Visualization, analysis and shape reconstruction of unorganized data sets

Hongkai Zhao
Stanley Osher

ABSTRACT
In this chapter mathematical models and efficient algorithms are developed for the visualization, analysis and shape reconstruction for an arbitrary data set that can include unorganized points or continuous manifolds of any codimension, such as pieces of curves and surface patches. The distance function to the data set and its contours are used for fast visualization and analysis of the data set. A minimal surface and a convection model are used for shape reconstruction from the data set. All formulations and numerical algorithms are based on implicit representations on simple rectangular grids which extend to any number of dimensions and which also can easily be combined with the level set method for dynamic shape deformation and other manipulations.

1 Introduction

Visualization and analysis of large data sets of unorganized points have important applications in scientific computing, computer graphics/vision, computer aided design, medical imaging etc. Mathematically, interpolation or shape reconstruction from a set of unorganized data points is an ill-posed problem, i.e., there is no unique solution. For large data sets in three and higher dimensions the problem becomes more challenging due to the following: (1) the geometry and topology of the real shape is not known á priori and can be very complicated, (2) finding the connection or ordering of the data points can be difficult and expensive, especially for data that are not uniformly spaced. In real applications, noise or other uncertainty in the data makes things even more complicated. A desirable procedure should be able to deal with all these difficulties and should have a representation and data structure that is not only good for static rendering but also good for dynamic deformation and other manipulations. Most of the previous approaches to this problem can be classified as continuous vs. discrete in formulation and explicit vs. implicit in representation. For continuous formulations, one often uses variational methods or partial

differential equations, e.g., [BW90, CG91, MT93, SPOK95], to define the desired solution in an appropriate function space equipped with a certain regularity (smoothness). The formulation is meant to combine the interpolation or other constraints with a regularization to remove the ill-posedness. Approaches using continuous formulations are usually robust and allow flexibility in dealing with noise. However the choice of regularization in the formulation is not obvious and can make the problem difficult to solve numerically. For discrete approaches, one tries to sort out connections or orderings among data points based on exploring precise local structure and relation of points, lines and planes etc. Interpolation is then used to connect points or reconstruct the shape in a piecewise smooth fashion. Discrete approaches are usually based on simple geometric relations and attempt to interpolate the data exactly. However, in three and higher dimensions, sorting out correct connections for an arbitrary data set can be very difficult. Furthermore, the global structure pieced together from local information may not be consistent or may lack smoothness. Noise or uncertainty in the data can also be a problem.

The representation of reconstructed surfaces (shape) can be classified as 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 using NURBS e.g., [PT97, Rog00], and triangulated surfaces using Voronoi diagrams and Delaunay triangulations e.g., [ABE98, ABK98, Boi84, Ede98, EM94]. Tracking of large deformations and topological changes can be a problem using explicit surfaces.

Recently, implicit surfaces or volumetric representations have attracted a lot of attention. The traditional approach [BBB+97, Mur91, TO99] uses a combination of smooth basis functions (primitives) 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. The computational cost is very high for large data sets, since the construction is global. This requires solving a large linear system and a single data point change can result in changes of all the coefficients. Recently, in [CBC+01] polyharmonic Radial Basis Functions (RBF) and multipole methods were used, enabling the authors to model large data sets by a single RBF. However, human interaction and dynamic deformation are still difficult tasks. More recently, the signed distance function has been used to reconstruct and represent an implicit surface on a rectangular grid with the signs to distinguish inside and outside [BBX95, BC00, HDD+92]. Most of the constructions of signed distance functions from unorganized points are based on discrete approaches. Similar ideas have been applied to shape reconstruction from range data and image fusion [CL96, HSIW98]. The main advantages of implicit surfaces include topological flexibility, a simple data structure and depth/volumetric information. Using the signed distance representation, many surface operations such as Boolean operations, ray tracing and

computing offsets become quite simple [PASS95, WGG99]. Efficient algorithms, see e.g. [NB93, WMW86], are available to turn an implicit surface into a triangulated surface. In fact the level set method [OS88] provides a general framework for the deformation of implicit surfaces according to arbitrary physical and/or geometric rules. Recent applications based on implicit surfaces and level set method range from computer animations, dynamic visibility, and solving partial differential equations on manifolds, see e.g., [DCG98, FF01, BCOS01, CBMO02, TBC+02].

In this chapter, we present some recent work by the authors and collaborators [ZOMK00, ZOF01, Zha02a] on visualization, analysis and shape reconstruction for unorganized data set based on continuous formulations and implicit representations. Our algorithms are developed based on rectangular grids and work in any number of dimensions.

2 Fast multiscale visualization and analysis of large data sets using distance functions

In many applications, for data sets that are noisy and of large size, perhaps involving multiple dimensions and with complicated geometry and topology, visualization or analysis techniques based on interpolation, which requires one to explore the exact relations among all data points, often result in high computational cost. In many situations we only need representations that are good enough for approximate analysis and visualization, i.e., one can compromise between the interpolation accuracy and efficiency. At the same time we also want to keep the consistency and fidelity of desired features and topological/geometric properties of the data set as much as possible. A consistent multiresolution and multiscale framework is another desired property for these approximate procedures.

In [Zha02a] efficient multiscale data analysis and processing procedures based on simple applications of distance function, distance contours and connectivity are proposed that can:

- find all disconnected components on a given scale,
- provide quick visualization on different resolution and scale,
- characterize and approximate some important topological or geometric properties of the data set,
- extend to any number of dimensions.

The basic idea is that the distance contours of the data set can be viewed as approximate offsets of the shape represented by the data set and can be used to visualize the data, to dissect disconnected components and to characterize some important topological and geometric information for the

data set. Efficient numerical algorithms are developed based on rectangular grids to compute the distance function to an arbitrary data set, extract appropriate distance contours, and identify and characterize each disconnected component. The grid resolution and the choice of the distance contour depend on the prescribed scale or the data sampling density. A natural multiresolution framework can be implemented in a hierarchical way. The complexity of the whole algorithm is of order $N + M$, where N is the number of grid points and M is the number of the data points.

2.1 The distance function and the fast sweeping method

The distance function $d(\mathbf{x}) = dist(\mathbf{x}, \mathcal{S})$ is the most important and intrinsic information for a data set $\mathcal{S}$. It is invariant under rotation and translation. Given the distance function to a data set, we can choose an appropriate scale parameter ϵ, which may be prescribed or may depend on the sampling density of the data, so that:

1. the distance contour $d(\mathbf{x}) = \epsilon$, with a good choice of ϵ, is an approximate ϵ-offset of the manifold represented by the data set and thus can be used to approximately represent and visualize the shape of the data set.

2. the ϵ distance contour can be used to dissect and analyze the disconnected components of the data set on the scale ϵ.

3. the ϵ-shell$= \{\mathbf{x} : d(\mathbf{x}) < \epsilon\}$ can be viewed as an approximate ϵ-covering of the manifold represented by the data set and thus can be used to approximate the Hausdorff dimension of each disconnected piece of the data set

In numerical computations we compute the distance function on a rectangular grid that contains the data set with a grid size h that resolves the prescribed scale ϵ. We then construct the ϵ distance contour or ϵ-shell, and dissect and characterize the disconnected components on this grid. To compute the distance function to an arbitrary data set on a rectangular grid, we use the fast sweeping algorithm that was used in [ZOMK00] and analyzed in detail in [Zha02b]. The distance function $d(\mathbf{x})$ to an arbitrary data set $\mathcal{S}$ solves the following Eikonal equation:

$$|\nabla u(\mathbf{x})| = 1, \qquad u(\mathbf{x}) = 0, \ \mathbf{x} \in \mathcal{S}. \tag{1.1}$$

From a PDE point of view, the information propagates along straight lines away 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 the following fast sweeping method that combines upwind differencing with Gauss-Seidel iterations of different sweeping orders to solve the equation (1.1) on rectangular grids. For simplicity of exposition, the

algorithm is presented in two dimensions. Extensions to higher dimensions are straightforward. We use $\mathbf{x}_{i,j}$ to denote the grid points at which distance values are to be computed. h is the grid size and $u^h(\mathbf{x})$ denotes the numerical solution.

The fast sweeping algorithm:

1. Discretization:
   At interior grid points the following upwind difference scheme is used to discretize the PDE (1.1):
   $$\begin{array}{l}[(u^h_{i,j} - u^h_{xmin})^+]^2 + [(u_{i,j} - u^h_{ymin})^+]^2 = h^2, \\ i = 2, \ldots, I-1,\ j = 2, \ldots, J-1,\end{array} \tag{1.2}$$
   where $u^h_{xmin} = \min(u^h_{i-1,j},\ u^h_{i+1,j})$, $u^h_{ymin} = \min(u^h_{i,j-1}, u^h_{i,j+1})$ and $(x)^+ = \begin{cases} x & x > 0 \\ 0 & x \le 0 \end{cases}$. At the boundary, one sided difference is used if needed. For example, at the left boundary, one sided difference is used in the x direction,
   $$[(u^h_{1,j} - u^h_{2,j})^+]^2 + [(u_{1,j} - u_{ymin})^+]^2 = h^2.$$

2. Initialization:
   Assign exact values or interpolated values at grid points in or near $\mathcal{S}$. These values are fixed in later calculations. Assign large positive values at all other grid points. These values will be updated later.

3. Gauss-Seidel iterations with alternating sweeping orders:
   At each grid $\mathbf{x}_{i,j}$ compute the solution, denoted by $\overline{u}$, of (1.2) from the current value of its neighbors $u^h_{i\pm1,j}, u^h_{i,j\pm1}$ and then update $u^h_{i,j}$ to be the smaller one of $\overline{u}$ and its current value, i.e., $u^{new}_{i,j} = \min(u^{old}_{i,j}, \overline{u})$. We sweep the whole domain with four alternating orderings repeatedly,
   $$\begin{array}{ll}(1)\ i = 1:I, j = 1:J & (2)\ i = I:1, j = 1:J \\ (3)\ i = I:1, j = J:1 & (4)\ i = 1:I, j = J:1\end{array}$$

Computing local distance: The computation of the distance function can be easily restricted to a neighborhood of the the data set using a simple cutoff criterion. For example, if we want to restrict the distance computation to the neighborhood $\{\mathbf{x}_{i,j} : d(\mathbf{x}_{i,j}) < \overline{d}\}$, in the Gauss-Seidel iteration we update the distance value at a grid point $\mathbf{x}_{i,j}$ only if at least one of its neighbors has a distance value smaller than $\overline{d}$, i.e., if $\min(u^h_{xmin}, u^h_{ymin}) < \overline{d}$.

Implementation details can be found in [Zha02b], in which it is shown that 2^n number of sweeps is enough to compute the distance function to an arbitrary data set in n dimensions. The alternating sweeping idea is similar to the Danielsson's algorithm [Dan80]. However our formulation is based on solving the PDE (1.1) and can treat much more general situations [TCOZ01].

2.2 Dissection and Characterization of Disconnected Components

After we have computed the distance function on a rectangular grid, we can use an appropriate distance contour and a connectivity condition to classify all grid points as exterior points, neighboring points and interior points associated with each disconnected component. The boundaries between these points are corresponding distance contours which can be used to visualize and characterize the data set. For simplicity of exposition our notation and algorithms are defined for two dimensions. The extension to three and higher dimensions is straightforward.

Given a scale ϵ, which is resolved by the grid on which the distance function has been computed, we define the following three sets for all grid points $\mathbf{x}_{i,j}$.

- the set of exterior points, denoted by Ω^E:
  $\Omega^E = \{\mathbf{x}_{i,j} | d(\mathbf{x}_{i,j}) > \epsilon \text{ and } \mathbf{x}_{i,j} \text{ is connected to infinity}\}$
- the set of neighboring points, denoted by Ω^N:
  $\Omega^N = \{\mathbf{x}_{i,j} | d(\mathbf{x}_{i,j}) < \epsilon\}$
- the set of interior points, denoted by Ω^I, are the grid points complimentary to $\Omega^E \cup \Omega^N$.

We define the connected neighboring points of a grid point $\mathbf{x}_{i,j}$ to be $\mathbf{x}_{i\pm1,j}, \mathbf{x}_{i,j\pm1}$ in two dimensions and define the boundary of a set Ω, denoted by $\partial\Omega$, to be the set of those grid points in Ω for which at least one of its four neighbors is not in Ω.

We first identify the set of exterior grid points, Ω^E. We start with an arbitrary initial subset $\Omega_0^E \subset \Omega^E$. For example, Ω_0^E can be a known exterior region or simply a seed point in Ω^E such as a vertex point of the computational domain. We expand the initial set of exterior points Ω_0^E to Ω^E by repeatedly marching the boundary of the set of temporary exterior points, denoted by Ω_t^E, by adding those grid points that are connected to the boundary $\partial\Omega_t^E$ and have a distance value that is greater than ϵ. The expansion stops when there is no more connected exterior grid point to add. After the identification of all exterior points we also have the boundary between Ω^E and Ω^N, which can be used to visualize the data set as is shown below. For the unmarked grid points, we can easily identify those that have a distance value less than ϵ and mark them as neighboring points, and the remaining unmarked points are the interior points. With careful bookkeeping and marking, every grid point needs to be visited and checked only once.

The most subtle point in the classification of all grid points is the choice of the scale ϵ. If the sampling density of the data set satisfies a separation condition, i.e., the distance between two disconnected components or disjoint

parts is larger than the largest distance between two connected neighboring data points, (which is a reasonable assumption to avoid any ambiguity about connectivity,) then we can use any ϵ in this range to separate all disconnected components. The set of neighboring points can be regarded as an ϵ covering of the real manifold represented by the data set. Moreover, if the real manifold represented by the data set is closed, the union of the neighboring set Ω^N and the interior set Ω^I has the same topology as the interior region enclosed by the real manifold. Hence the boundary between the set of exterior points and the set of neighboring points, which we have identified in the above classification algorithm, is homeomorphic to the real manifold and can be regarded as an approximate ϵ offset. If the real manifold represented by the data set is open and the distance between disjoint parts is larger than the largest distance between two connected neighboring points, the boundary between the set of exterior points and the set of neighboring points looks like a thin shell enclosing the true manifold and can be still used as a good approximation in visualization, as is shown below. All the above situations are demonstrated in figure 1, in which the red curves correspond to an distance contour of the distance function to a data set represented by the blue dots. The data set contains several disconnected components. The sets of exterior points, neighboring points and interior points can be identified easily. We can also see clearly how the distance contour separates the disconnected components and how well it approximates the shape of the data set. By using this ϵ offset we can avoid finding complicated connections among data points.

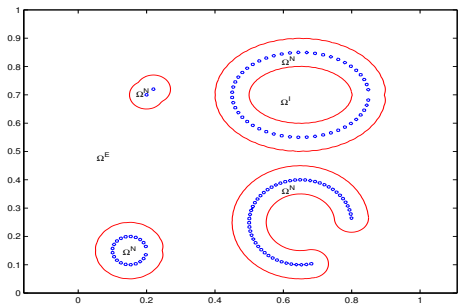


FIGURE 1. distance contours of a data set

Using the classification of the grid points we can dissect all disconnected components and characterize their properties on the scale ϵ. We start with any grid point in $\Omega^N \cup \Omega^I$ and find all grid points in $\Omega^N \cup \Omega^I$ that are connected to it and label them as the first component denoted by Ω_1. If there is no grid point left in $\Omega^N \cup \Omega^I - \Omega_1$, then there is only one connected component, i.e., Ω_1. Otherwise we start with any left point and find all grid points in $\Omega^N \cup \Omega^I - \Omega_1$ that are connected to it as the second connected component denoted by Ω_2. We go on until all grid points in

$\Omega^N \cup \Omega^I$ are marked. Now we have identified all disconnected components represented by the data set on the scale ϵ. By using the above algorithm for identifying all disconnected components, we can also find out the following useful geometric and topological properties of each component on the scale of ϵ:

- The total number of interior grid points in each connected component, i.e., $\Omega_k \cap \Omega^I$, which can be regarded as an approximation of the volume of the component Ω_k. In particular if a component has no interior point then we can say that data points in that component represent an open manifold on the scale of ϵ.
- The total number of neighboring grid points in each connected component, i.e., $\Omega_k \cap \Omega^N$, which can be regarded as an approximation of the Hausdorff dimension (i.e. surface area) of the manifold represented by the data in the k-th component. Also the ratio between the number of interior points and the number of neighboring points approximate the ratio between surface area and the volume, which illustrates the "thickness" of a volumetric object.
- The total number of data points contained in each component (by counting and adding the number of data points in each grid cell in Ω_k) can be used to tell the significance of the cluster of data points. For example, we can use the number of data points in each component to single out and remove those isolated outliers.

If the sampling density varies for different components of the data set, we can first extract any particular component and apply a different scale ϵ on each component for the above analysis. In a more general setting, we can define the scale ϵ as a spatially varying function $\epsilon(\mathbf{x})$ that takes into account local sampling density, uncertainty or statistics of the data.

2.3 Extraction of distance contours and visualization of the data set

After the classification of all grid points, we can extract an appropriate distance contour as an offset to the real shape to visualize the data set. As is shown in figure 1, the distance contour $d(\mathbf{x}) = \epsilon$ which is exactly $\partial\Omega^N$, may be composed of two pieces for data points that form a closed manifold. The first one is the boundary between the set of exterior points, Ω^E, and the set of neighboring points, Ω^N, which we call the exterior distance contour and is denoted by Γ_e^ϵ. The second one is the boundary between the set of interior points, Ω^I, which is not empty if the data set consists of a closed manifold on the scale of ϵ, and the set of neighboring points, Ω^N, which we call the interior distance contour and is denoted by Γ_i^ϵ. No matter what ϵ is, the exterior distance contour Γ_e^ϵ always exists,

assuming that the scale ϵ is resolved by the grid size. Thus we extract the exterior distance contour as an approximate offset of the shape representing the data set. Denote $\underline{d}$ to be the largest distance between points that should be connected and $\overline{d}$ to be the smallest distance between two disconnected components or disjoint parts, we say the sampling density of a data set satisfies the separation condition if $\underline{d} < \overline{d}$. The separation condition is to exclude possible connectivity ambiguity. If the sampling density of the data set satisfies the separation condition, we can choose ϵ as small as possible in the range $(\underline{d}/2, \overline{d}/2)$ so that the exterior distance contour is homeomorphic to the true shape. As the sampling density increases, i.e., $\underline{d} \to 0$, we can also take

$$\underline{d}/2 < \epsilon \to 0, \text{ so that } \|\Gamma - \Gamma_e^\epsilon\| = O(\epsilon) \to 0,$$

where Γ is the C^1 manifold represented by the data. Moreover, using the distance contour as an offset for the true shape can also automatically reduce the noise in the data set to some extent. In the examples shown below in section 2.4, we see that even for large real data sets that are noisy the visualization results look quite good.

To construct and visualize the distance contour, we construct an implicit representation, i.e., we construct a signed distance function to the exterior distance contour as follows. For those grid points in Ω^E (that are outside the exterior distance contour), their distance $\bar{d}(\mathbf{x}_{i,j})$ to the contour is just a shift by ϵ of its distance to the data set, i.e., $\bar{d}(\mathbf{x}_{i,j}) = d(\mathbf{x}_{i,j}) - \epsilon$. We also assign $\bar{d}(\mathbf{x}_{i,j}) = |d(\mathbf{x}_{i,j}) - \epsilon|$ to those boundary points in Ω^N that are immediate neighbors of Ω^E. Now we fix these correct distance values and use them as the initial values to solve the Eikonal equation (1.1) for $\bar{d}(\mathbf{x}_{i,j})$ by the fast sweeping method. Then we negate $\bar{d}(\mathbf{x}_{i,j})$ for those grid points in $\Omega^N \cup \Omega^I$ to get the signed distance function to the exterior distance contour. The whole procedure is again of $O(N)$ complexity for N grid points. Using the same procedure we can also find the signed distance to the interior distance contour. Since we can think of the distance contour we construct as an offset to the real shape represented by the data set, we can use the distance contour as an initial guess and move it closer to the data set to get a better approximation. We will discuss this in section 3

Data coarsening Here we propose a simple data processing procedure that can be used to reduce the amount of data to a prescribed resolution. We lay down a grid according to a prescribed resolution. For each grid cell that contains data points, we compute the weighted mass center of all data points in that grid cell or data points in a neighborhood with a given radius and assign the total weight of those data points to the mass center. For example, the weight can be dependent on the uncertainty of each data point. We can replace the original set of data points by those weighted mass centers on this resolution for visualization or other analysis. Now each grid cell only has one point and is associated with some weight. Computing the mass center is a kind of averaging and can also help to remove noise

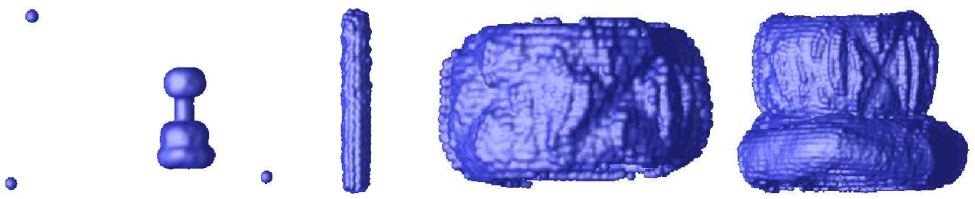
or redundancy in the data to some extent. We demonstrate in one of our numerical examples that the data coarsening process can greatly reduce the total number of data points of the original data set and we see no difference in the visualization result.

2.4 Examples

In this section, we present results and timings of our algorithms on real data sets. In particular we would like to demonstrate (1) the efficiency and quality of using distance contours for visualization of large data sets, (2) dissection, analysis and process of large data sets without surface reconstruction. All our computations are done on a Linux PC with Pentium 600Mhz processor and 1GB memory, which allows a maximum number of grid points around 220^3. The CPU time is measured in second. The timing includes every step except the rendering time using Data Explorer. Table 1.1 shows the number of data points, size of the grid and timing for our examples.

Model	Data points	Grid size	CPU (second)
drill (raw)	50,643	100x69x54	3
drill base	34,106	149x118x151	14
drill cap	12,247	117x74x120	7
drill bit	4,283	24x120x24	0.4
dragon (raw)	1,769,513	149x124x147	49
dragon	1,723,751	311x222x146	93
dragon (scaled)	285,231	311x222x146	94
Buddha	543,652	156x371x156	39
terrain (raw)	100,860	600x118x99	51
terrain	98,725	601x452x29	28

TABLE 1.1. timing table



(a) the raw data set (b) the drill bit (c) the drill cap (d) the drill base

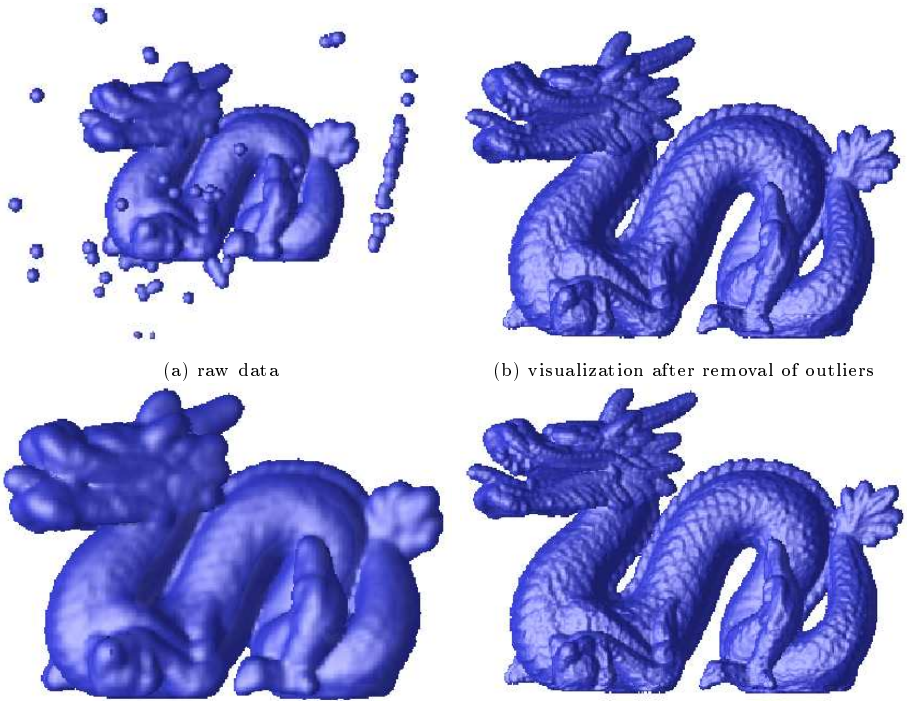
FIGURE 2.

Figure 2 shows the data processing and visualization of a drill. Figure 2(a) shows the visualization of the raw data obtained by a 3D scanner from a few different angles. The data has outliers and disconnected components.

The exterior distance contour $d(\mathbf{x}) = 2h$, where h is the grid size, is used for the visualization and data processing. The whole data set has a total of 50,643 points. The number of data points in each of the three sets of outliers is 2, 4, 1 respectively. There are three disconnected components after the removal of the outliers which are: the drill base which has 34,106 points, the drill cap which has 12,247 data points, and the drill bit which has 4,283 points. We can dissect these three parts and visualize them using distance contours on different grids accordingly in figure 2(b),(c),(d). Figure 3 shows the visualization and processing of the raw data from 3D scanning data for a dragon statue. We did not use those backdrop data for hole filling that is used for the construction at www-graphics.stanford.edu/data/3Dscanrep. The whole data set has 1,769,513 points. Figure 3(a) visualizes the raw data on a 149x124x147 grid using the distance contour $d(x) = h$. On this grid and with scale $\epsilon = 2h$ we can dissect 43 disconnected components, 42 of which correspond to outliers that have 45,518 points all together. After removal of the outliers we visualize the dragon on a 311x222x146 grid (the largest possible on our PC) using an exterior distance contour $d(\mathbf{x}) = h$ in figure 3(b). We visualize the same data set on a coarse grid of size 100x73x49 in figure 3(c), which takes only 32 seconds. We can also rescale the data set after the removal of the outliers to the grid resolution of a 311x222x146 grid, as described in section 2.3. The total number of data points is reduced to 285,231 which is visualized using $d(\mathbf{x}) = h$ in figure 3(d). Almost no difference can be seen from using the original data set in figure 3(b). In figure 4 we show the visualization of a Buddha statue on a 156x371x156 grid from a 3D scanning data set of 543,652 points. The distance contour used is $d(\mathbf{x}) = h$ and the computation takes only 39 seconds. Figure 5 is the visualization of laser radar data for a terrain. The data set is very noisy and there are many bad data points due to occlusions and non-reflections. Moreover, the scale is very different in the horizontal and vertical directions. Figure 5(a) is the visualization of the raw data and shows how bad it is. After removal of the a total of 2,135 bad points, e.g. disconnected outliers, using our procedure, we visualize the data in figure 5(b). Now we can see quite clearly the buildings, the road, bushes and shadows.

3 Construction of implicit surfaces using the level set method

In [ZOMK00],[ZOF01] mathematical formulations and numerical algorithms were developed for surface reconstruction for unorganized data sets using differential geometry and partial differential equations. The level set method and dynamic implicit surfaces were used to provide a general framework for surface modeling, analysis, deformation and many other applica-



(a) raw data (b) visualization after removal of outliers

(c) visualization on a coarse grid (d) visualization using rescaled data

FIGURE 3.

tions. A "weighted" minimal surface model, which takes into account both the surface area and closeness to the data set, was proposed. The variational formulation allowed us to balance them in an optimal way. The reconstructed surface is smoother than piecewise linear, thus the results look good on relatively coarse data sets. In addition, there is a regularization that is adapted to the local sampling density in the spirit of [AB98] and sharp features can be kept if a simple local sampling condition is satisfied. The formulation handles noisy as well as non-uniform data and works in an arbitrary number of dimensions. In [PBZ00] we have recently extended the method to also interpolate data giving the value of the unit normal to the surface at arbitrary points, curves and surface patches. A physically motivated convection model and a very fast tagging algorithm were also developed to give a very good initial approximation to the local minimizer, and thus to our minimal surface reconstruction.

3.1 The Weighted Minimal Surface Model

Let $\mathcal{S}$ denote a general data set which can include data points, curves or pieces of surfaces. Define $d(\mathbf{x}) = dist(\mathbf{x},\mathcal{S})$ to be the distance function to

(a) front (b) diagonal (c) back

FIGURE 4.

$\mathcal{S}$. In [ZOMK00] the following surface energy is defined for the variational formulation:

$$E(\Gamma) = \left[\int_\Gamma d^p(\mathbf{x})ds\right]^{\frac{1}{p}}, \quad 1 \le p \le \infty, \tag{1.3}$$

where Γ 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 $p = \infty$, $E(\Gamma)$ is the value of the distance of the point on Γ most remote from $\mathcal{S}$. For $p < \infty$, The surface energy $E(\Gamma)$ is equivalent to $\int_\Gamma d^p(\mathbf{x})ds$, 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.

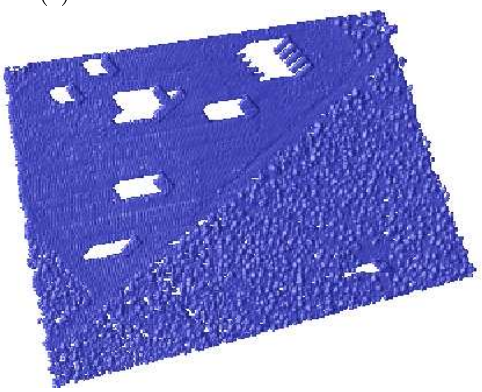
As derived in [ZOMK00] the gradient flow of the energy functional (1.3) is

$$\frac{d\Gamma}{dt} = -\left[\int_\Gamma d^p(\mathbf{x})ds\right]^{\frac{1}{p}-1} d^{p-1}(\mathbf{x})\left[\nabla d(\mathbf{x})\cdot\mathbf{n} + \frac{1}{p}d(\mathbf{x})\kappa\right]\mathbf{n}, \tag{1.4}$$

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

$$d^{p-1}(\mathbf{x})\left[\nabla d(\mathbf{x})\cdot\mathbf{n} + \frac{1}{p}d(\mathbf{x})\kappa\right] = 0, \tag{1.5}$$

(a) raw data with noise and outliers



(b) visualization after removal of outliers

FIGURE 5.

where $\mathbf{n}$ is the unit outward normal and κ is the mean curvature. We see a balance between the attraction $\nabla d(\mathbf{x}) \cdot \mathbf{n}$ and the surface tension $d(\mathbf{x})\kappa$ in the equations above. Moreover the nonlinear regularization due to surface tension has a desirable scaling $d(\mathbf{x})$. 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(1.5) above, since $|\nabla d \cdot \mathbf{n}| \leq 1$, we have a local sampling density condition similar to the one proposed in [ABE98], which says sampling densities should resolve fine features locally. To construct the minimal surface we used a continuous deformation in [ZOMK00]. We start with an initial surface that encloses all data and follow the gradient flow (1.4). The parameter p affects the flexibility of the membrane to some extent. When $p = 1$, the surface energy defined in (1.3) has the dimension of volume and the gradient flow (1.4) is scale invariant i.e., dimensionless. In practice we find that $p = 1$ or 2 (similar to a least squares formulation) are good choices. More details can be found in [ZOMK00].

In two dimensions, it was shown in [ZOMK00] 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.

3.2 The Convection Model

The evolution equation (1.4) 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 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. We convect a flexible surface Γ in the potential field of the distance function $d(\mathbf{x})$ to the data set $\mathcal{S}$,

$$\frac{d\Gamma(t)}{dt} = -\nabla d(\mathbf{x}). \tag{1.6}$$

The velocity field at any point, except those equal distance points, is a unit vector pointing toward its closest point in $\mathcal{S}$. The set of equal distance points has measure zero. Hence points on a curve or a surface, except those equal distance points, are attracted by their closest points in the data set (see Fig. 6(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. Since the convection equation is a first order linear differential equation, we can solve it using a time step $\Delta t = O(h)$. The convection model very often results in a good surface reconstruction by itself.

3.3 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 dynamic deformation of implicit surfaces. The level set method is based on a

continuous formulation using PDEs and allows one to deform an implicit surface 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. [AS95, PMO+99]. Two key steps for the level set method are: (1) Embed the surface: we represent a surface Γ as the zero isocontour of a scalar (level set) function $\phi(\mathbf{x})$, i.e. $\Gamma = \{\mathbf{x} : \phi(\mathbf{x}) = 0\}$. Geometric properties of the surface Γ 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., $\Gamma(t) = \{\mathbf{x} : \phi(\mathbf{x}, t) = 0\}$,

For geometric motions, i.e. where the motion law (velocity) depends only on the geometry of the moving surface, the most natural way is to apply the same motion law for all level sets of the level set function, which will result in a morphological PDE [AGLM93]. For example, the gradient flow (1.4) is a geometric motion. If we use $p = 1$ and extend the geometric motion to all level sets, the gradient flow in level set formulation becomes

$$\frac{\partial \phi}{\partial t} = |\nabla \phi| \nabla \cdot \left[d \frac{\nabla \phi}{|\nabla \phi|} \right] = |\nabla \phi| \left[\nabla d \cdot \frac{\nabla \phi}{|\nabla \phi|} + d \nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} \right], \tag{1.7}$$

For the convection model (1.6), since the velocity field $-\nabla d(\mathbf{x})$ is defined everywhere, we can naturally extend the convection to all level sets of $\phi(\mathbf{x}, t)$ to obtain

$$\frac{\partial \phi}{\partial t} = \nabla d(\mathbf{x}) \cdot \nabla \phi. \tag{1.8}$$

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. We use a numerical procedure called reinitialization, see e.g. [PMO+99, SSO94], 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.

3.4 Finding a good initial guess

We can use an arbitrary initial surface that contains the data set, such as a rectangular bounding box, to begin with. However, a good initial surface is important for the efficiency and convergence of our PDE based method. On a rectangular grid, we view an implicit surface as an interface that separates

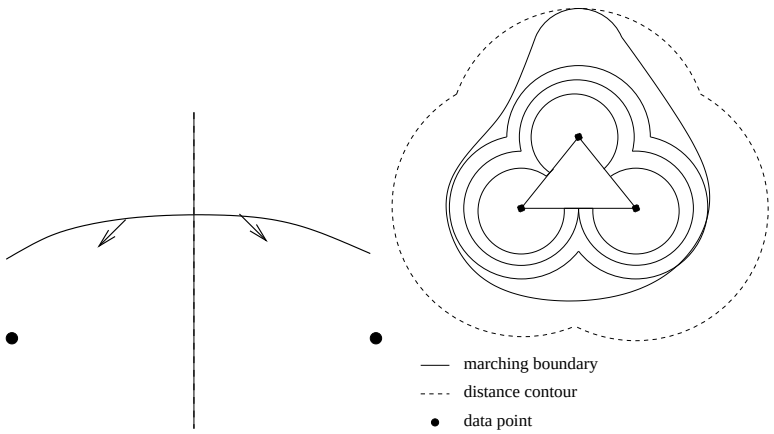
the exterior grid points from the interior grid points. An extremely efficient tagging algorithm was proposed in [ZOF01] that tries to identify as many correct exterior grid points as possible and hence provides a good initial implicit surface. As always, we start from any initial exterior region that 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 grid size, 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 each interior grid point is visited 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.

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. We can use either a bounding box of data set or an outer contour of the distance function, $d(\mathbf{x}) = \epsilon$, as the initial temporary boundary to start the fast tagging algorithm.

Remark: Since the maximum distance for the boundary heap is strictly decreasing, the algorithm converges and 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.

Figure 6(b) illustrates how the fast tagging algorithm works. The furthest point on the initial temporary boundary is tangent to a distance contour and does not have an interior neighbor that is farther away. The furthest



(a) the attraction of a piece of curve by two points (b) the tagging algorithm

FIGURE 6.

point is tagged as an exterior point and the boundary moves inward at that point. Another point on the temporary boundary becomes the furthest point and hence the whole temporary boundary moves inward. Gradually the temporary boundary follows distance contours and moves closer and closer to the data set until the distance contours begin to break at the equal distance point. We see that the temporary boundary at the breaking point of the distance contour has neighboring interior points that have a larger distance. So this temporary boundary point will be tagged as a final boundary point 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.

3.5 Multiresolution and Efficient Storage

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(b) 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.

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 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 very quickly.

3.6 Numerical Implementations and Examples

There are three steps in our implicit surface reconstruction algorithm. First, we compute the distance function to an arbitrary data set on a rectangular grid. Second, we find a good initial surface. Third, we start the continuous deformation following either the gradient flow (1.4) or the convection (1.6) using the corresponding level set formulation (1.7) or (1.8). Our numerical implementations are based on standard algorithms for the level set method. Details can be found in, for example, [PMO+99, ZCMS96, ZOMK00]. The convection model is simple but the reconstructed surface is close to a piecewise linear approximation. In contrast the gradient flow is more computationally expensive but reconstructs a smoother weighted minimal surface. In particular, the gradient flow can be used as a smoothing process for implicit surfaces. In most of our applications, less than one hundred time steps in total are enough for our continuous deformation to converge.

Figure 7 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 8 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 reconstruction we only provide the unsigned distance function to the curves which can be viewed as an extreme case of non-uniform data. After the initial reconstruction using a distance contour and/or fast tagging algorithm, 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 other reconstructions. Figure 9 shows the reconstruction of

the Happy Buddha. Figure 9(a) is the reconstruction on a fine grid. Figure 9(b) shows the reconstruction on a coarse grid.

Model	Data points	Grid size	CPU (minute)
Rat brain	1506	80x77x79	3
Buddha	543652	146x350x146	68
Buddha	543652	63x150x64	7

TABLE 1.2. timing table

Acknowledgments. Data sets for the drill, the dragon and Buddha are from The Stanford 3D Scanning Repository. The laser radar data is from Naval Air Warfare Center at China Lake.

4 References

[AB98] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *14th ACM Symposium on Computational Geometry*, 1998.

[ABE98] 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.

[ABK98] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH'98*, pages 415–421, 1998.

[AGLM93] 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.

[AS95] D. Adalsteinsson and J.A. Sethian. A fast level set method for propagating interfaces. *J. Comput. Phys.*, 118:269–277, 1995.

[BBB+97] 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.

[BBX95] 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.

[BC00] J.D. Boissonnat and F. Cazals. Smooth shape reconstruction via natural neighbor interpolation of distance functions. *ACM Symposium on Computational Geometry*, 2000.

[BCOS01] M. Bertalmio, L.-T. Cheng, S. Osher, and G. Sapiro. Variational problems and partial differential equations on implicit surfaces: The framework and examples in image processing and pattern formation. *J. Comput. Phys.*, 174(2):759–780, 2001.

[Boi84] J.D. Boissonnat. Geometric structures for three dimensional shape reconstruction. *ACM Trans. Graphics 3*, pages 266–286, 1984.

[BW90] M. Bloor and M. Wilson. Using partial differential equations to generate free-from surface. *Computer Aided Des.*, (22):257–266, 1990.

[CBC+01] J. C. Carr, R. K. Beatson, J.B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3d objects with radial basis functions. *proceedings, SIGGRAPH 2001*, 2001.

[CBMO02] L.-T. Cheng, P. Burchard, B. Merriman, and S. Osher. Motion of Curves Constrained on Surfaces Using a Level Set Approach. *J. Comput. Phys.*, 175(2):604–644, 2002.

[CG91] G. Celniker and D. Gossard. Reformable curve and surface finite element for free-form shape design. *Computer Graphics*, (25):257–266, 1991.

[CL96] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH'96 Proceedings*, pages 303–312, 1996.

[Dan80] P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[DCG98] M. Desbrun and M.P. Cani-Gasceul. Active implicit surfaces. *Graphics Interface*, 1998.

[Ede98] 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.

[EM94] H. Edelsbrunner and E. P. Mücke. Three dimensional α shapes. *ACM Trans. Graphics 13*, pages 43–72, 1994.

[FF01] N. Foster and R. Fedkiw. Practical animation of liquids. *SIGGRAPH*, 2001.

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

[HSIW98] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Implicit surface - based geometric fusion. *Comput. Vision and Image Understanding*, 69:273–291, 1998.

[MT93] T. McInerney and D. Terzopoulos. A finite element model for 3D shape reconstruction and nonrigid motion tracking. *Proc. IEEE 4th Int. Conf. on Comp. Vision*, pages 518–523, 1993.

[Mur91] S. Muraki. Volumetric shape description of range data using "blobby model". In *Computer Graphics (Proc. SIGGRAPH)*, volume 25, pages 227–235, July 1991.

[NB93] P. Ning and J. Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics and Applications*, 13(6):33–41, November 1993.

[OS88] 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.

[PASS95] 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.

[PBZ00] S. Osher P. Burchard, S. Chen and H.K. Zhao. Surface reconstruction from normals. Technical report, *Level Set Systems* Report, 2000.

[PMO+99] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A pde-based fast local level set method. *J. Comput. Phys.*, 155(2):410–438, 1999.

[PT97] L. Piegl and W. Tiller. *The NURBS book*. Berlin, Germany: Springer-Verlag, 2nd edition edition, 1997.

[Rog00] D.F. Rogers. *An Introduction to NURBS*. Morgan Kaufmann, 2000.

[SPOK95] 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.

[SSO94] 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.

[TBC+02] Richard Tsai, Paul Burchard, Li-Tien Cheng, Stanley Osher, and Guillermo Sapiro. Dynamic visibility in a level set based implicit frame work. *UCLA CAM report 02-06*, 2002.

[TCOZ01] Y.R. Tsai, L.-T Cheng, S. Osher, and H.K. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *UCLA CAM report 01-27, submitted to SINUM*, 2001.

[TO99] G. Turk and J. ÒBrien. Shape transformation using variational implicit functions. *SIGGRAPH99*, pages 335–342, August 1999.

[WGG99] 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.

[WMW86] B. Wyvill, C. McPheeters, and G. Wyvill. Data structure for soft objects. *The Visual Computer*, 2(4):227–234, 1986.

[ZCMS96] 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.

[Zha02a] H.K. Zhao. Analysis and visualization of large set of unorganized data points using the distance function. *preprint*, 2002.

[Zha02b] H.K. Zhao. Fast sweeping method for eikonal equations i: Distance function. *preprint, submitted to SIUNM*, 2002.

[ZOF01] H. Zhao, S. Osher, and R. Fedkiw. Fast surface reconstruction and deformation using the level set method. *Proceedings of IEEE Workshop on Variational and Level Set Methods in Computer Vision, Vancouver*, July, 2001.

[ZOMK00] 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.

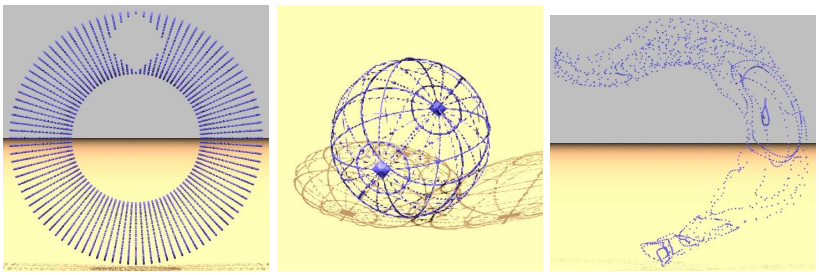


FIGURE 7. initial data

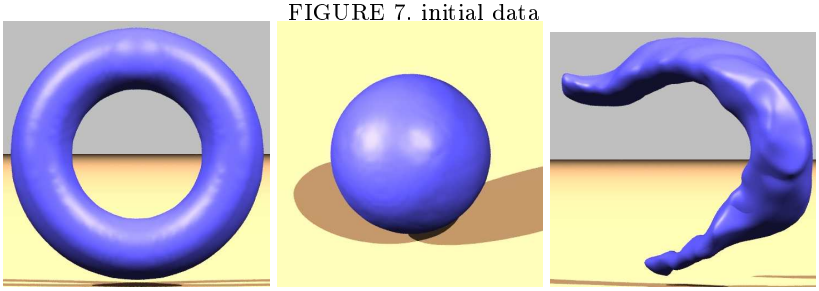


FIGURE 8. final reconstruction

(a) reconstruction on a fine grid (b) reconstruction on a coarse grid

FIGURE 9. reconstruction of the Happy Buddha