Cluster-Based Generation of Hierarchical Surface Models

Bjoern Heckel
Center for Image Processing and Integrated Computing (CIPIC)
Department of Computer Science
University of California, Davis, CA 95616-8562, U.S.A.
heckel@cs.ucdavis.edu

Antonio E. Uva
Dipartimento di Progettazione e Produzione Industriale
Politecnico di Bari, Viale Japigia 182, 70126 Bari, Italy
uva@cs.ucdavis.edu

Bernd Hamann
Center for Image Processing and Integrated Computing (CIPIC)
Department of Computer Science
University of California, Davis, CA 95616-8562, U.S.A.
hamann@cs.ucdavis.edu

Abstract

We present a highly efficient, automatic method for the generation of hierarchical surface triangulations. Given a set of scattered points in three-dimensional space, without connectivity information, our method reconstructs a valid triangulated surface model in a two-step procedure. First, we apply clustering to the set of given points and identify point subsets in locally nearly planar regions. Second, we construct a surface triangulation from the output of the clustering step. The output of the clustering step is a set of 2-manifold tiles, which locally approximate the underlying, unknown surface. We construct the triangulation of the entire surface by triangulating the individual tiles and triangulating the "gaps" between the tiles. Since we apply point clustering in a hierarchical fashion we can generate model hierarchies by triangulating various levels resulting from the hierarchical clustering step.

1. Introduction

Surface reconstruction is concerned with the extraction of shape information from point sets. Often, these point sets describe complex objects and are generated by scanning physical objects, by sampling other digital representations, or by merging data from different sources. Consequently, they might embody incompleteness, noise and redundancy, which makes a general approach for reconstructing surfaces a challenging problem. In many instances, the described objects are characterized by high complexity and high level of detail. Different levels of representation are needed to allow rapid rendering and interactive exploration and manipulation. Surface reconstruction problems arise in a wide range of scientific and engineering applications. The most important application is reverse engineering, i.e., the reconstruction of surfaces from digitized data.

We introduce an extremely fast surface reconstruction technique that is based on cluster analysis. Our approach generates a multiresolution representation of reconstructed surfaces from arbitrary point sets. Furthermore, our method allows us to control the level of detail locally.

Our paper is structured as follows: In Section 2, we briefly summarize related work in the fields of surface reconstruction, multiresolution modeling and clustering. In Section 3, we provide an overview of our approach, which is presented in greater detail in Section 4. The application of our surface reconstruction approach to a variety of data sets is described in Section 5. We conclude with summarizing the results in Section 6 and give an outlook on future work.

0-7695- 0503-1/00 $10.00 © 2000 IEEE

2. Related Work

2.1. Surface Reconstruction

The goal of surface reconstruction methods can be described like this:

Surface Reconstruction: Given a set of sample points X assumed to lie on or near an unknown surface U, construct a surface model S approximating U.

Most surface reconstruction methods require additional topological knowledge, such as connectivity between data, surface topology, or orientation.

Parametric reconstruction represents the reconstructed surface by mapping a two-dimensional parameter domain to a surface in three-dimensional space. This method usually requires knowledge of the topological type of the surface. Moreover, in order to converge to a valid model, this method also requires an initial embedding that is sufficiently close to the original surface and assumes a "good" parameterization that may be difficult to construct.

Function reconstruction methods deal with surfaces that are graphs of bivariate functions $f(x, y)$. Various applications are concerned only with this surface type, including digital elevation maps, digital satellite imaging, and medical imaging. It is also possible to apply these non-parametric methods in a local manner to general two-manifold data, which we can locally represent by a function $f(x, y)$.

Constriction methods attempt to find a surface mesh interpolating a set of data points in three-dimensional space without any given topological information. A three-manifold (tetrahedral) triangulation of the points (often Delaunay triangulation) is constructed first. The boundary of the triangulation is a mesh that defines the convex hull of the points. Since many surfaces are not convex, "undesired" elements must be eliminated. Therefore, one has to use iterative techniques creating a new triangulation by removing "undesired" tetrahedra. The result of these methods is a closed surface mesh.

Only a few methods developed fairly recently [15] [14] can create valid topological and geometrical models from only three-dimensional coordinates of sample points. Work in this field includes Edelsbrunner and Muecke [4], who generalize the notion of convex hull to that of an alpha hull (α hull). An interesting, directly related concept is the concept of alpha shapes (α shapes) used for the approximation of shapes in three-dimensional space (or even higher dimensions). These shapes are derived purely from finite sets of scattered, unorganized points. Applications for alpha shapes are automatic mesh generation, cluster identification for point sets in three-dimensional space, and modeling complex molecular structures.

Alpha shapes can be viewed as a generalization of the Delaunay triangulation. Ignoring certain degenerate cases, the Delaunay triangulation of a point set in three dimensions is characterized by the fact that the sphere passing though the four vertices of any tetrahedron does not contain any other point but the four vertices. The Delaunay triangulation defines a complex of edges, triangles, and tetrahedra. Given a specific alpha value, an edge in this complex belongs to the alpha shape if the radius of the smallest circle passing through the edge's end points is less than alpha. Similarly, a triangle (tetrahedron) in the complex belongs to the alpha shape if the radius of the smallest sphere passing through the triangle's (tetrahedron's) vertices is less than alpha. The Delaunay triangulation itself has an associated alpha value of infinity. Gradually decreasing the alpha value towards zero leads to structures consisting of increasingly "isolated sub-complexes," e.g., strings of edges, chains of triangles, groups of connected tetrahedra, and isolated points.

The alphashape approach has great potential for a general surface reconstruction paradigm. The alpha-shape approach will, in general, lead to geometric model with relative "thickness", i.e., it may locally describe a three-manifold region. This usually happen when the samples are noisy or when the underlying surface is not sufficiently smooth.

2.2. Multiresolution Modeling

A multiresolution model is a model defined by a set of detail levels of an object, which can be used to efficiently access any one of those levels on demand. Surface simplification is a particular form of multiresolution modeling where the goal is to take a polygonal model as input and generate a simplified model, i.e., an approximation of the original model. A variety of methods have been developed, including:

- *Image Pyramids*. They provide a successful and fairly simple multiresolution technique for raster images [18].

- *Volume Methods*. They allow multiresolution representations for models that are acquired as volumes and will be rendered as volumes. If the simplified volumes must be rendered using polygon-based rendering, then these volume methods may become less attractive [11].

- *Decimation Techniques*. There are vertex, edge, face and cell decimation techniques, which are all iterative simplification algorithms. In each step, an element is selected for removal according to one of several rules, and the algorithm must locally re-triangulate the area. Most algorithms were developed to reduce the density while preserving topology. Such algorithms become computationally expensive for large data sets [20] [13] [17] [7] [8].

- *Vertex Clustering*. This is a method that subdivides an object's bounding box into an octree-like grid structure, and all the vertices included in a single grid cell are clustered together and replaced by a single vertex. This is a very fast and general method, but it is often affected by a severe loss of detail and by distortions in the model [19].
- *Simplification Envelopes*. Such methods provide a global error measure for approximation quality of the simplified model, and they preserve topology. However, these methods require the original model to be an oriented manifold and can have problems with sharp edges [2].
- *Wavelet Methods*. They typically require a tensor product structure for a mesh to be represented at multiple resolution levels. Recently, different approaches have been introduced to overcome topology restrictions [3] [23].

2.3. Clustering

Clustering is a classical data analysis technique that has thoroughly been studied over the last few decades and has also been adopted as a standard technique in the emerging field of data mining [1]. So far, very little research has been done on applying data mining and clustering techniques to visualization problems. Vertex clustering, briefly described in 2.2, is one of the very few applications of clustering to visualization. Conversely, visualization techniques can be applied to the output of cluster analysis to present extracted patterns effectively [12] [21].

3. Overview of Approach

Starting with scattered point data in three-dimensional space, our method is able to generate a multiresolution model extremely quickly and automatically reconstruct a valid geometrical and topological model. Our method is unique in two aspects: (1) It does not require any connectivity information to be supplied with the sample points, and (2) it is significantly faster than most currently used methods. We achieve this by using a clustering methodology tailored to the specific needs of surface reconstruction from scattered data at multiple levels of detail.

Knowing that the data points originate from some unknown underlying two-dimensional manifold in three dimensions, we associate points with a certain cluster based on coplanarity and distance checks. By using more or less restrictive conditions regarding cluster membership, more or less points are associated with the individual clusters, and we thus have a means for controlling the levels of detail. Each cluster is essentially characterized by a closed polygon in three-dimensional space and an associated normal vector. This information suffices to locally characterize the underlying surface. These clusters can thus be thought of as "tiles" – unfortunately unconnected – approximating the surface from which all sample points originated.

This paradigm leads to an extremely efficient algorithm for tile generation at multiple levels of detail. Furthermore, the membership criterion used to associate samples with a cluster could easily be changed to accommodate hierarchical representations of vector fields in two or three dimensions or inherently three-manifold data (volumetric data).

We apply the Delaunay triangulation to the set of tile center points leading to a set of tetrahedra. At this point, we have established a topologically complete representation which, unfortunately, contains "too much connectivity" information: After all, we know that our samples belong to some underlying surface and thus the triangulation that we need for the tile centers must define a two-manifold surface. Using a set of rules, we produce a model that, in the end, will describe a true two-manifold surface triangulation in space.

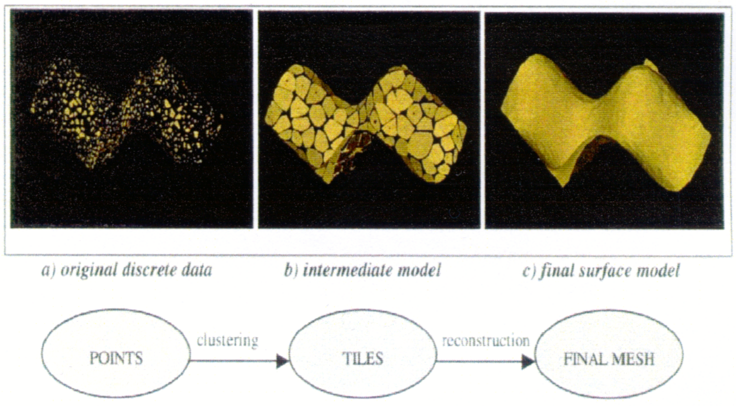


Figure 1. Illustration of the principal phases of the algorithm.

The input data set consists of a list of points in space (without connectivity). The output is a valid geometrical and topological surface model for the given samples. Our clustering-based surface reconstruction algorithm consists of two stages:

- Stage 1: Tile generation

  This stage uses the input data and applies a clustering algorithm that splits the original data set in quasi-planar tiles according to a user-defined error tolerance.

The goal of the clustering process is to construct a hierarchy of representations of the model based on the surface description by tiles. The output of the first stage is a set of tiles representing clusters of nearly coplanar points.

- Stage 2: Reconstruction

  In this stage, we fill the gaps between tiles by using a triangulation algorithm, i.e., we construct the missing connectivity information. For the triangulation-step, we consider only the boundary of the tiles. Since the triangulation algorithm will lead to a model whose boundary is the convex hull of the entire point set, a post-processing phase is necessary to delete "undesired" triangles/tetrahedra from our model.

4. Surface Reconstruction

4.1. Clustering

The input for our clustering process is a set of scattered points in three-dimensional space. Our method for creating the cluster hierarchy is based on an incremental and divisive paradigm. Initially, all points are placed in one cluster, which is recursively split. A cluster C is a subset of the given data set. The center of a cluster $C_{center} = (c_x, c_y, c_z)$ is defined as the geometric mean of the points $P_i = (x_i, y_i, z_i), i = 1, 2, \ldots, k$, associated with a cluster of size k. At each stage of the clustering process, every point is associated with exactly one cluster, which is the cluster with its center being the closest in terms of Euclidian distance. The internal error of a cluster is the sum of the distances from the cluster center to the associated points, i.e.,

$$Error_{internal}(C) = \sum_{p[i] \in C} \| p[i] - C_{center} \| . \qquad (1)$$

The global error is defined as the sum of the internal error over all clusters. In each iteration of the clustering process, the cluster C_i with the highest internal error is split into two clusters. Each iteration decreases the global error and the maximum internal error. The centers of the generated clusters are determined by adding/subtracting an offset vector to/from the center of the original cluster. This offset vector lies either in the direction of highest deviation of the cluster data or in the direction of maximum variance. The direction of maximum variance is computed by performing principal component analysis (PCA) for the 3-by-3 covariance-matrix M, given by

$$M = \Delta^T \cdot \Delta, \qquad (2)$$

where Δ is the cluster's normalized k-by-3 data matrix Δ, defined as

$$\Delta = \begin{bmatrix} x_1 - c_x & y_1 - c_y & z_1 - c_z \\ x_2 - c_x & y_2 - c_y & z_2 - c_z \\ \ldots & \ldots & \ldots \\ x_k - c_x & y_k - c_y & z_k - c_z \end{bmatrix} . \qquad (3)$$

We compute the eigenvalues and "eigendirections" for M. The direction of maximum variance is equivalent to the eigendirection with the largest eigenvalue. Using the direction of maximum variance is generally more accurate, but this is only feasible when the number of points is relatively small e.g., less than 500. Therefore, a threshold variable τ_{PCA} is used to control which splitting mechanism is used. If the number of points associated with the cluster C_i to be split exceeds τ_{PCA}, the cluster C_i is split in the direction of highest deviation. Otherwise, the eigenvector with the highest eigenvalue is used to determine the centers of the new clusters that result from splitting C_i. After splitting a cluster, a local reclassification scheme is used to improve the quality of the classification.

The points to be reclassified are given by all points that are assigned to the split cluster or to the Gabriel neighbors of that cluster. Hierarchical clustering is illustrated for curve reconstruction in Figure 2.

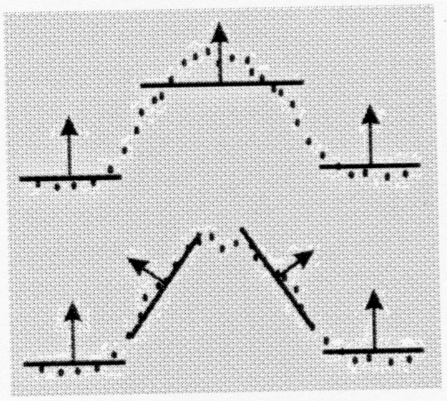


Figure 2. Cluster splitting.

Two clusters are Gabriel neighbors when they are connected in the Gabriel graph constructed from the cluster centers. Two points p and q are connected in the Gabriel graph when only p and q—but no other point— are contained in the hypersphere with the midpoint of p and q as its center and the distance from p to q as its diameter. (The Gabriel graph of a point set is a subset of its Delaunay graph and a superset of its relative neighborhood graph [16].)

After each reclassification step we update the local neighborhood. The cluster centers are moved to reflect the changes in the point-cluster association due to local reclassification. The Gabriel graph is updated locally and another

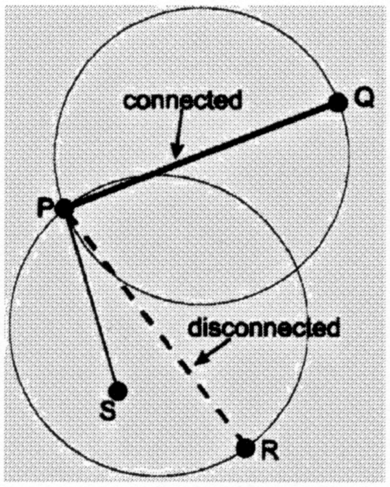


Figure 3. Gabriel neighbors

cluster is split subsequently. The clustering process terminates when the smallest eigenvalues of all clusters do not exceed the threshold $\tau_{PCA\ lim}$.

Algorithm 1 *Creating the cluster hierarchy*

```
**Input: set of points**
**Output: set of clusters**
while $\tau_{PCA\ lim}$ is exceeded {
determine cluster $C_s$ to be split;
split cluster $C_s$ into two new clusters and define their centers;
create list of clusters in the local neighborhood of split cluster;
perform local reclassification;
update neighborhood;
}
```

4.2. Tile Generation

At this point, we have generated a set of clusters that partition the original data set. Since we choose a relatively small value for $\tau_{PCA\ \mathrm{lim}}$, the points associated with a certain cluster are nearly coplanar. Thus, the point clusters have an almost flat shape. For each cluster C_i, the cluster center and the two eigendirections with the two largest eigenvalues define a plane P_i that locally minimizes the sum of the plane-to-point distances of the associated points. We project all points p_k associated with cluster C_i into the plane P_i and compute the convex hull H_i for the projected points p'_k in the associated plane. We map the points p'_l on H_i back to their original locations in three-dimensional space. The result is a closed, nearly planar polygon in three-dimensional space, defining the so-called tile T_i of cluster C_i consisting of a set of points p''_l lying in H_i. For each cluster we generate a pseudo tile T'_i by replacing the points p''_l, defining the tile boundary, by their original points p_l.

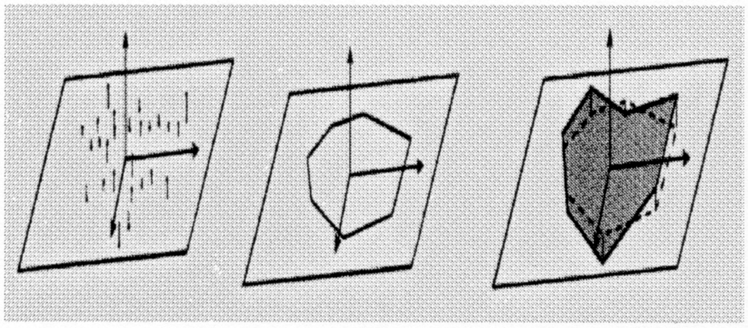


Figure 4. Tile generation.

Once tile generation is completed, each cluster C_i has two corresponding representations by its tile T_i and its pseudo tile T'_i.The normal directions of T_i and T'_i are defined by the cluster's eigendirection with the smallest eigenvalue.

4.3. Reconstruction

For the reconstruction process we triangulate pseudo tiles by connecting the cluster center with the vertices of its boundary polygon. The resulting set of triangles $\mathbf{T}_1$ (*tile triangulation*) is a close approximation of the unknown surface. However, it does not yet describe a complete model due to the lack of connectivity information for the tiles.

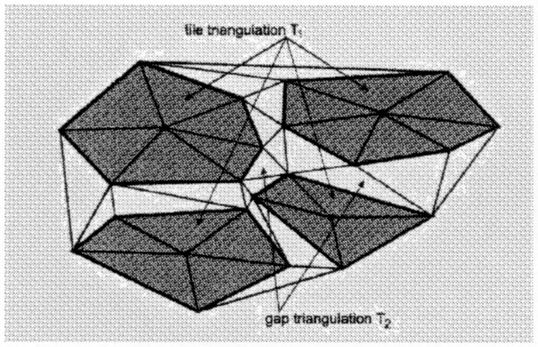


Figure 5. Triangulation of tiles and between tiles.

A "connected model" is obtained by $\mathbf{T}_1$ with a set of triangles $\mathbf{T}_2$ (*gap triangulation*) that fills the gaps between the tiles. To determine $\mathbf{T}_2$ we apply a Delaunay triangulation algorithm to the boundary points of the pseudo tiles [1].

[1]The result of the Delaunay triangulation step is a "true volumetric" triangulation of the input points, i.e., a set of tetrahedra, from which we have to eliminate "undesired" edges, triangles, tetrahedra according to a set of heuristics.

The algorithm we use employ a variant of the randomized incremental-flip algorithm developed by Edelsbrunner and Shah [5]. Based on the characteristics of the data, we use two different variants of the Delaunay triangulation:

- For functional data (i.e., data sets where each point of the input lies on the graph of a bivariate function $f(x, y)$) we project the points in the x-y plane and apply the two-dimensional Delaunay triangulation algorithm. We use the resulting triangulation to establish the connectivity of the points in three-dimensional space (Figure 6).
- For an arbitrary data set, the three-dimensional Delaunay triangulation algorithm is directly applied to the boundary points of the pseudo tiles.

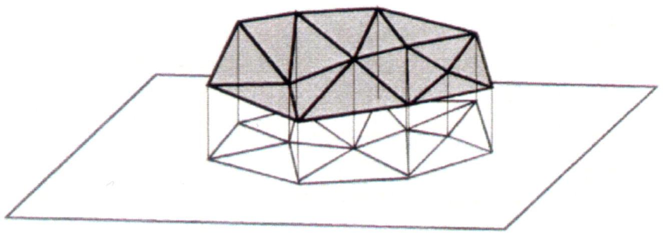


Figure 6. Triangulation of functional data.

Since the boundary of the Delaunay triangulation describes the convex hull of the point set, we have to remove certain elements of the mesh (triangles or tetrahedra) that "do not belong" to the desired surface. Our approach to reduce the Delaunay triangulation to $\mathbf{T}_2$ is related to the alpha shape definition by Edelsbrunner and Mücke [4]. We remove triangles from the Delaunay triangulation when

- all points lie in one tile or
- the points do not fit in a sphere of radius b,

where the value of b is chosen by locally adjusting the global alpha threshold depending on the tile area.

In the original alpha shape approach, the quality of the reconstructed model is very sensitive to the alpha value, which is used globally to identify triangles to be removed. When using an alpha value that is too large, "undesired" triangles do not get removed, while an α-value that is too small might remove features that do belong to the model. A valid surface model cannot be reconstructed in many cases, due to an inadequately chosen alpha value. Our clustering-based approach yields additional information (e.g., tile area, connectivity, local density of point distribution) that can be used to locally refine the alpha value.

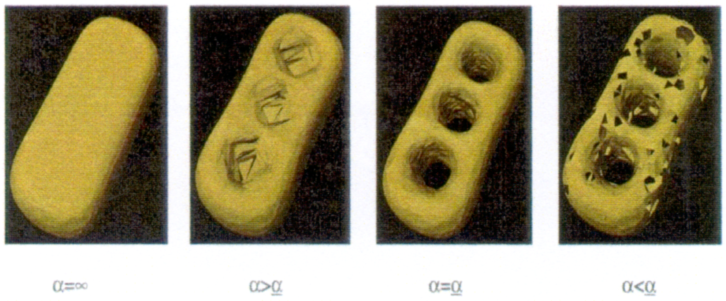

$\alpha=\infty$
$\alpha>\underline{\alpha}$
$\alpha=\underline{\alpha}$
$\alpha<\underline{\alpha}$

Figure 7. Models obtained using varying alpha values ($\underline{\alpha}$=optimal alpha value)

Since we have "almost defined a topology" after the tile generation step, completing the reconstruction is a far simpler problem than applying the alpha shape approach directly to the original set of points. Because the gaps between the tiles are relatively small, one can choose a very small alpha value without removing important features of the model. However, long, skinny triangles along the tile boundaries can result due to small alpha values. We deal with this problem by inserting vertices along boundary edges when they exceed a certain length.

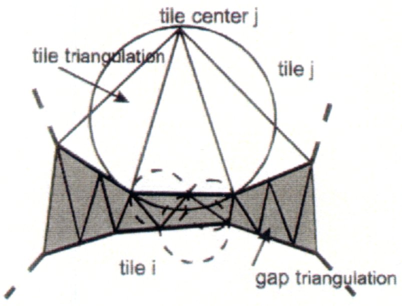


Figure 8. Removing "undesired" triangles.

With this enhancement, our approach is relatively insensitive to changes of the alpha value. Since a wide range of alpha values yields a consistent model, the global alpha value can be determined automatically as a linear function of the average edge length. However, when a the data set contains a large number of discontinuities (the underlying surface is actually discontinuous) we propose to select the alpha value manually. Removing the undesired triangles from the Delaunay triangulation of the tile boundary points yields the set of triangles $\mathbf{T}_2$ that fill the gap between tiles. By merging $\mathbf{T}_1$ and $\mathbf{T}_2$ we obtain a consistent model.

Algorithm 2 *Reconstruct model*

```
    Input: set of pseudo tiles
    Output: consistent model
```

```
reconstruct model {
generate $\mathbf{T}_1$ by triangulating pseudo tiles;
generate $\mathbf{T}_2$:
    if boundary edge length $> \epsilon$
    then insert additional vertices;
    triangulate vertices using Delaunay triangulation;
    remove undesired triangles;
    merge $\mathbf{T}_1$ and $\mathbf{T}_2$;
}
```

5. Results

We have applied our method to a variety of data sets. The number of tiles produced by our algorithm is—for a given data set—a function of the threshold for the smallest eigenvalue of a cluster (see 4.1). The error E_k of a point p_k of the original data set associated with cluster C_i is defined as the distance of p_k to the plane H_i containing tile $\mathbf{T}_1$, i.e.,

$$E_k = Dist\,(p[k], H_i)\,. \tag{4}$$

The global error E_{global} is defined as the sum of the errors of all points in the original data set, i.e.,

$$E_{global} = \sum_{p[k]} Dist\,(p[k], H_i)\,. \tag{5}$$

To measure the root-mean-square (RMS) error, we compute the object diameter, which is the diameter of the smallest sphere that includes the object. We have tested our approach for five different data sets, which are listed in Table 1.

Table 1. Number of points and object diameters

Data set	Original # of pts.	Object diameter[2]
Rabbit	35929	0.1557
3Holes	4000	208
Car	20621	0.8
Step function[3]	8000	1
Mt. St. Helens	151728	567,602

We obtain an RMS error measure E_{RMS} by dividing the global error by the product of object diameter and number of points in the original data set, i.e.,

$$E_{RMS} = \frac{E_{global}}{number\ of\ points \times object\ diameter}\,. \tag{6}$$

The listed execution times were obtained for an SGI O^2 workstation with a 180 MHz R5000 processor. Table 2 summarizes the performance and quality information for our five data sets.

Table 2. Data sets: 1-Rabbit, 2-3Holes, 3-Car, 4-step function[3], 5-Mt. St. Helens

Set	$\tau_{PCA\ lim}$	$t_{cluster}$	N_{tiles}	N_{points}	E_{RMS}
1	0.006	7.83	50	1110	1.68
1	0.002	15.05	354	4764	0.049
1	0.001	18.94	727	8016	0.026
2	10	0.97	59	762	1.7
2	5	1.24	91	1002	0.9
2	2	1.77	202	1815	0.39
2	1	2.38	402	2620	0.19
3	0.0015	8.37	517	5174	0.064
4	0.001	1.63	60	715	0
5	80	43.08	154	3100	0.0034
5	40	51.06	382	6307	0.0025
5	20	74.70	1391	17513	0.0014

Figures 9–13 show reconstructed models for our five test data sets. Figures 9 and 10 show tiles and reconstructed, triangulated surfaces. Figure 9 demonstrates how our method can be used to preserve or remove a discontinuity.

Figures 11, 12, and 13 illustrate the power of our technique to generate multiresolution models. Three examples show that our algorithm can handle extremely large data sets (Mt. St. Helens) and surfaces that are topologically complicated (3Holes, Rabbit).

6. Conclusions

The algorithm we have presented allows the generation of a hierarchy of surface models from discrete point sets without known connectivity information. While we have demonstrated the power of our approach only for surface models, we are quite confident that the same clustering paradigm, when applied to more general two- or three-manifold, or even time-varying data, would significantly speed up the process of computing level-of-detail representations.

We plan to extend our approach to the clustering of more general scattered data sets describing scalar and vectors fields, defined over either two-dimensional or three-

[2] We define the object diameter as the diameter of the smallest sphere including all original data.

[3] $f(x, y) = 1, y \geq x$, and $f(x, y) = 0, y < x;\ \ x, y \in [0, 1]$

dimensional domains. Faster algorithms for the generation of data hierarchies for scientific visualization will become more important as our ability to generate ever larger data sets increases: Computing a data hierarchy prior to the application of a visualization algorithm should not require minutes or hours but seconds instead. We believe that our clustering methodology provides one viable answer to this problem.

Currently, we are working on a scalable parallelization of our approach. We believe that this parallelization will allow us to analyze data sets that consist of several millions of points in real time.

Furthermore, we plan to extend our algorithm to ensure that all tetrahedra are removed, such that the reconstructed model is a true two-manifold representation. Regarding the triangle elimination phase, we will develop means to guarantee that no "holes" are inserted into the model.

Acknowledgements
This work was supported by the IBM Almaden Research Center, the National Science Foundation under contract ASC 9624034 (CAREER Award) to the University of California, Davis. In addition, this project was funded by the Office of Naval Research under contract N00014-97-1-0222, the Army Research Office under contract ARO 36598-MA-RIP; the NASA Ames Research Center through an NRA award under contract NAG2-1216; the Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreement B347878 and under Memorandum Agreement B503159; and the North Atlantic Treaty Organization (NATO) under contract CRG.971628 awarded to the University of California, Davis. We would like to thank all members of the Visualization Thrust at the Center for Image Processing and Integrated Computing (CIPIC) at the University of California, Davis. We particularly thank Terence T. Cheung for formatting this paper.

References

[1] P. Arabie, L. J. Hubert, and G. De Soete. *Clustering and Classification*. World Scientific Publ., River Edge, 1996.

[2] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, B. F., and W. Wright. Simplification envelopes. In *Proc. of SIGGRAPH '96*, pages 119–128, 1996.

[3] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Proc. of SIGGRAPH '94*, pages 173–182, 1995.

[4] H. Edelsbrunner and E. P. Muecke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1), January 1994.

[5] H. Edelsbrunner and R. Shah. Incremental topological flipping works for regular triangulations. In *In the Proceedings of the 8th Annual Symposium on Computational Geometry*, pages 43–52, 1992.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, November 1996.

[7] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of SIGGRAPH '97*, pages 209–216, 1997.

[8] T. S. Gieng, B. Hamann, K. I. Joy, G. L. Schussman, and I. J. Trotts. Smooth hierarchical surface triangulations. In *Yagel, R. and Hagen, H., eds., Visualization '97, IEEE Computer Society Press*, pages 379–386, Los Alamitos, CA, 1997.

[9] B. Hamann. A data reduction scheme for triangulated surfaces. *Computer Aided Geometric Design*, 11(2):197–214, November 1994.

[10] B. Hamann and B. Jordan. *Triangulations from repeated bisection*, pages 229–236. Vanderbilt University Press, Nashville, TN, 1998.

[11] T. He, L. Hong, A. Kaufman, A. Varshney, and S. Wang. Voxel-based object simplification. In *Proc. of IEEE Visualization '95*, pages 296–303. IEEE Comput. Soc. Press, 1995.

[12] B. Heckel and B. Hamann. Visualization of cluster hierarchies. In *Proceedings of IT&S/SPIE Electronic Imaging Symposium 1998*, pages 162–171, January 1998.

[13] H. Hoppe. Progressive meshes. In *Proc. of SIGGRAPH '96*, pages 99–108, 1996.

[14] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *Proc. of SIGGRAPH '94*, pages 295–302, 1994.

[15] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. In *Proc. of SIGGRAPH '94*, pages 71–78, 1992.

[16] A. Okabe, B. Boots, and K. Sugihara. *Spatial Tessellations*. John Wiley & Sons, Chichester, 1992.

[17] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *Proc. of SIGGRAPH '97*, pages 217–224, 1997.

[18] A. Rosenfeld. *Multiresolution Image Processing and Analysis*. Springer-Verlag, Berlin, 1994.

[19] J. Rossignac and P. Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Proc. of Modeling in Computer Graphics: Methods and Applications*. Springer-Verlag, 1993.

[20] W. Schroeder, J. Zarge, and W. E. Lorensen. Decimation of triangle meshes. In *Proc. of SIGGRAPH '92*, pages 65–70, 1992.

[21] T. Sprenger, M. H. Gross, A. Eggenberger, and M. Kaufmann. A framework for physically-based information visualization. In *Eigth EuroGraphics-Workshop on Visualization in Scientific Computing*, pages 77–86, France, 1997.

[22] D. Zorin, P. Schroeder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Proc. of SIGGRAPH '96*, pages 182–192, 1996.

[23] D. Zorin, P. Schroeder, and W. Sweldens. Interactive multiresolution mesh editing. In *Proc. of SIGGRAPH '97*, pages 259–268, 1997.

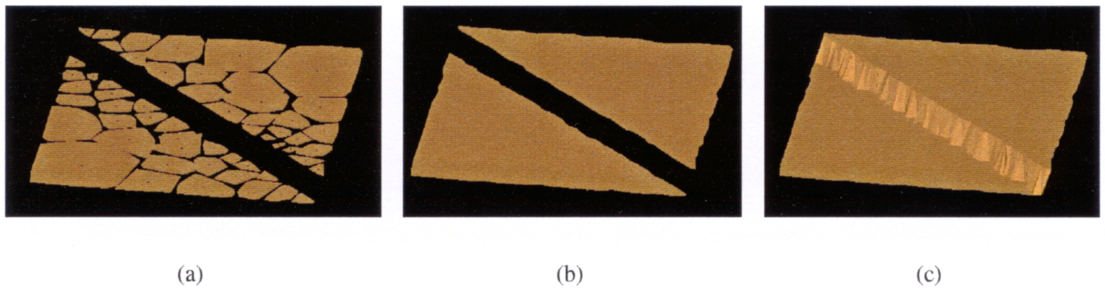


Figure 9. Discontinuous example (step function); (a) tiles, (b) discontinuous triangulation, (c) continuous triangulation.

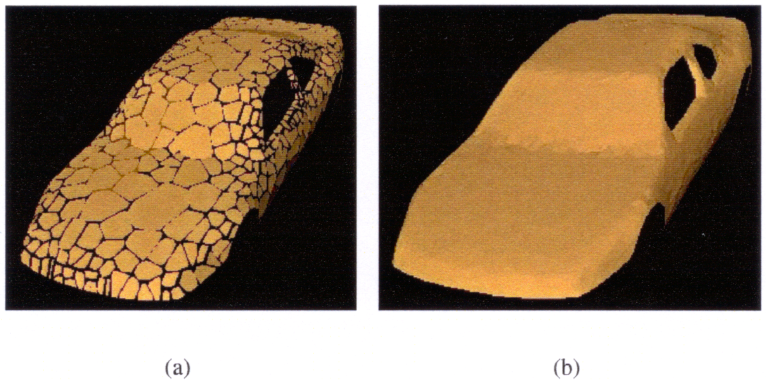


Figure 10. Car data set; (a) tiles, (b) reconstructed model.

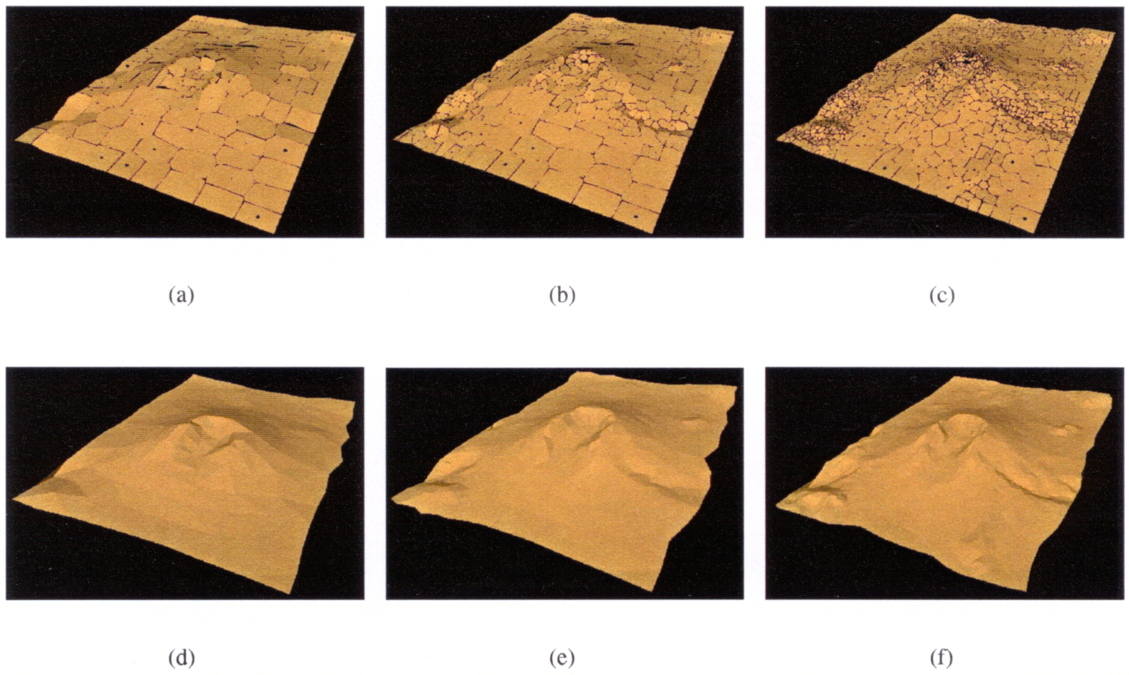
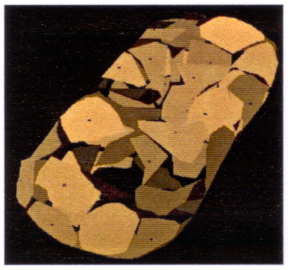
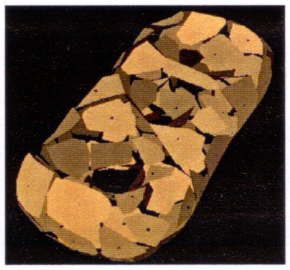
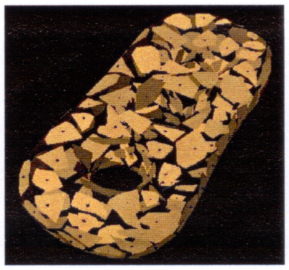
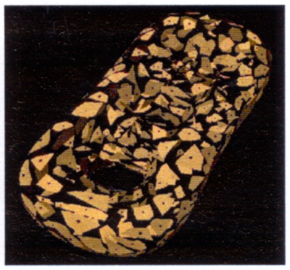


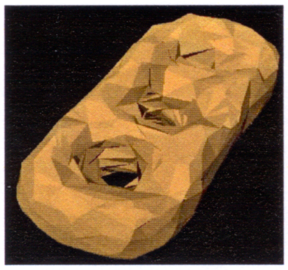
Figure 11. Three resolution levels of Mt. St. Helens data set (154, 382 and 1391 tiles); (a)–(c) tiles, (d)–(f) reconstructed surface.

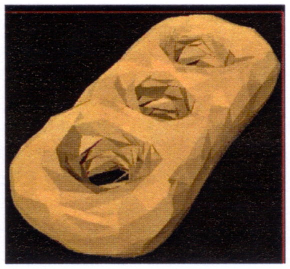

(a)

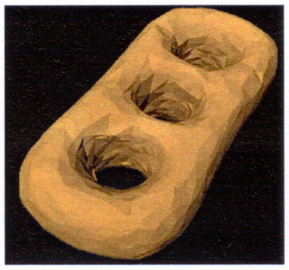

(b)


(c)


(d)


(e)


(f)


(g)

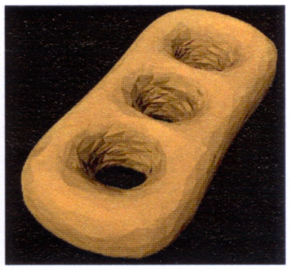
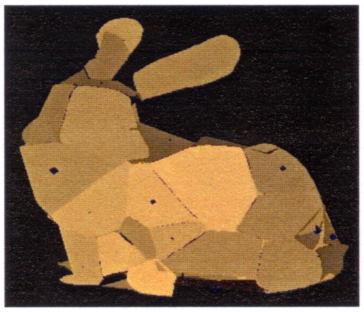
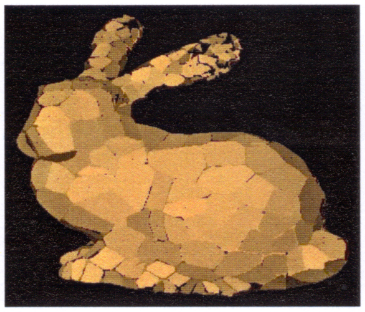
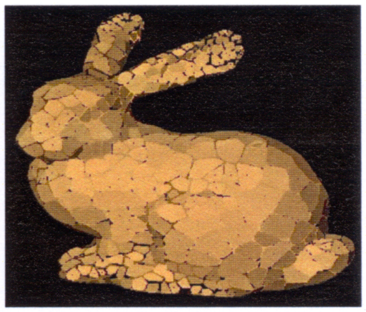
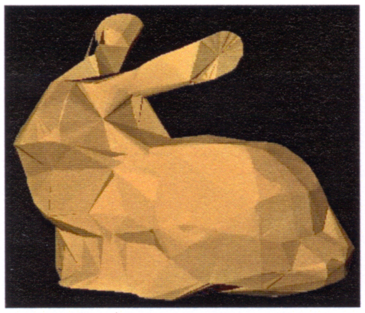

(h)

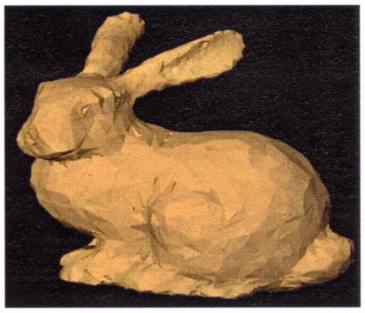
Figure 12. Four resolution levels of three-hole data set 59, 91, 202 and 402 tiles); (a)–(d) tiles, (e)–(h) reconstructed model.


(a)


(b)


(c)


(d)


(e)

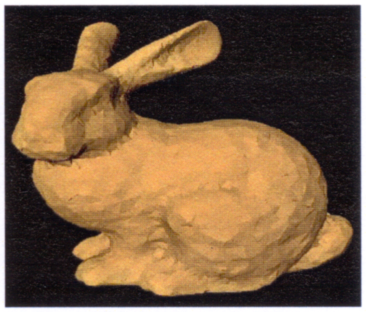

(f)

Figure 13. Three resolution levels of rabbit data set (50, 354 and 727 tiles); (a)–c) tiles, (d)–(f) reconstructed model.