ANALYSIS AND VISUALIZATION OF LARGE SET OF UNORGANIZED DATA POINTS USING THE DISTANCE FUNCTION

HONGKAI ZHAO*

Abstract. An efficient algorithm to analyze, process and visualize a large set of unorganized data points is devleoped. The idea is based on the use of distance function and appropriate distance contours on a prescribed scale or resolution. In particular, an appropriate choice of the distance contour is used for quick visualization of the data set on different scale and resolution. Distance contours are also used to characterize and quantify topological and geometrical properties of the data set such as finding all disconnected components, estimating the Hausdorff dimension of the data set, etc. Our numerical algorithms are based on fixed rectangular grids according to a given scale or resolution. Efficient algorithms are developed for computing the distance function, extracting distance contours, characterizing and processing the data set. The whole procedure has a complexity of order $O(N+M)$, where M is the number of data points and N is the number of grid points, and works in any number of dimensions.

1. Introduction. Efficient algorithms for analysis, processing and visualization of large data sets of unorganized points are very important in computer visualization, data mining, range data analysis, biomedical imaging, and many other applications. Most of the current approaches are based on the idea of interpolation of the data set, for instances, triangulated surfaces using Voronoi diagram and Delauney triangulation in computation geometry [1, 2, 5, 13, 14, 26, 27], splines such as NURBS in computer aided design [7, 19, 20], implicit surfaces using distance functions [6, 9, 11, 16, 15] or a combination of basis functions [4, 18, 22, 10], and other interpolation techniques. In many applications the data set is large and noisy, has complicated topology, and may have multiple Hausdorff dimensions. These interpolation based reconstructions or representations are too expensive or difficult to use in practice. Simple and efficient approximate numerical procedures are desirable. The key observation is that for most existing approaches, the emphasis is on the exploration of the exact relations among all data points and interpolation accuracy which often results in high computational cost and difficulties. In many situations we only need a representation that are good enough for approximate analysis and visualization. One can compromise between efficiency and interpolation accuracy.

Here we propose an efficient data analysis and visualization procedure based on distance function, distance contours and connectivity relation that can

- find all disconnected components on a given scale,
- provide quick visualization on different resolution and scale,
- extract some topological or geometric information of the data set,

The basic idea is that the distance contours, which are level sets of the distance function to a data set, can be viewed as approximate offsets of the shape represented by the data set. We use appropriate distance contours to visualize the data, dissect disconnected components and characterize some implortant topological and geometric information of the data set. Efficient numerical algorithms based on rectangular grids are developed to compute the distance function to an arbitrary data set, extract appropriate distance contours, identify and characterize each disconnected component. The grid resolution and the choice of the distance contour depend on the prescribed scale or sampling density of the data set. In particular a fast sweeping method is used to solve the Eikonal equation, $|\nabla d(\boldsymbol{x})| = 1$, for the distance function $d(\boldsymbol{x})$. A simple and efficient connectivity checking algorithm that visits every grid point once is used to identify each disconnected component of the data set. 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.

Distance contour is equivalent to the notion of α diagram in the definition of α shape [14] or the union of ball approximation in [3]. However, these computation geometry algorithms are based on Voronoi diagram and Delauney triangulation and construct triangulated surfaces. Our method is grid based and use implicit surface representation. Moreover, we can easily extract multiple distance contours and do data analysis on different scale and resolution once we compute the distance values on all grid points. Shape offsets via level sets for image contours are used in [17].

In section 2 we briefly review a few basic facts about the distance function and explain the fast sweeping

*Department of Mathematics, University of California, Irvine, CA 92697-3875. e-mail: zhao@math.uci.edu. The research is partially supported by Sloan Fundation, NSF DMS0112416, ONR grant N00014-02-1-0090 and DARPA grant N00014-02-1-0603

method for computing distance function. We then discuss the dissection and characterization of disconnected components of the data set and extraction of appropriate distance contours for visualization in section 3. Finally we present numerical results using real data sets in section 4.

This paper is motivated by the earlier work done in [25, 24]. There we used the distance function, the variational level set method and a fast tagging algorithm to construct implicit surfaces for unorganized data points. Here we use distance function directly for the analysis, processing and visualization of a data set.

2. The Distance Function and the Fast Sweeping Method. The distance function $d(\boldsymbol{x}) = dist(\boldsymbol{x}, \mathcal{S})$ is one of the most important and intrinsic information for an arbitrary data set $\mathcal{S}$. Given the distance function to a data set, we 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 $d(\boldsymbol{x}) = \epsilon$, with the right choice of ϵ, is an approximate ϵ-offset of the manifold represented by the data set. Thus it can be used to approximate and visualize the shape of the data set.
2. the ϵ distance contour can be used to dissect and analyze disconnected components of the data set on the scale ϵ.
3. the ϵ-shell$= \{\boldsymbol{x} : d(\boldsymbol{x}) < \epsilon\}$ can be viewed as an approximate ϵ-covering of the manifold represented by the data set. It can be used to approximate the Hausdorff dimension of each disconnected component of the data set

Our numerical implementations are based on fixed rectangular grids. The grid resolution depends on the data sampling density and the prescribed scale ϵ. We first compute the distance function to the data set on the grids and then construct the ϵ distance contour, dissect and characterize disconnected components. Hence one of the most crucial component of our numerical algorithms is the computation of the distance function to an arbitrary data set on a rectangular grid. Here we use the fast sweeping method that was used in [25] and was further analyzed in [23]. The distance function $d(\boldsymbol{x})$ to an arbitrary data set $\mathcal{S}$ solves the following Eikonal equation:

$$|\nabla d(\boldsymbol{x})| = 1, \qquad d(\boldsymbol{x}) = 0, \ \boldsymbol{x} \in \mathcal{S}. \tag{2.1}$$

The characteristics of this simple Hamilton-Jacobi equation are straight lines that radiates from the data set $\mathcal{S}$. Information propagates along characteristics and the solution is strictly increasing along characteristics, which reveals the causality: the solution at a grid point should be determined only by its neighboring grid points that have smaller values. The fast sweeping method uses an upwind differencing to enforce this causlity and combines with Gauss-Seidel iterations of different sweeping ordering to solve the discretized system. A similar algorithm was proposed in [8]. For simplicity we describe the fast sweeping method in two dimensions. We use $\boldsymbol{x}_{i,j}$ to denote a grid point in the computational domain, use h to denote the grid size and $u^h_{i,j}$ to denote the numerical solution at $\boldsymbol{x}_{i,j}$. Extension to higher dimensions is straightforward.

The Fast Sweeping Method:

Discretization: We use the following upwind difference scheme to discretize the partial differential equation (2.1) at interior grid points

$$\begin{array}{l} [(u^h_{i,j} - u^h_{xmin})^+]^2 + [(u^h_{i,j} - u^h_{ymin})^+]^2 = h^2 \\ i = 2, \dots, I-1, \ j = 2, \dots, J-1, \end{array} \tag{2.2}$$

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 $(x)^+ = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases}$. One sided difference is used at the boundary of the computation domain. For example, at a point $\boldsymbol{x}_{1,j}$ on the left boundary, a one sided difference is used in the x direction,

$$[(u^h_{1,j} - u^h_{2,j})^+]^2 + [(u^h_{1,j} - u^h_{ymin})^+]^2 = h^2.$$

At the four corner points, a one sided difference scheme is used in both directions.

Initialization: To enforce the boundary condition, $u(\boldsymbol{x}) = 0, \boldsymbol{x} \in \mathcal{S}$, we assign exact distance 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.

Gauss-Seidel iterations with alternating sweeping ordering: At each grid $\boldsymbol{x}_{i,j}$, whose value is not fixed in the above initialization, compute the solution, denoted by $\overline{u}$, of (2.2) from the current values 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,

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

At each point the following equation

$$[(x-a)^+]^2 + [(x-b)^+]^2 = h^2, \tag{2.3}$$

has to be solved, where $a = u^h_{xmin}, b = u^h_{ymin}$. The unique solution is

$$\overline{x} = \begin{cases} \min(a,b) + h & |a-b| \geq h \\ \frac{a+b+\sqrt{h^2-(a-b)^2}}{2} & |a-b| < h \end{cases} \tag{2.4}$$

In n dimensions the unique solution $\overline{x}$ to

$$[(x-a_1)^+]^2 + [(x-a_2)^+]^2 + \cdots + [(x-a_n)^+]^2 = h^2 \tag{2.5}$$

can be found in the following systematic way. First we order a_k's in increasing order. Without loss of generality, assume $a_1 \leq a_2 \leq \ldots \leq a_n$ and define $a_{n+1} = \infty$. There is an integer $1 \leq p \leq n$, such that $\overline{x}$ is the unique solution that satisfies

$$(x-a_1)^2 + (x-a_2)^2 + \cdots + (x-a_p)^2 = h^2 \quad \text{and} \quad a_p < \overline{x} \leq a_{p+1}, \tag{2.6}$$

We find $\overline{x}$ and p in the following recursive way. Start with $p = 1$, if $\tilde{x} = a_1 + h \leq a_2$, then $\overline{x} = \tilde{x}$. Otherwise find the unique solution $\tilde{x} > a_2$ that satisfies

$$(x-a_1)^2 + (x-a_2)^2 = h^2$$

If $\tilde{x} \leq a_3$, then $\overline{x} = \tilde{x}$. Otherwise repeat the procedure until we find p and $\overline{x}$ that satisfy (2.6).

In practice it may be desirable to restrict the computation to a neighborhood of the data set $\mathcal{S}$. For example, if we want to restrict the computation in the neighborhood where distance is less than $\overline{d}$, we can use the following simple cutoff criterion: in the Gauss-Seidel iteration we update the solution at a grid point $\boldsymbol{x}_{i,j}$ only if at least one of its neighbors has a value smaller than $\overline{d}$, i.e., if $\min(u^h_{xmin}, u^h_{ymin}) < \overline{d}$.

For the initialization step we first assign a large value to all grid points. The only requirement for the large value is that it is larger than the maximum possible distance value in the domain. Next we go through each data point and update the distance values of its neighboring grid points. For example, for each data point we find the grid cell that contains it and then compute the exact distance value of vertices of the grid cell to the data point. We replace the current distance values of the vertices whenever the distance to this data point provides a smaller value. Of course we can include more neighboring grid points for which the exact distance values are computed. After going through all data points, we have computed exact distance values at those grid points in a neighborhood of the data set. These values will be fixed in all later computations. The whole initialization procedure is of complexity $O(M+N)$ for N grid points and M data points. We can deal with more general data sets as long as the distance values on grid points neighboring the set are provided or computed initially. At the boundary we always use one sided difference to enforce the propagation of information to be from inside to outside. This is equivalent to the fact that there are no data points outside the domain.

It is shown in [23] that 2^n sweeps are enough for computing distance function in n dimensions. So the total complexity of the fast sweeping method to compute the distance function to a data set of M points on a rectangular grid of N grid points is $O(M+N)$.

As a matter of fact any efficient numerical algorithms for computing distance function on a rectangular grid can be applied here. As a remark, the differences between this algorithm and the well known Danielsson's distance mapping algorithm [12] are: (i) our data points are not necessary grid points. That is why we need an initialization procedure; (ii) our algorithm is based on partial differential equation and can be applied to more general data sets or equations where $d(\boldsymbol{x})$ is not a Euclidean distance. For example, the data set does not have to be discrete points. It might consist of manifolds of any dimensions, such as curves or surface patches. Also the right hand side of the Eikonal equation (2.1) can be a given function in $\boldsymbol{x}$, i.e., a weighted distance function according to some metric. In [21], the fast sweeping algorithm is applied to even more general Hamilton-Jacobi equations that can compute the distance function on a manifold. The fast sweeping method for computing distance function is versatile and efficient and is used extensively in our numerical algorithms.

3. Dissection of Disconnected Components and Construction of Distance Contours.

3.1. A simple classification of grid points. 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 to each disconnected component of the data set. 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 notations 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 $\boldsymbol{x}_{i,j}$.

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

We define the connected neighbors of a grid point $\boldsymbol{x}_{i,j}$ to be $\boldsymbol{x}_{i\pm1,j}, \boldsymbol{x}_{i,j\pm1}$ 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 connected neighbors is not in Ω. To classify all grid points to one of the above three sets, we only have to go through every grid point once, i.e., the process is of $O(N)$ complexity for N grid points. 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 out the boundary of the intermediate set of 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. Here is the expansion algorithm at each intermediate step. Given an intermediate set of exterior points $\Omega_t^E \subset \Omega^E$, whose points have been classified already. At each point of the boundary $\partial\Omega_t^E$, we go through its connected neighbors that have not been marked yet and perform the following tasks:

- mark all its neighbors that have a distance value greater than ϵ as exterior points and put it into Ω_t^E,
- mark all its neighbors that have a distance value less or equal than ϵ as neighboring points as well as the boundary points between Ω^E and Ω^N.

Those newly added exterior points form the new boundary for the expanding set Ω_t^E and will be used in the next round of expansion. The expansion stops when all unmarked connected neighbors of the expanding boundary has a distance value less than or equal to ϵ, i.e., 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 later. For those unmarked grid points, we easily identify those that have a distance value less than ϵ and mark them as neighboring points. The remaining unmarked points are the interior points. With careful bookkeeping and marking each grid point is visited and checked once to finish the classification process. We have the set of exterior points Ω^E, the set of neighboring points Ω^N, and the set of interior points Ω^I (maybe empty) as well as the boundary between Ω^E and Ω^N and the boundary between Ω^I and Ω^N in $O(N)$ operations.

One of the key issues in the classification of all grid points is the choice of the scale ϵ. Assume that the grid size h resolves ϵ, typically $h \leq \epsilon/3$ is enough in three dimensions, let us see how the three sets change with ϵ. When ϵ is small compared to the shortest distance between any two data points,then the set of interior points is empty, the set of neighboring points is the union of isolated small balls around each data point, and all the remaining points are exterior points. When we increase the scale, those isolated balls are going to expand and start to merge. At some stage, if the union of balls forms a closed shell to include some region, then all three sets are non-empty. 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 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, is homeomorphism to the real manifold and can be regarded as an approximate ϵ offset by moving the manifold outward. However, 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 is not an open manifold. Instead it looks like a thin shell enclosing the true manifold and can still be used as a good approximation in visualization as is described below. All the above situations are demonstrated in figure 3.1, in which the red curves correspond to a distance contour of the distance function to a data set represented by the blue dots. The data set contains several disconnected components. We can see clearly how the distance contour separates the disconnected components and how well it approximates the shape of the data set. The sets of exterior points, neighboring points and interior points can be seen clearly.

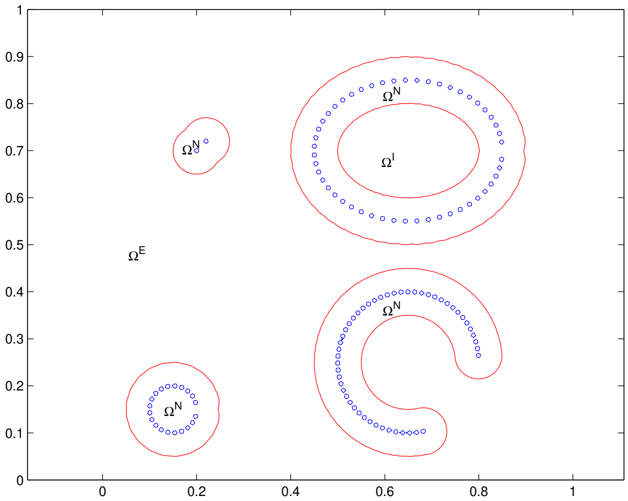


FIG. 3.1. *distance contours of a data set*

3.2. Dissection and characterization of disconnected components of the data set. Using the classification of the grid points we can dissect all disconnected components and characterize their properties on the scale ϵ. We use the simple checking and marking algorithm as the classification algorithm mentioned above to find all disconnected sets of grid points in $\Omega^N \cup \Omega^I$ as follows. 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. During 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 represents 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. The whole checking and marking procedure can be done in an efficient expanding fashion as before. The method visits every grid point just once and works the same in any number of dimensions. In a more general setting, we can define the scale ϵ as a spatially varying function $\epsilon(\boldsymbol{x})$ that takes into account local sampling density, uncertainty or statistics of the data, which will be explored in future study.

3.3. 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 shape of the data set for visualization. As is shown in figure 3.1 the distance contour $d(\boldsymbol{x}) = \epsilon$, 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. We call it 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 represents a closed manifold on the scale of ϵ, and the set of neighboring points Ω^N. We call it 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 represented by the data set. Denote $\underline{d}$ to be the largest distance between any two points that should be connected and $\overline{d}$ to be the smallest distance between two disconnected components or two disjoint parts, we say the sampling density of a data set satisfies the separation condition if $\underline{d} < \overline{d}$. The separation condition can exclude connectivity ambiguity. If the sampling density of the data set satisfies the separation condition, we can choose any ϵ in the range $(\underline{d}/2, \overline{d}/2)$ so that the exterior distance contour is homeomorphism to the true shape. As the sampling density increases, i.e., $\underline{d} \to 0$, we can take

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

where Γ is the C^1 mainifold 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 in section 4, we see that even for large real data sets that are quite noisy the visualization results still 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 $\boldsymbol{x}_{i,j} \in \Omega^E$ their distance to the contour Γ_e^ϵ is just a shift by ϵ of its distance to the data set $\mathcal{S}$, i.e., $d(\boldsymbol{x}_{i,j}, \Gamma_e^\epsilon) = d(\boldsymbol{x}_{i,j}, \mathcal{S}) - \epsilon$. We also assign $|d(\boldsymbol{x}_{i,j}, \mathcal{S}) - \epsilon|$ to those grid points $\boldsymbol{x}_{i,j} \in \Omega^N$ that form the boundary between Ω^N and Ω^E, as is described in section 3.1. Now we have correct distance values at grid points in a neighborhood of Γ_e^ϵ and in Ω^E. We fix these values and use the fast sweeping method to compute the distance to the contour Γ_e^ϵ at all other grid points. Then we negate $d(\boldsymbol{x}_{i,j}, \Gamma_e^\epsilon)$ for those grid points $\boldsymbol{x}_{i,j} \in \Omega^N \cup \Omega^I$ 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 exactly the same procedure we can find the signed distance to the interior distance contour if needed.

Since we can think of the distance contour as an offset to the true 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. The simplest way is just to move the extracted exterior distance contour inward normal to itself by ϵ (or a little bit less in practice). As another benefit, small osscillations on the distance contour can be smoothed out due to the motion normal to itself and numerical viscosity. Hence by using distance contour and moving it inward we can reduce the effect of noise in the data as well as get a smoother surface. More refined models, such as the convection model proposed in [24] and the minimal surface model proposed in [25], can be applied directly to the extracted distance contour for even better results. Both of those methods are based on implicit surfaces. Efficient storage and reconstruction of implicit surfaces using the distance function is discussed in [24].

3.4. Data coarsening. Here we also 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 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 specific resolution for visualization or other analysis. Now each grid cell has only one point and is associated with a certain 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.

4. Numerical results. In this section, we present results and timing of our algorithms on real data sets. In particular we would like to demonstrate (1) the efficiency and quality of using distance contours for the visualization of large data sets, (2) dissection, analysis and processing of large data sets without surface reconstruction. All our computations are done on a Laptop PC with Pentium 600Mhz processor and 1GB memory, which allows a maximum grid size of about 220^3. The CPU time is measured in second. The timing includes computing the distance function, data processing and construction of distance contours but does not include rendering time using Data Explorer. Table 4.1 shows the number of data points, grid size and timing for our examples. Data sets for the drill, the dragon and Buddha are from The Stanford 3D Scanning Repository and data sets for the skull and skeleton are from Georgia Institute of Technology's Large Geometric Models Archive. The laser radar data is from Naval Air Warfare Center at China Lake.

In our computations we first find a rectangular bounding box for the data set and generate a uniform grid according to a certain resolution. We then compute the distance function and perform our data analysis and visualization on this grid. Figure 5.1 shows the data processing and visualization of a drill. Figure 5.1(a) shows the visualization of the raw data obtained by a 3D scanner from a few different views. The data set has a total of 50,643 points and has noise and outliers. We first dissect the data set into disconnected components and define a disconnected component to be an outlier if the total number of data points in that component is fewer than a given threshhold. For the drill data we first use a 100x69x54 grid and the exterior distance contour $d(\boldsymbol{x}) = 2h$, where h is the grid size, for the visualization and data processing. The whole process takes 3 seconds. On this scale the whole data set can be decomposed into six disconnected components. 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 dissect these three parts and visualize them using distance contours on finer grids in figure 5.1(b),(c),(d). The grid size and timing are shown in table 4.1. The drill looks quite smooth on the first grid, which is relatively coarse on the drill part since the bounding box has to include the outliers. The data looks quite noisy on a finer scale.

Figure 5.2 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 5.2(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 Laptop) using exterior distance $d(\boldsymbol{x}) = h$ in figure 5.2(b). We visualize the same data set on a coarse grid of size 100x73x49 in figure 5.2(c). It takes only 32 seconds. We can also rescale the data set after the removal of the outliers to the resolution of a 311x222x146 grid, as is described in section 3.4. The total

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
skull	256,343	155x300x219	72
skeleton	875,549	224x330x143	81

TABLE 4.1
timing table

number of data points is reduced to 285,231 which is visualized using $d(\boldsymbol{x}) = h$ in figure 5.2(d). Almost no difference from using the original data set in figure 5.2(b) can be seen. The whole rescaling and visualization takes 94 seconds. In figure 5.3 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 $d(\boldsymbol{x}) = h$ and it takes only 39 seconds. Figure 5.4 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 5.4(a) is the visualization of the raw data and shows how bad it is. After the removal of a total of 2,135 bad points, e.g., outliers, we visualize the data in figure 5.4(b). Now we can see quite clearly the buildings, roads, bushes and shadows. The timing for the processing of raw data and visualization of the processed data is shown in table 4.1. Figure 5.5 is the visualization of a skeleton decomposed into two parts, the skull and the body. Our construction is very fast and yet is comparable to the reconstruction at the Large Geometric Models Archive.

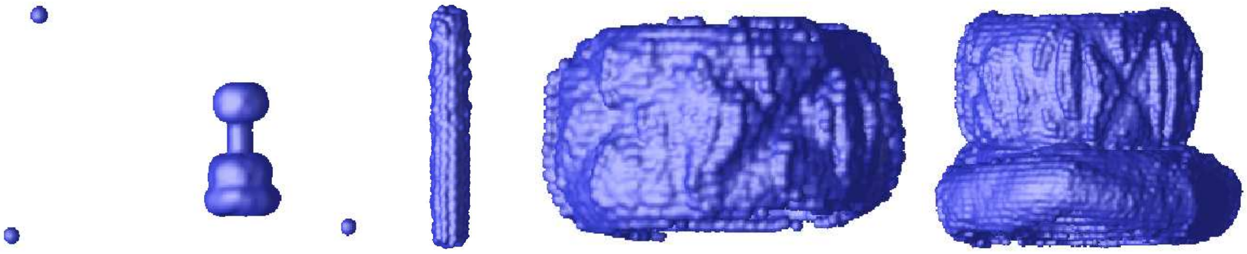
5. Conclusions. In this paper, efficient algorithms for the analysis and visualization of large sets of unorganized data points are developed. The basic idea is to use distance function and appropriate distance contours to dissect, characterize and visualize the data set on different scales. The numerical procedure is based on fixed rectangular grids and implicit representations. The algorithm is very simple and extends to general data sets in any number of dimensions.

Acknowledgment: The author would like to thank Dr. Steve Marschner in the Computer Science Department at Stanford University for the help of acquiring the raw data at The Stanford 3D Scanning Repository and to thank Dr. Alan Van Nevel at Naval Air Warfare Center at China Lake for providing the laser radar data. The author would also like to thank Mr. Chohong Min for suggesting using local distance function in the neighborhood of the data set.

REFERENCES

[1] 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.
[2] N. Amenta, M. Bern, and M. Kamvysselis. A new Voronoi-based surface reconstruction algorithm. *Proc. SIGGRAPH'98*, pages 415–421, 1998.
[3] N. Amenta and R. Kolluri. Accurate and efficient unions of balls. *ACM Symposium on Computational Geometry*, 2000.
[4] 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.
[5] J.D. Boissonnat. Geometric structures for three dimensional shape reconstruction. *ACM Trans. Graphics 3*, pages 266–286, 1984.
[6] J.D. Boissonnat and F. Cazals. Smooth shape reconstruction via natural neighbor interpolation of distance functions. *ACM Symposium on Computational Geometry*, 2000.
[7] C. De Boor. *A practical guide to splines.* Springer-Verlag, New York, 1978.

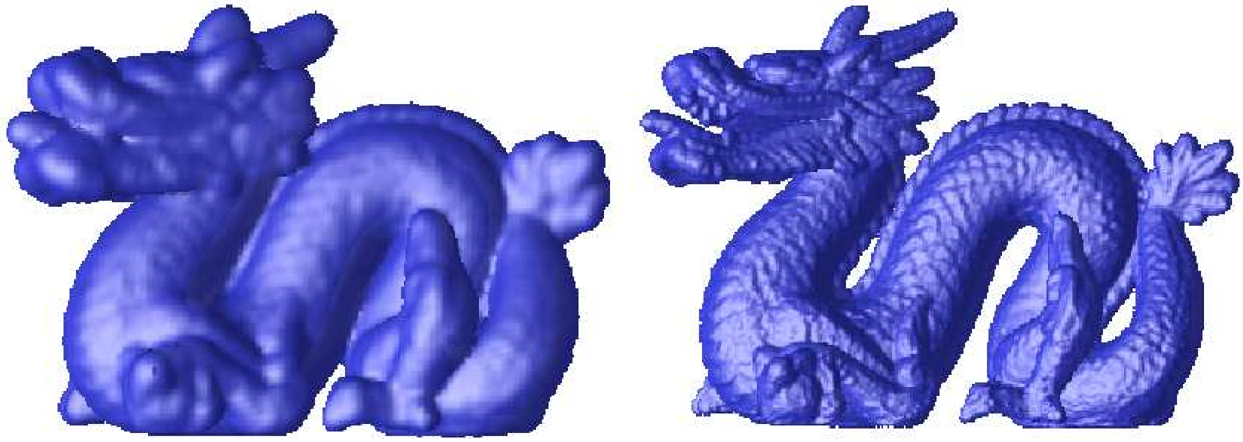
[8] M. Boué and P. Dupuis. Markov chain approximations for deterministic control problems with affine dynamics and quadratic cost in the control. *SIAM J. Numer. Anal.*, 36(3):667–695, 1999.
[9] D.E. Breen, S. Mauch, and R.T. Whitaker. 3d scan conversion of csg models into distance. *1998 Volume Visualization Symposium*, pages 7–14, October 1998.
[10] 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.
[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. *SIGGRAPH'96 Proceedings*, pages 303–312, 1996.
[12] P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.
[13] 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.
[14] H. Edelsbrunner and E. P. Mücke. Three dimensional α shapes. *ACM Trans. Graphics 13*, pages 43–72, 1994.
[15] S. F. Frisken, R. N. Perry, A. P. Rockwood, and T. R. Jones. Adaptively sampled distance fields: a general representation of shape for computer graphics. *proceedings, SIGGRAPH 2000*, 2000.
[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] R. Kimmel and A.M. Bruckstein. Shape offsets via level sets. *Computer-aided Design*, 1993.
[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] L. Piegl and W. Tiller. *The NURBS book*. Berlin, Germany: Springer-Verlag, 2nd edition edition, 1997.
[20] D.F. Rogers. *An Introduction to NURBS*. Morgan Kaufmann, 2000.
[21] Y.R. Tsai, L.-T Cheng, S. Osher, and H.K. Zhao. Fast sweeping algorithms for a class of Hamilton-Jacobi equations. *SINUM*, 2003.
[22] G. Turk and J. ÒBrien. Shape transformation using variational implicit functions. *SIGGRAPH99*, pages 335–342, August 1999.
[23] H. Zhao. Fast sweeping method for eikonal equations. *to appear in Math. Comp.*, 2003.
[24] 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.
[25] H. 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.
[26] D. Zorin and P. Schröder Eds. Subdivision for modeling and animation. *course notes for SIGGRAPH*, 2000.
[27] D. Zorin and P. Schröder. A unified framework for prime/dual quadrilateral subdivision schemes. *CAGD*, to appear.



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

Fig. 5.1.

(a) raw data (b) visualization using a distance contour after removal of outliers

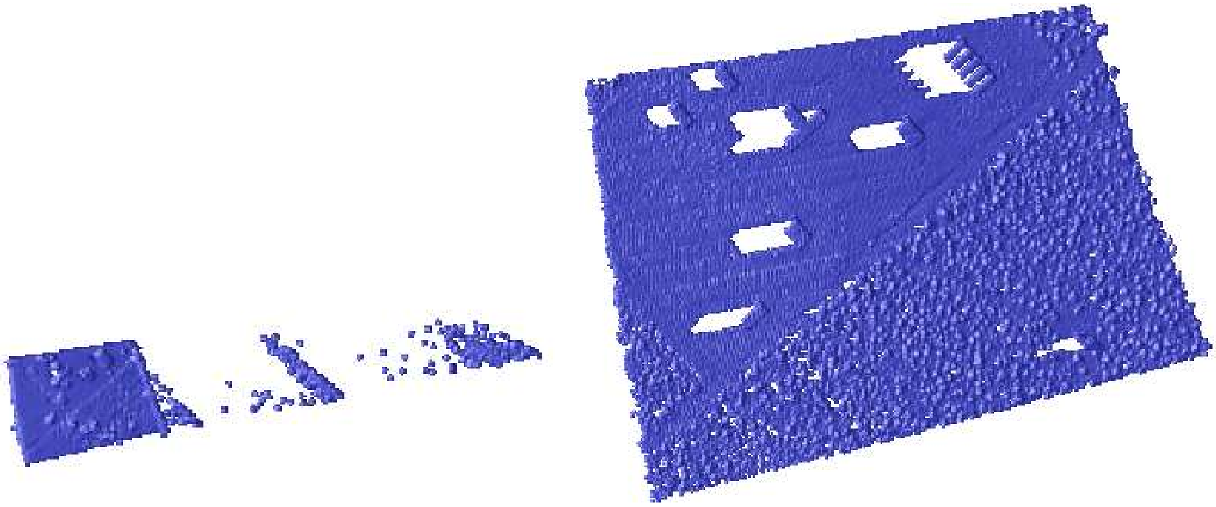


(c) visualization on a coarse grid (d) visualization using rescaled data

Fig. 5.2.

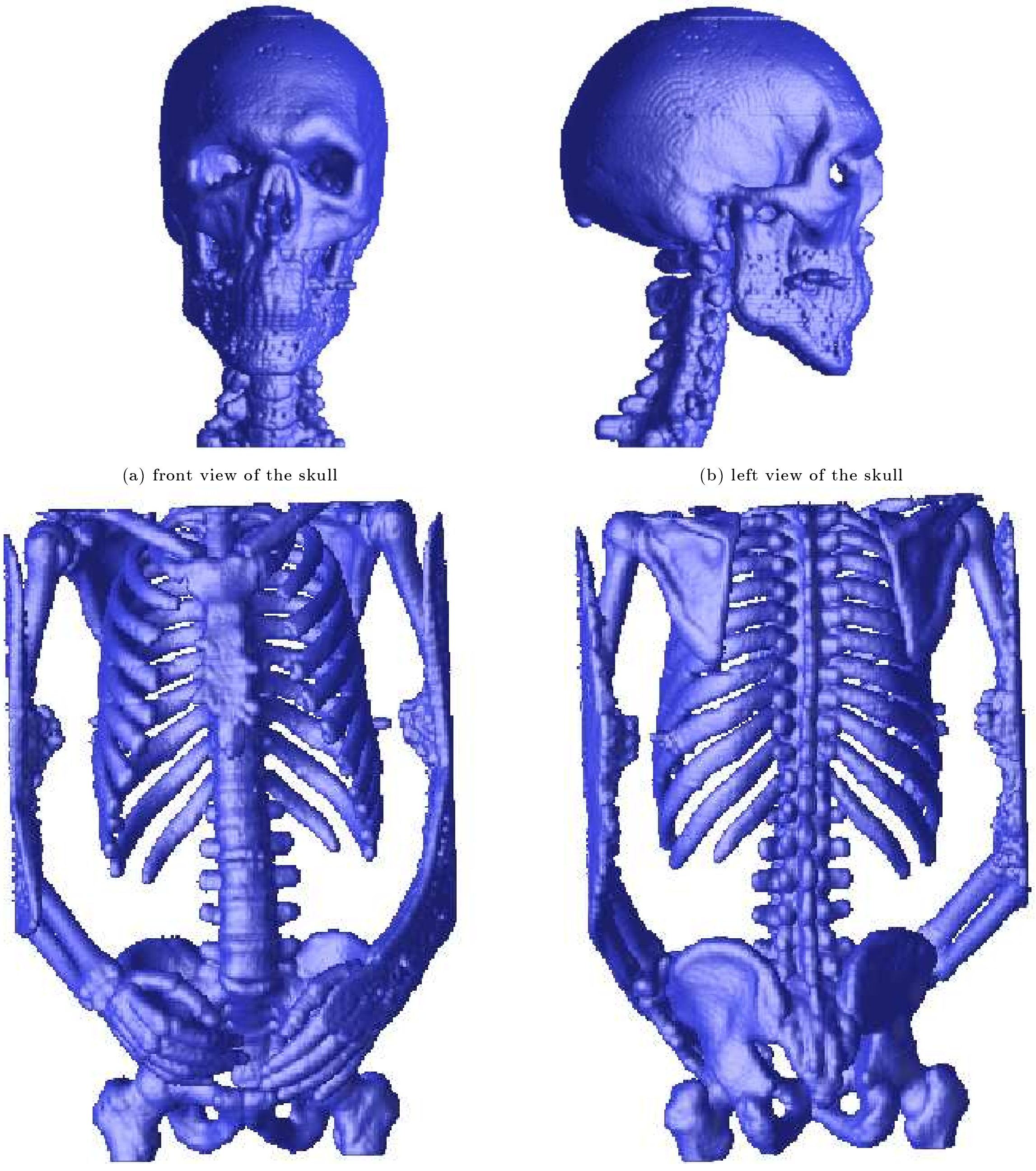
(a) front (b) diagonal (c) back

FIG. 5.3.



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

FIG. 5.4.



(a) front view of the skull

(b) left view of the skull

(c) front view of the skeleton

(d) back view of the skeleton

FIG. 5.5.