}\}$.

Repeat this process until either (a) the temporary boundry set becomes empty, or (b) maximum distance of the untagged temporary boundary points, (the set $\partial\Omega_{n,t}$) to $\mathcal{S}$ is less than Δ.

We now prove the algorithm is viable and converges. If condition (a) is satisfied at stage n, then since $\partial\Omega_{n+1,t} \subset \partial\Omega_{n,t}$ $\boldsymbol{x}_{ij}^{(n+1)} \leq \boldsymbol{x}_{ij}^{(n)}$. If condition (b) is satisfied then $\partial\Omega_{n+1,t}$ will include new points that are neighbors of $\boldsymbol{x}_{ij}^{(n)}$ and are closer to $\mathcal{S}$ than $\boldsymbol{x}_{ij}^{(n)}$. Thus $\boldsymbol{x}_{ij}^{(n+1)} \leq \boldsymbol{x}_{ij}^{(n)}$. This ends the proof that the algorithm is viable and converges.

Remark1: Our tagging algorithm produces a very crude approximation to the steady state solution of the convection equation, which we rewrite: $\nabla d \cdot \nabla \phi = 0$. We solve this crudely on a grid for a function $\phi_i j$ which has value either +1 or -1. We initialize so that $\phi_i j = 1$ in the exterior of $\Omega_0, \phi_i j = -1$ inside Ω_0. At every grid point $\boldsymbol{x}_i j$ to be updated we march in an "upwind" direction, which means the new $\phi_i j$ depends only on values at the four neighbors which are further from $\mathcal{S}$ than $\boldsymbol{x}_i j$. Thus we order the temporary boundary points and update the largest untagged point via a crude process

$$\phi_{ij} = P[\phi^n_{i-1,j}, \phi^n_{i+1,j}, \phi^n_{i,j-1}, \phi^n_{i,j}]$$

where P denotes a procedure that picks one of the values from its arguments that corresponds to a more remote point as follows: if there are any more remote interior points, it picks the one which is furthest from $\mathcal{S}$. Else it picks the furthest exterior point. This is equivalent to using a convex combination of either interior or exterior points to approximate ∇d and enforce ϕ to be constant along ∇d It is easy to see that this approximates the level set equation (11) and is exactly our tagging algorithm.

Remark2: At every stage of our tagging algorithm, all points $x^n_{\nu\mu}$ which are interior points of Ω_n and which are more remote than the point being tagged, x^n_{ij}, will remain in Ω_N for all $N > n$, and hence in the final set Ω, since the maximum distance on the untagged temporary boundary is decreasing. Thus we generally obtain a nontrivial limit set Ω.

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 tagging method 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 any exterior point, such as the corners of of our rectangular domain, and expanding the exterior region by repeatedly tagging those grid points which are connected to the starting exterior point and have a distance larger than ϵ as exterior 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 $d(\boldsymbol{x}) = \epsilon$ contour, first proposed in [31], 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 set is fine enough to resolve the real surface, then we can find an appropriate ϵ and get a very good initial surface with $O(N+M)$ operations. When combined with the above fast tagging algorithm, we can find a good initial approximation very efficiently. For non-uniform data points the intersection of a bounding box and a distance contour with moderate ϵ, which is a simple Boolean operation for implicit surfaces, often gives a good initial surface.

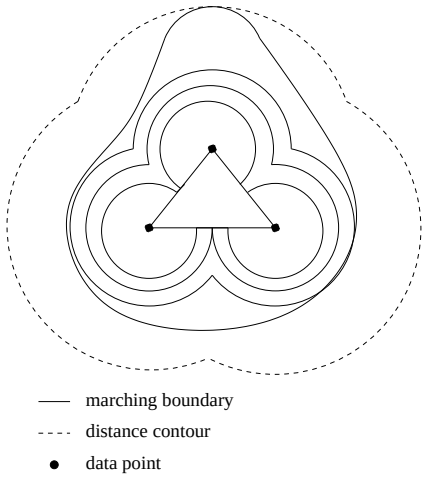


Figure 2:

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. The one dimensional Delta function $\delta(x)$ and Heaviside function $H(x)$ are approximated numerically if needed.

Details can be found in, for example, [21, 30, 31]. The convection model is simple and fast 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 complicated and computationally expensive but reconstructs a smooth weighted minimal surface. These two continuous deformations can be combined, and 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 regularly 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 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 filesize 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 funtion 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 a few numerical examples that illustrate the efficiency and quality of our surface construction. In particular we show (1) how the level set method handles surface deformation and topological change easily, (2) how quickly our tagging algorithm constructs a good initial guess, (3) how smooth the reconstructed surfaces are by using either the convection model or the minimal surface model, (4) how our algorithm works with non-uniform, noisy or damaged data, and (5) how multiresolution works in our formulation. All calculations were done with a Pentium III, 600Mhz processor. Data points for the drill, the dragon and the Buddha were obtained from www-graphics.stanford.edu/data/3Dscanrep and data points for the hand skeleton and turbine blade were obtained from www.cc.gatech.edu/projects/large_models. Only locations of the data points are used in our reconstructions.

The first group of examples show surface reconstruction from synthesized data. Figure 4 show surface reconstruction, a torus, from damaged data, which is like hole filling. Figure 5 shows the reconstruction of a sphere from a box using eight longitudinal circles and eight latitudinal circles. For this example we do not have any discrete data points. We only provide the unsigned distance function. This can also be viewed as an extreme case of non-uniform data. Figure 5(a) shows those circles and figure 5(b) shows reconstruction using the convection model. Figure 5(c) shows the final minimal surface reconstruction following the gradient flow on top of figure 5(b).

The second group of examples are from real data. 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. For data sets that are fairly uniform, such as the drill, the dragon, the Buddha and the hand skeleton, 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 for a rat brain from MRI slices. The data set is very non-uniform and noisy. We start with the intersection of a bounding box and an outer distance contour with relatively large $\epsilon = 12h$, which is shown in figure 6(b). The next example, figure 7 is our reconstruction of a 1.6mm drill bit from 1961 scanned data points. It is a quite challenging example for most methods for surface reconstruction from unorganized data as is shown in [11]. Figure 8 shows the reconstruction of a hand skeleton. Figure 9 shows the reconstruction of the Happy Buddha. Figure 9(a) shows the initial reconstruction using the fast tagging algorithm only. We start with an outer distance contour, $d = 3h$, initially and it takes only 3 minutes for half a million points on a $146 \times 350 \times 146$ grid. Figure 10 is the reconstruction of the dragon. Figure 9(b) is the final reconstruction. Figure 9(c) is the reconstruction on a much under resolved coarse grid $63 \times 150 \times 64$ using the same amount of data points. It only takes 7 minutes and the result is quite good. For the example of the dragon, we show the initial reconstruction in figure 10(a), reconstruction using the convection model only in figure 10(b) and the final weighted minimal surface reconstruction in figure 10(c). Figure 10(d) shows the reconstruction using a much lower resolution data set on the same grid and the result is quite comparable to figure 10(c). The final example shows the reconstruction of a turbine blade on a $178 \times 299 \times 139$ grid for almost a million data points.

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

Model	Data points	Grid size	CPU (initial)	CPU (total)
Rat brain	1506	80x77x79	.12	3
Drill	1961	24x250x32	0.1	2
Buddha	543652	146x350x146	3	68
Buddha	543652	63x150x64	.3	7
Dragon	437645	300x212x136	4	77
Dragon	100250	300x212x136	3	66
Hand	327323	200x141x71	.5	10
Turbine blade	882954	178x299x139	2.5	60

Figure 3: timing table

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 and M. Bern. Surface reconstruction by Voronoi filtering. *14th ACM Symposium on Computational Geometry*, 1998.

[4] 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.

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

[6] 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.

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

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

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

[10] 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.

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

[12] 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.

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

[14] L.C. Evans and R.F. Gariepy. Measure theory and fine properties of functions. *Studies in Advanced Mathematics, CRC Press Inc.*, 1992.

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

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

[17] W.E. Lorenson and H.E. Cline. Marching cubes: A high resolution 3d surface recomstruction algorithm. *Computer Graphics*, 21:163–169, 1987.

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

[19] S. Osher and R. Fedkiw. Level set methods: an overview and some recent results. *to appear in J. Comp. Phys.*, 2000.

[20] 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.

[21] 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.

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

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

[24] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.

[25] 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.

[26] J.N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40((9)):1528–1538, 1995.

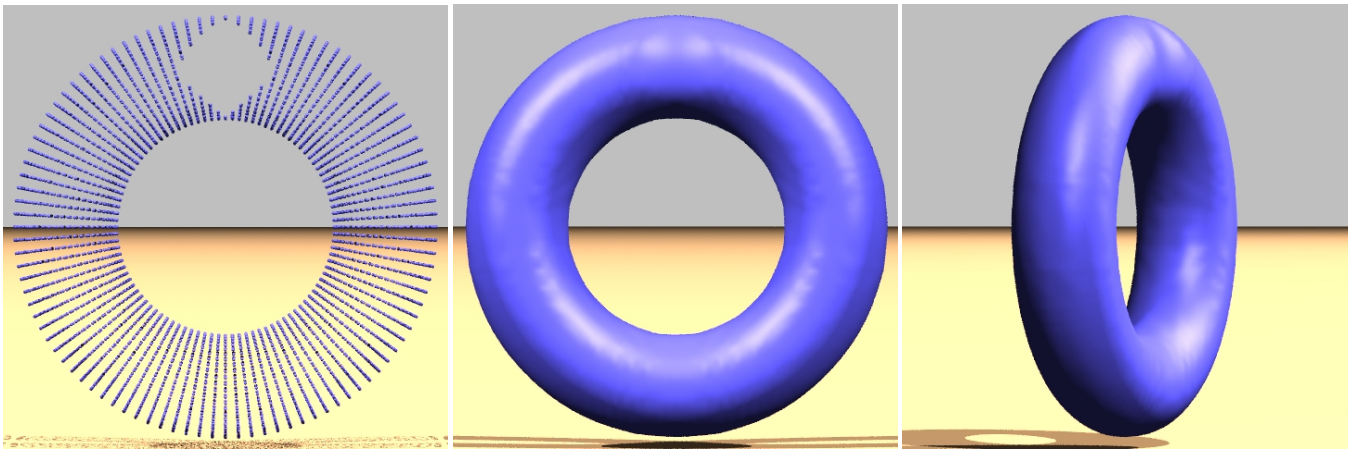
[27] G. Turk and J. ÒBrien. Shape transformation using variational implicit functions. *SIGGRAPH99*, pages 335–342, August 1999.

[28] R. Whitaker. A level set approach to 3D reconstruction from range data. *International journal of Computer Vision*, 1997.

[29] A.P. Witkin and P.S. Heckbert. Using particles to sample and control implicit surfaces. *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH94)*, pages 269–278, July 1994.

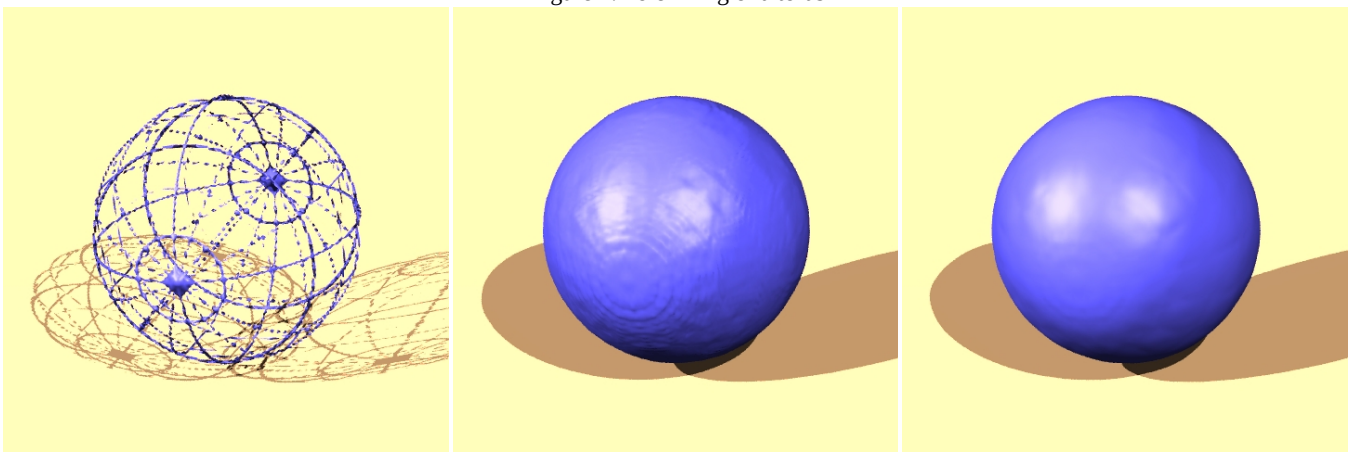
[30] H.K. Zhao, T.F. Chan, B. Merriman, and S.Osher. A variational level set approach to multiphase motion. *J. Comp. Phys.*, 127:179–195, 1996.

[31] 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.



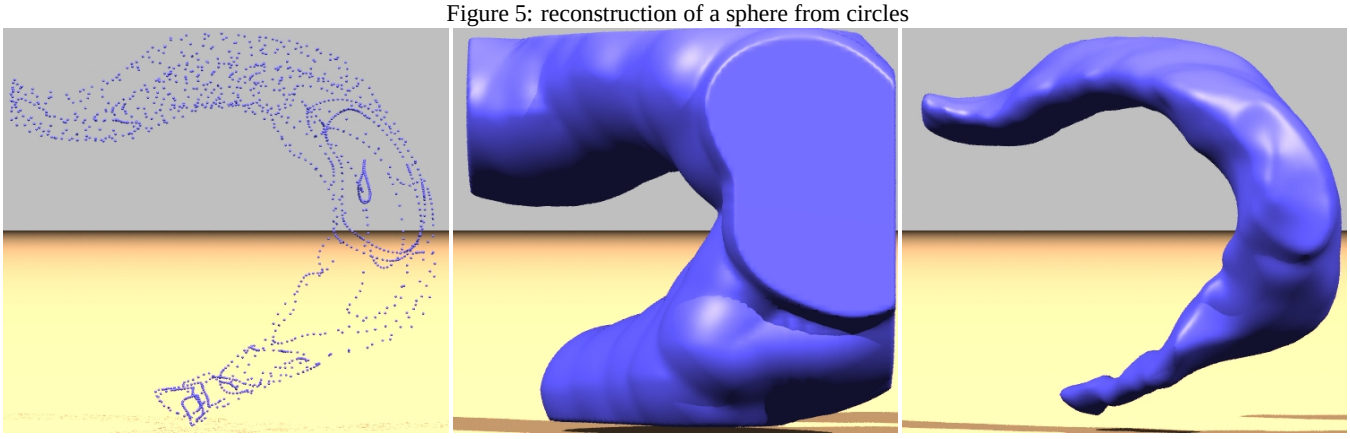
(a) data points (b) front view of the final reconstruction (c) side view of the final reconstruction

Figure 4: hole filling of a torus



(a) initial data (b) reconstruction using convection (c) final reconstruction

Figure 5: reconstruction of a sphere from circles



(a) data points (b) starting surface (c) final reconstruction

Figure 6: reconstruction of a rat brain

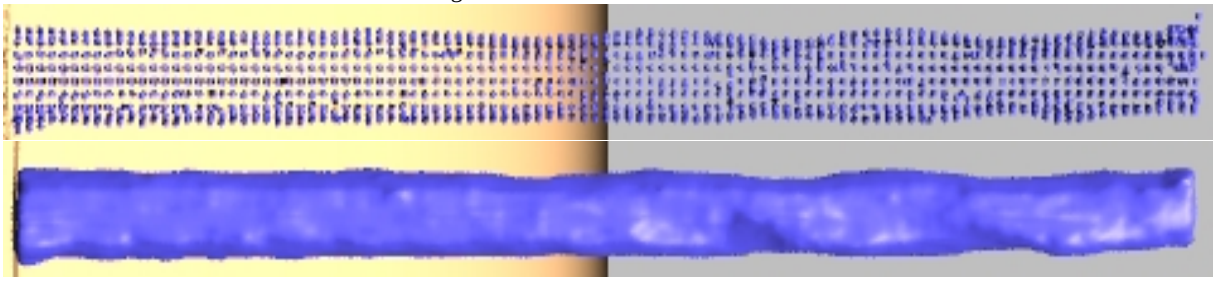
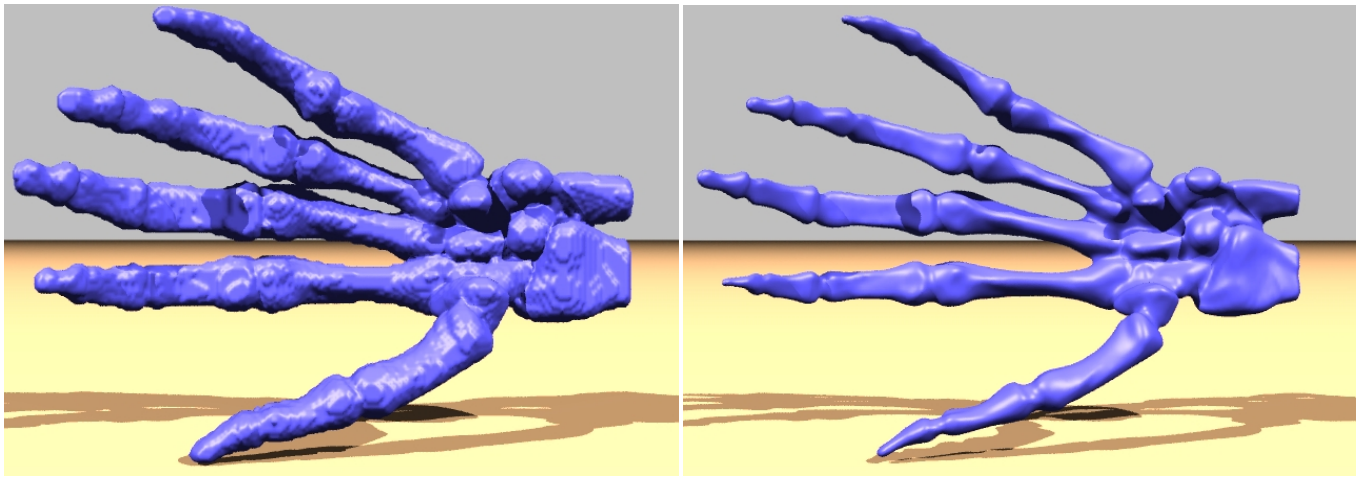
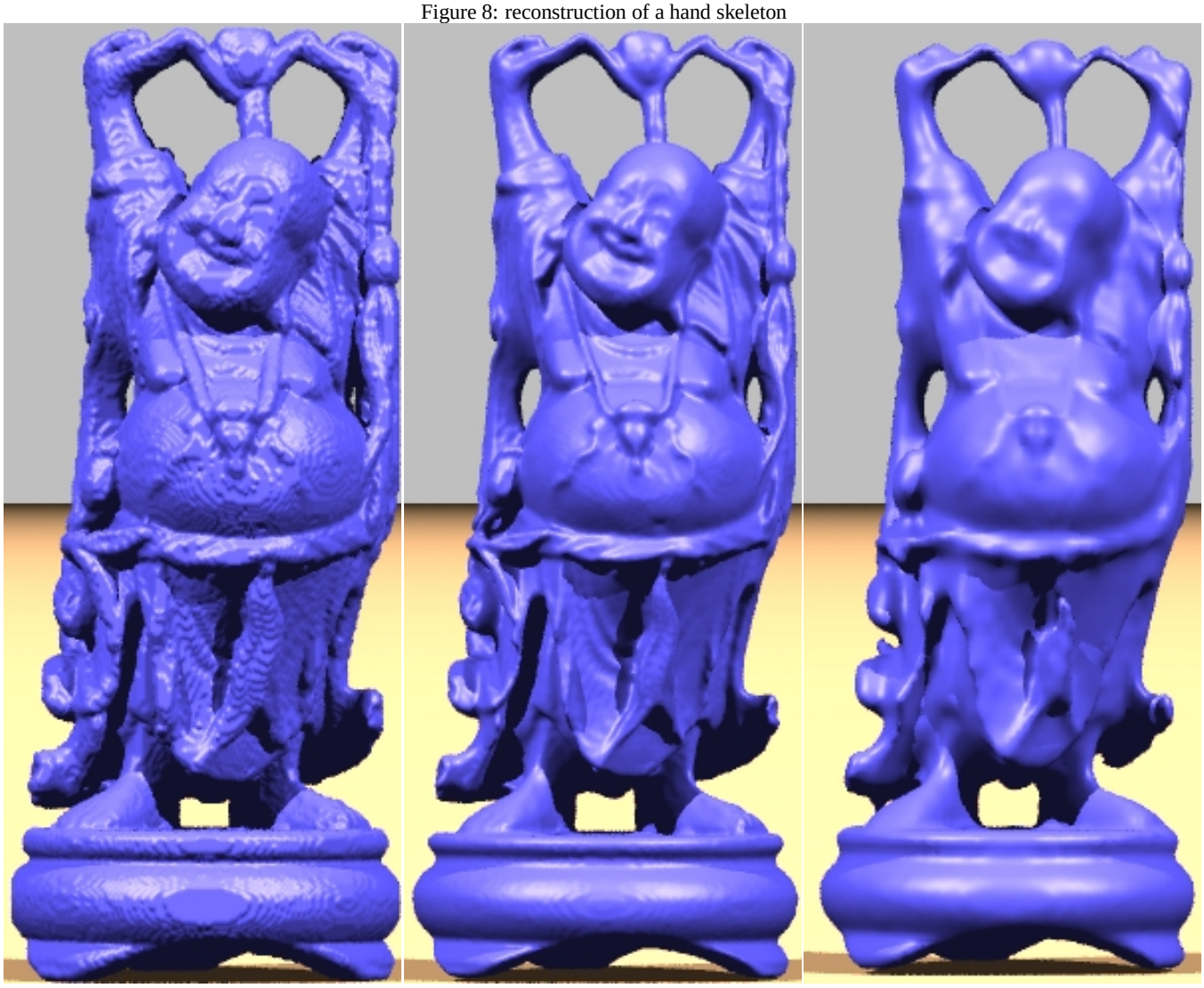


Figure 7: reconstruction of a drill



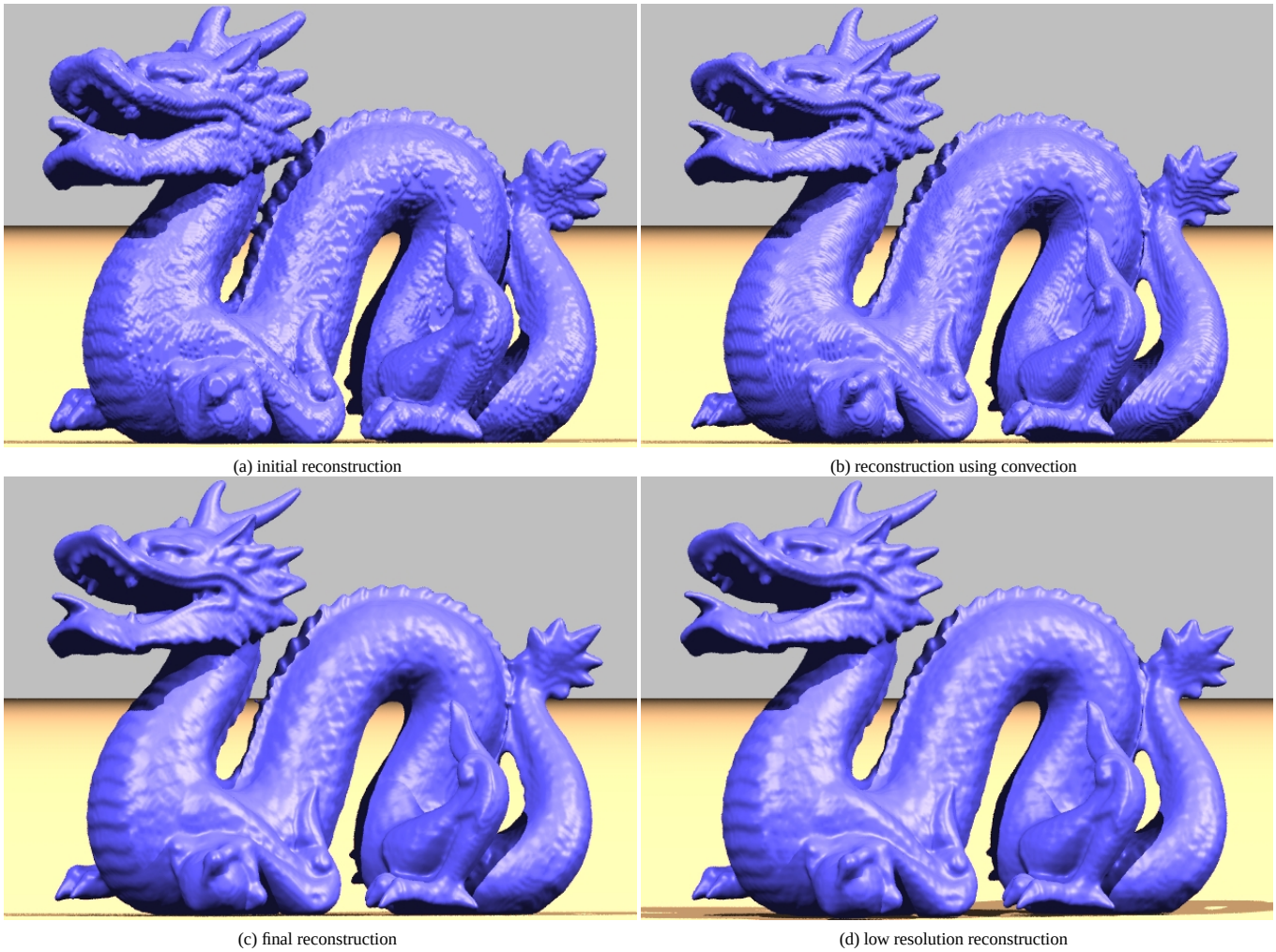
(a) initial reconstruction (b) final reconstruction

Figure 8: reconstruction of a hand skeleton



(a) initial reconstruction (b) final reconstruction (c) reconstruction on a coarse grid

Figure 9: reconstruction of the Happy Buddha



(a) initial reconstruction

(b) reconstruction using convection

(c) final reconstruction

(d) low resolution reconstruction

Figure 10: reconstruction of the dragon

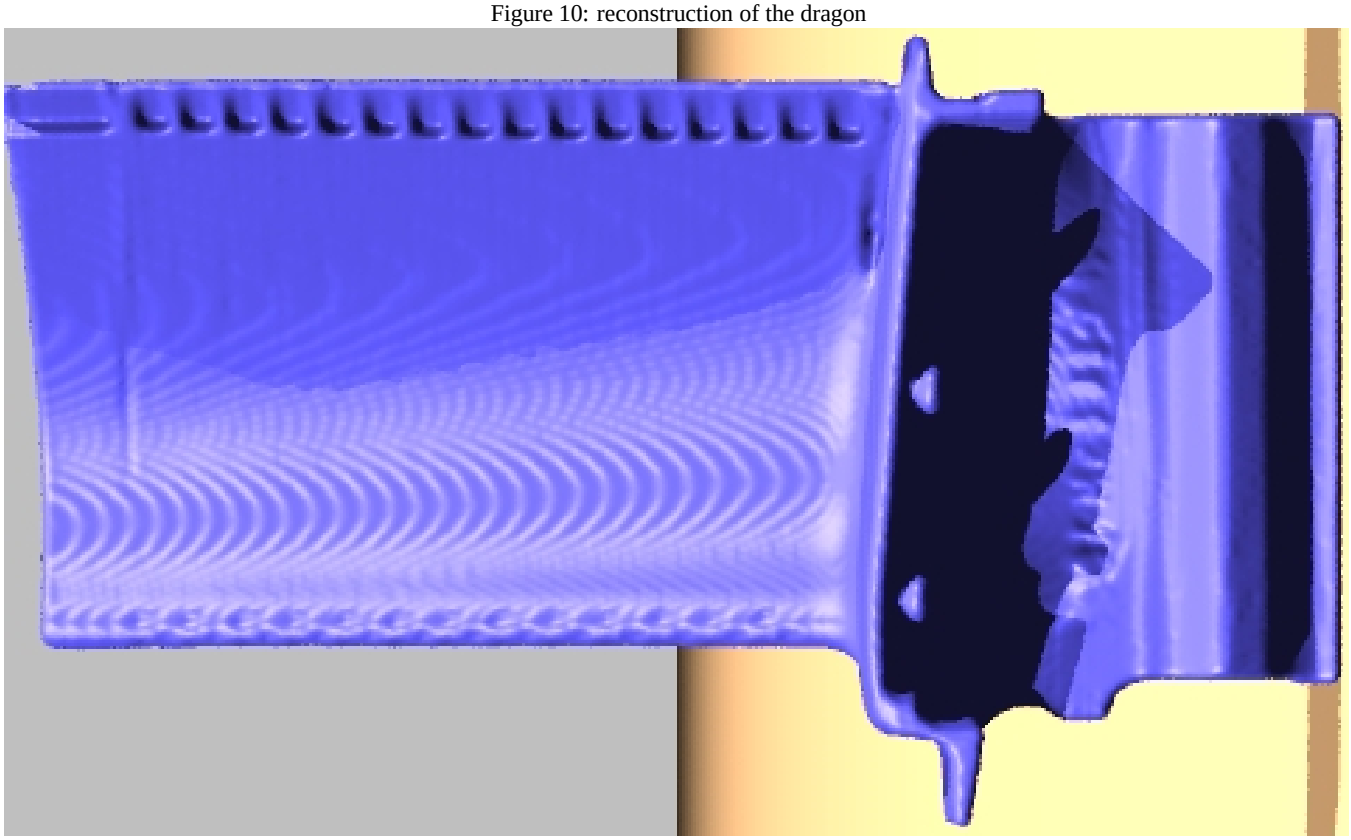


Figure 11: reconstruction of a turbine blade