Polygonizing Non-uniformly Distributed 3D Points by Advancing Mesh Frontiers

Indriyati Atmosukarto, Luping Zhou, Wee Kheng Leow, Zhiyong Huang *
School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543, Singapore
indriyat, zhoulupi, leowwk, huangzy@comp.nus.edu.sg

Abstract

3D digitization devices produce very large sets of 3D points sampled from the surfaces of the objects being scanned. A mesh construction procedure needs to be applied to derive polygon mesh from the 3D point sets. As the 3D points derived from digitization devices based on digital imaging technologies are inherently non-uniformly distributed over regions that may contain surface discontinuities, existing methods are not suitable for polygonizing them. This paper describes a novel polygonization algorithm for constructing triangle mesh from unorganized 3D points. In contrast to existing methods, this algorithm begins the mesh construction process from 3D points lying on smooth surfaces, and advances the mesh frontier towards 3D points lying near surface discontinuities. If 3D points along the edges and at the corners are sampled, then the algorithm will form an edge where two advancing frontiers meet, and a corner where three or more frontiers meet. Otherwise, the algorithm constructs approximations of the edges and corners. It can be shown that this frontier advancing algorithm performs 2D Delaunay triangulation of 3D points lying on a plane in 3D space.

1. Introduction

3D digitization devices produce very large sets of 3D points sampled from the surfaces of the objects being scanned. These 3D points are usually not suitable for direct use in computer graphics applications. A mesh construction procedure needs to be applied to derive polygon mesh from the 3D points. Polygon mesh is among the most common data structure used for representing objects in computer graphics. Its popularity stems from the following reasons:

1. Simplicity for fast rendering: In virtual reality, 3D games, and multimedia applications, 3D objects and scenes must be represented as meshes so that graphics acceleration hardware can be utilized to generate high quality images of the objects and scenes. Furthermore, most animation techniques such as facial animation, which is popular in multimedia presentation, are applicable only to mesh representations.

2. Standard for the industry: In the manufacturing industry, almost all computer-aided design, engineering, and manufacturing (CAD/CAE/CAM) software require 3D meshes for finite element analysis (FEM), assembly planning, process automation, and manufacturing using numerical control (NC).

3. Popularity in Web applications: In World Wide Web (WWW), the inclusion of 3D objects into web pages is becoming a major trend. In many Internet standards for 3D contents, e.g., VRML2, MPEG-4, and Java3D, mesh representation takes the central role.

As the 3D points, in particular those derived from digitization devices based on digital imaging technologies, are inherently non-uniformly distributed over regions that may contain surface discontinuities, existing methods are not suitable for polygonizing them. This paper presents novel geometry-based approach called *frontier advancing polygonization* which has the following properties:

1. It identifies possible surface discontinuities. This polygonization algorithm identifies reliable points lying on relatively flat surfaces and ambiguous points lying near surface discontinuities. Quantitative test results show that the method can effectively distinguish reliable points from ambiguous points.

2. It is progressive. This algorithm begins the mesh construction process from reliable 3D points, and advances the mesh frontier towards ambiguous 3D points lying near surface discontinuities. If 3D points along the edges and at the corners are sampled, then the algorithm will form an edge where two advancing frontiers meet, and a corner where three or more frontiers meet.

*This research is supported by NUS ARF R-252-000-051-112

Otherwise, the algorithm constructs approximations of the edges and corners. In addition, it can also construct a mesh from non-uniformly distributed 3D points.

3. It produces a 2D Delaunay triangulation for 3D points lying on a plane in 3D space. For 3D points lying on a smooth curved surface, the constructed mesh is Delaunay Triangulation of a piecewise planar approximation of the curved surface. It should be noted that a standard 3D Delaunay triangulation that produces 3D polyhedras is not appropriate for our application since the 3D points always lie on the object's surfaces. Instead, the frontier advancing algorithm produces 2D triangles that approximate the surfaces of the 3D model.

2. Related Work

Many different methods exist for mesh construction from 3D points. Hoppe et al. formalized the problem as follows [10]: given a set of points in R^3 without any information about the structure or organization, construct a polygon mesh, possibly with boundary, from the 3D points. They proposed a contouring algorithm that extracts the zero set of the signed distance function that approximates the surface. Curless and Levoy proposed a volume refining framework taking the similar idea [3]. The zero set algorithm produces an approximating rather than interpolating mesh. Amenta et al. proposed the crust algorithm, the first algorithm based on the 3D Voronoi diagram with provable guarantees [1]. The mesh produced is guaranteed to be topologically correct and to converge to the original surface as the sampling density increases. Unlike in an ideal situation, the 3D points recovered from an image sequence are constrained by the features present in the images. It is difficult for these 3D points to meet the dense sampling criterion of the crust algorithm. Moreover, the crust algorithm does not solve the problem of reconstructing sharp boundaries.

The α-shape of Edelsbrunner et al. [6, 5] is a parameterized construction that associates a polyhedral shape with an unorganized set of points. Its major idea is that a simplex (edge, triangle, or tetrahedron) is included in an α-shape if it contains some circumspheres with no interior sample points. A 3-D circumsphere is a sphere of radius α whose surface touches at least three sample points. The spectrum of α-shapes, that is, the α-shapes for all possible values of α, gives an idea of the overall shape and natural dimensionality of the point set. We observed that the α-shape cannot be directly used for our problem. First, due to non-uniform sampling, there is not a unique α for the entire set of 3D points. Second, the 3D points cannot be easily clustered so that each cluster has a constant sampling density.

The α-shape and crust algorithms make use of Delaunay triangulation [2] to construct triangle mesh. Several algorithms for Delaunay triangulation are well known. Green and Sibson devised an incremental algorithm that computes the Voronoi diagram [8], which is the dual of Delaunay triangulation, of a set of points. Fang and Piegl [7] used a uniform grid to implement their delaunay triangulation. Lawson developed an algorithm by flipping diagonals of triangles [11] and Guibas, Knuth, and Sharir presented an optimal implementation of Lawson's method based on randomized algorithm [9]. These algorithms, and most other randomized incremental algorithms in computational geometry, all work according to the principle of *structure maintaining* [4]: To add the next triangle, the algorithm first finds out which part of the current structure has to be changed to resolve conflicts with the new triangle. Then, it updates the structure locally, removing the conflicting triangles and adding the new triangle. So, the initial triangulation result of a subset of 3D points may not be retained though it is optimal with respect to the subset.The principle of structure maintaining is inefficient and inconvenient for our application.

Oblonsek and Guid [12] presented a new three-phase method for object reconstruction from 3D scattered points. The first phase generates a base approximation of object surface. The second phase extracts sharp edges and corners which are used as constraints for the reconstruction in the last phase. Like our method, this method attempted to reconstruct sharp edges and corners but it adopted a different approach. To handle surface discontinuities, our algorithm adopts the strategy of constructing the mesh starting at relatively smooth and flat surfaces and working progressively towards edges and corners. This strategy can be implemented more elegantly and efficiently by adopting a mesh construction process that only adds triangles and never removes triangles.

3. Frontier Advancing Polygonization

The frontier advancing polygonization algorithm consists of two main steps:

1. Identifying reliable points:
   Reliable points are 3D points that lie on relatively flat surfaces. 3D points that lie near surface discontinuities or on surfaces with large curvatures are called *ambiguous points*. The closer a point is to an edge or a corner, the larger is its ambiguity.

2. Advancing Mesh Frontier:
   A mesh is first constructed around a reliable point. Then, 3D points near the frontier of the mesh are added, and the process continues in increasing order of ambiguity.

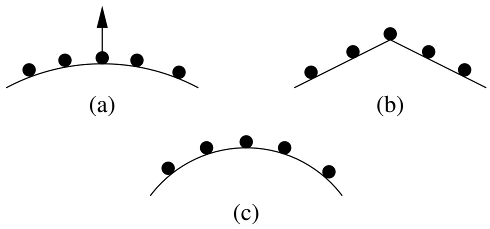


Figure 1. Identifying reliable points. (a) 3D points (dots) distributed over a relatively flat surface (line, viewed from the side) have a small third eigenvalue, whereas those lying on a sharp edge (b) or a surface with a large curvature (c) have large third eigenvalues. The arrow indicates the direction of the third eigenvector.

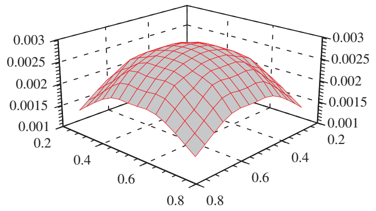


Figure 2. Graph illustrating the ambiguity level of points lying near the peak of a Gaussian surface. The ambiguity levels decrease smoothly from the peak position outward.

4. Identifying Reliable Points

Reliable points that lie on relatively flat surfaces can be identified using Principal Component Analysis (PCA) in a manner similar to the method described in [10]. Given a set of 3D points, PCA performs eigen-decomposition of the covariance matrix of the coordinates of the 3D points. It produces three eigenvectors $\mathbf{e}_i$ with associated eigenvalues λ_i, $i = 1, 2, 3$, in decreasing value. The eigenvectors are orthogonal to each others and are aligned with the directions of maximum variations. For a set of points lying on a relatively flat surface, the third eigenvector would point in the direction of the surface normal. The third eigenvalue λ_3 would be very small (Fig. 1a), approaching the value 0 when the surface approaches a 3D plane. Conversely, for a set of points distributed near a surface discontinuity (Fig. 1b) or a surface with a large curvature (Fig. 1c), λ_3 would be large compared to λ_1. Therefore, the ratio of λ_3 over λ_1 can be used as a measure of the likelihood that a point lies near a surface discontinuity.

4.1. Algorithm

The following algorithm summarizes the process of identifying reliable and ambiguous points:

A1: Identifying Reliable Points:
For a point p,
  Find the neighbors of p within a sphere of radius r centered at p.
  Perform PCA on this set of 3D points.
  Compute ambiguity level = λ_3/λ_1.
  If $\lambda_3/\lambda_1 < \Gamma_a$,
    p is a reliable point.
  Else, p is an ambiguous point.

The radius r is varied adaptively to capture about 10 points in the sphere so that λ_3 can be estimated reliably. In this way, points located in densely sampled regions will have smaller neighborhood size than those in coarsely sampled regions. The threshold Γ_a can be a fixed parameter because it is independent of the sampling density of the 3D points. In the current implementation, it is fixed at 0.05.

4.2. Test Results

Tests were conducted to verify the accuracy of identifying the reliable points. Three types of synthetic surfaces were used in the tests: (1) a Gaussian surface, (2) an edge composed of two planes forming an obtuse angle, and (3) a corner composed of three orthogonal planes. The first surface is a smooth curved surface, while the third surface is sharper than the second one. 3D points were sample from the surfaces at regular intervals, and their ambiguity levels were computed. Figures 2 to 4 illustrate 3D plots of the ambiguity levels of the points on the three surfaces. Points lying on flat parts of the edge and the corner have 0 ambiguity and points lying on the smooth curve have very low ambiguity level. These points have ambiguity levels below the ambiguity threshold (0.05) and are regarded as reliable points. As the points get closer to the edge and the corner, their ambiguity level increase. Since the edges less sharp then the corner, points on the edges have lower ambiguity than the corner point. A comparison of the ambiguity levels of the three surfaces is summarized in Fig. 5. These results show that the effectiveness of computing ambiguity levels.

5. Frontier Advancing Polygonization

After identifying reliable points, an initial mesh is first constructed around a randomly chosen reliable point, say, c. Essentially, the polygonization algorithm has to determine which of c's neighbors should be considered as *mesh*

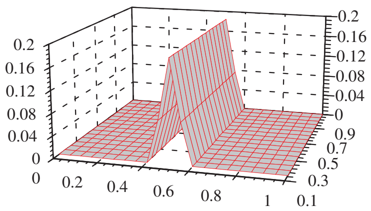


Figure 3. Graph showing the ambiguity level of points lying near an obtuse edge. Ambiguity level peaks at the edge and drops to 0 on the flat surfaces.

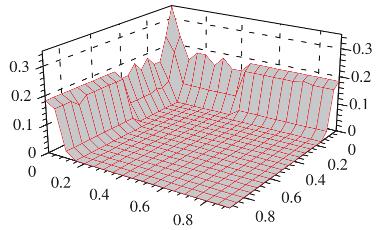


Figure 4. Graph illustrating the ambiguity level of points lying near a sharp corner. Ambiguity level peaks at the corner, drops to a lower value along the edges, and further drops to 0 on the flat surfaces.

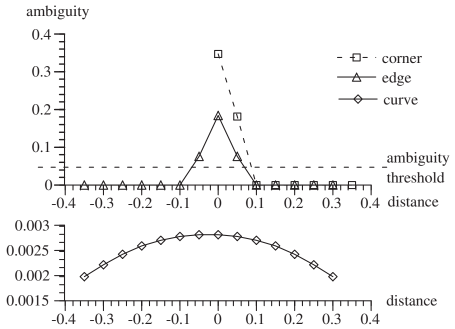


Figure 5. Comparison of the ambiguity levels of the points on three surfaces. This graph plots the ambiguity level with respect to the distance from the point with the highest ambiguity. For the Gaussian surface, the Gaussian peak has the largest ambiguity (Fig. 2). For the obtuse edge, points along the edge are most ambiguous (Fig. 3). For the corner data, the corner point is most ambiguous (Fig. 4). Points above the ambiguity threshold are considered ambiguous.

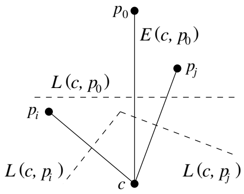


Figure 6. The edge $E(c, p_0)$ connecting a sample point c with its nearest neighbor p_0 must be a Delaunay edge. Otherwise, other points would be nearer than p_0 is to c.

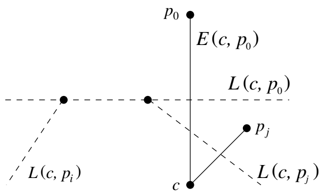


Figure 7. The two Voronoi vertices nearest to $E(c, p_0)$ must fall on the opposite sides of $E(c, p_0)$. Otherwise, other points would be nearer than p_0 is to c.

points, i.e., points to be connected to c to form the mesh triangles. The basic idea is based on the following observations (assuming that the points lie on a plane and are located in general positions, i.e., in any local neighborhood, not all the points in the neighborhood are co-linear):

1. The edge $E(c, p_0)$ connecting a sample point c with its nearest neighbor p_0 must be a Delaunay edge and its perpendicular bisector $L(c, p_0)$ must contain the corresponding Voronoi edge. Otherwise, there must be other neighbors p_i whose perpendicular bisectors $L(c, p_i)$ exclude $L(c, p_0)$ from the Voronoi cell at c (Fig. 6). So, the first step in the mesh construction is to connect a point with its nearest neighbor.

2. Given Observation 1, the two Voronoi vertices nearest to $E(c, p_0)$ must fall on the opposite sides of $E(c, p_0)$. Otherwise, there must again be other neighbors p_i whose perpendicular bisectors $L(c, p_i)$ exclude $L(c, p_0)$ from the Voronoi cell at c (Fig. 7). Observation 2 is used together with the next observation.

3. Given a known Delaunay vertex p_i and the corresponding Voronoi vertex v_i and bisector $L(c, p_i)$, the next Voronoi vertex v_{i+1} is given by the intersection between the bisectors $L(c, p_i)$ and $L(c, p_{i+1})$, and v_{i+1} is nearer than other intersections are to v_i. Otherwise, the bisectors of other sample points would exclude

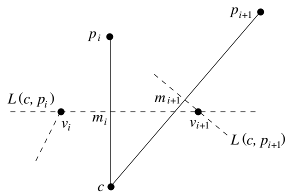


Figure 8. The next Voronoi vertex v_{i+1} is given by the intersection between the bisectors $L(c, p_i)$ and $L(c, p_{i+1})$ and v_{i+1} is nearer than other intersections are to v_i.

$L(c, p_{i+1})$ from the Voronoi cell at c (Fig. 8). Observations 2 and 3 together state that after determining the two Voronoi vertices on the opposite sides of the Delaunay edge $E(c, p_0)$, we can determine the remaining Voronoi vertices (and the corresponding Delaunay vertices) by going around c starting from one side of $E(c, p_0)$ and ending on the other side.

These three observations give an informal proof that the frontier advancing algorithm performs 2D Delaunay Triangulation of 3D points lying on a plane in 3D space.

5.1. Algorithm

The algorithm for constructing a mesh around a sample point can now be summarized as follows:

A2: Constructing A Mesh Around A Point
Given a point c and its neighbors,
  Set p_0 as the nearest neighbor of c.
  Find the point p_1 whose bisector $L(c, p_1)$ intersects $L(c, p_0)$ at the location v_1 nearest to the mid-point between c and p_0.
  Repeat for $i \geq 1$,
    Find the point p_{i+1} whose bisector $L(c, p_{i+1})$ intersects $L(c, p_i)$ at the location v_{i+1} nearest to v_i.
  Until $p_n = p_0$.
  Connect the Delaunay vertices p_i, $i = 1, \ldots, n$, and c to form a mesh.

The intersections of $L(c, p_i)$ and $L(c, p_{i+1})$ can be easily computed as the intersections of three planes, namely the tangent plane at c and the perpendicular planes containing $L(c, p_i)$ and $L(c, p_{i+1})$ and perpendicular to the edges $E(c, p_i)$ and $E(c, p_{i+1})$, respectively (Figs. 8, 9):

$$\begin{aligned} (\mathbf{x} - \mathbf{c}) \cdot \mathbf{n} &= 0 \\ (\mathbf{x} - \mathbf{m}_i) \cdot \mathbf{n}_i &= 0 \\ (\mathbf{x} - \mathbf{m}_{i+1}) \cdot \mathbf{n}_{i+1} &= 0 \end{aligned} \tag{1}$$

where $\mathbf{x}$ is a variable in 3D space, $\mathbf{c}$ is the vector form of point c, $\mathbf{m}_i$ is the mid-point along the edge connecting c

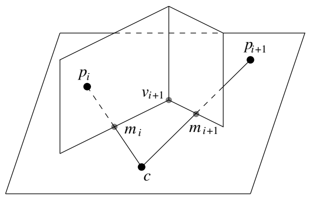


Figure 9. The intersection of three planes determine the location of v_{i+1}.

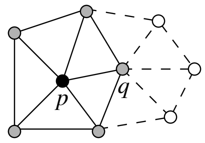


Figure 10. Mesh construction. The initial mesh (solid lines) is constructed around a reliable point p (black dot). The mesh points (gray dots) of p form the frontier of the initial mesh. Subsequently, the frontier is advanced by extending the mesh (dashed lines) and completing the mesh around the frontier point q.

and p_i, $\mathbf{n}$ is the unit normal vector of the tangent plane at c, and $\mathbf{n}_i$ is the unit normal vector from c to p_i. The tangent plane's normal vector $\mathbf{n}$ is estimated using PCA as the third eigenvector $\mathbf{e}_3$ of the set of point around c.

The mesh formed by connecting the Delaunay vertices p_i to the sample point c constitutes a Delaunay triangulation of the points in a plane. Once the initial mesh is constructed, the polygonization algorithm advances the mesh frontier by extending the mesh around the frontier points (Fig. 10). The same algorithm A2 is used to construct the mesh around a frontier point except that neighbors that are already included in the mesh are retained and never removed. The algorithm just adds triangles to complete the mesh around a frontier point.

Before applying algorithm A2, however, the normal vector of the tangent plane at an ambiguous frontier point must be re-estimated because the initial estimate given by PCA may not be accurate. The normal vector $\mathbf{n}$ of an ambiguous point p is computed as a distance-weighted average of the normal vectors $\mathbf{n}_i$ of reliable neighbors or neighbors whose normals have already been re-estimated:

$$\mathbf{n} = \frac{1}{\sum_i w_i} \sum_i w_i \mathbf{n}_i \ . \tag{2}$$

The weight w_i is inversely proportional to the distance d_i between the ambiguous point p and its neighbor p_i, i.e., $w_i = d_i^{-1}$.

The frontier advancing polygonization algorithm can now be summarized as follows:

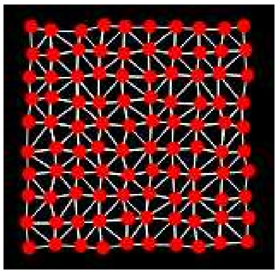
A3: Advancing Mesh Frontier
While there are free reliable points,
  Construct an initial mesh around a randomly selected free reliable point.
  For each reliable frontier points c,
    Complete the mesh around c using algorithm A2.
For each ambiguous frontier point c in increasing order of ambiguity,
  Re-estimate the normal vector of the tangent plane at c.
  Complete the mesh around c using algorithm A2.

Algorithm A3 constructs the mesh starting from a randomly selected free reliable point, i.e., a reliable point that is not connected to any mesh. It then advances the mesh frontier by completing meshes around reliable frontier points. This process continues until meshes have been constructed around all reliable points, except for reliable points on the edges of opened boundaries. At this time, the frontiers that can be extended are located only at ambiguous frontier points. Next, the algorithm further extends the frontiers by completing the meshes around ambiguous frontier points in increasing order of ambiguity. If 3D points along the edges and at the corners are sampled, the algorithm will form an edge at the meeting place of two advancing frontiers, and a corner at the place where more than two frontiers meet. Otherwise, the frontiers will meet at the most ambiguous sample points and produce approximations of edges and corners at these points. This polygonization method thus reduces the error in constructing surface discontinuities.

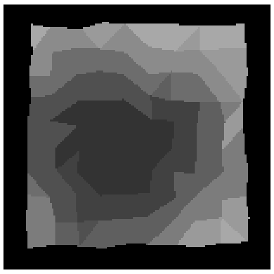
In each mesh completion step, algorithm A2 is guaranteed to add Delaunay edges to the mesh frontiers if the sample points lie on a plane. Otherwise, the edges added are not exactly Delaunay edges. However, the closer the curved surface is to a plane, the closer are the edges to the true Delaunay edges. A smooth curved surface can be approximated by a piecewise planar ptches. Therefore, for 3D points lying on a curved surface, the algorithm produces a Delaunay Triangulation of a piecewise planar approximation of the curved surface.

5.2. Test Results

Tests were performed to assess the performance of the frontier advancing polygonization algorithm on five sets of test data. The first and second sets contain 100 random points, lying on a plane and on a curve respectively. The third set contains 270 random points lying near a corner. The fourth set contains the standard data points for a mannequin model which consists of 12772 points. The last set contains the standard data points for a foot model which consists of 20021 points.



(a)



(b)

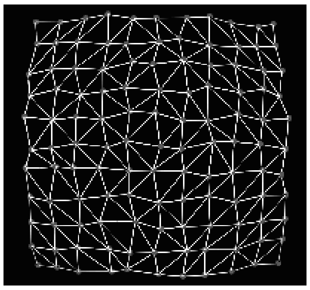
Figure 11. (a) The mesh constructed for random points lying on a plane. (b) Darker surfaces are constructed earlier while brighter surfaces are constructed later.

Figures 11 and 12 show the results of polygonizing random points lying on a plane and a smooth curved surface. The regularity of the triangles in the figures indicates that the algorithm indeed performs a Delaunay triangulation of the points. The shaded mesh in Fig. 11(b) shows the advancement of mesh frontiers from darker regions to lighter regions. Figure 13 shows the result of constructing the mesh for random points lying near a corner. The regularity of the mesh again indicates that Delaunay triangulation was performed on the surfaces. Moreover, the edges and the corner are correctly constructed. The shading of the mesh triangles reveals the advancement of mesh frontiers and meeting of frontiers at surface discontinuities.

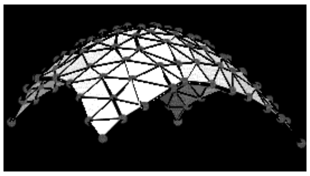
Polygonization results for standard data points of a mannequin, a foot model and a teapot model are shown in Figure 14 , 15 and 16 respectively. The shading of the mesh triangles again reveals the advancement and meeting of frontiers at surface discontinuities.

6. Conclusions

This paper presented a method for polygonizing non-uniformly distributed 3D points recovered from image sequences. The polygonization algorithm constructs the mesh

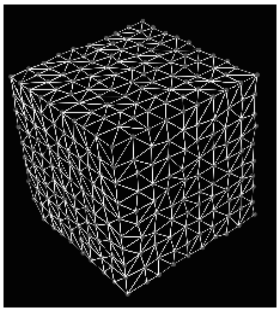


(a)

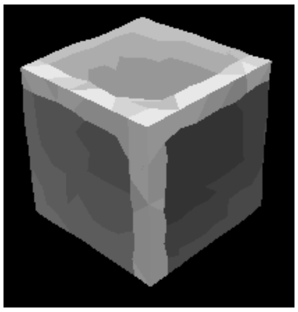


(b)

Figure 12. (a) Top view of the mesh constructed for random points lying on a curved surface from the top view. (b) A side view of the reconstructed mesh.

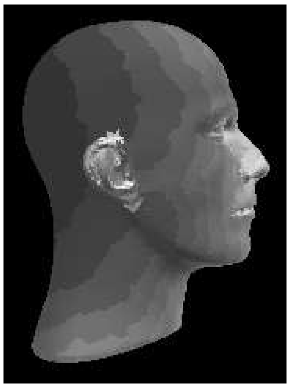


(a)

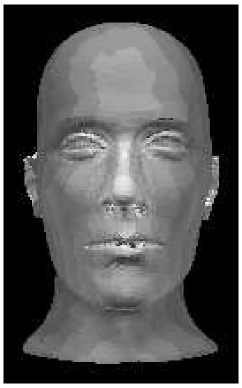


(b)

Figure 13. Polygonization results of random points lying near a corner. The shading of the mesh triangles reveals the advancement of mesh frontiers and meeting of frontiers at surface discontinuities.

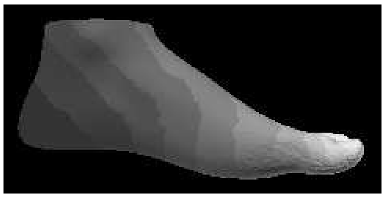


(a)

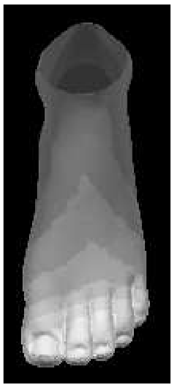


(b)

Figure 14. Results of polygonizing standard data points of a mannequin model. (a) Shading of the side of the mannequin model shows the advancement of mesh frontier (from dark to bright). (b) The front side of the mannequin model illustrates that the eyes, nose and mouth of the mannequin were constructed later due to their high ambiguity.



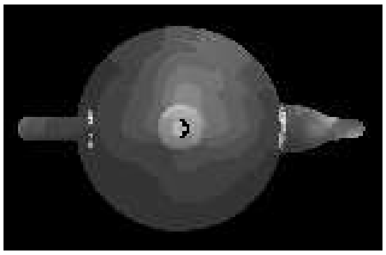
(a)



(b)

Figure 15. Results of polygonizing standard data points of a foot model. (a) Shading of the side of the foot model shows the advancement of mesh frontier (from dark to bright). (b) The front side of the foot models illustrates that the toes have higher level of ambiguity and hence were reconstructed last (light shade).

(a)



(b)

Figure 16. Results of polygonizing standard data points of a teapot model. (a) Shading of the top of the teapot model shows the advancement of mesh frontier (from dark to bright). (b) The side of the teapot model illustrates that the beginning of handle and the spout have higher level of ambiguity and hence were reconstructed last (light shade).

by advancing the mesh frontiers from reliable points lying on smooth and relatively flat surfaces to ambiguous points distributed near surface discontinuities.

In contrast to existing Delaunay triangulation algorithms, the frontier advancing algorithm only adds triangles to the mesh and never removes triangles. For 3D points lying on a plane, the algorithm has been proved to produce a Delaunay triangulation of the points. For 3D points lying on a smooth curved surface, the mesh constructed would be a Delaunay triangulation of the projections of the points onto a best fitting plane. The algorithm can also detect and construct surface discontinuities. If 3D points are sampled along the edges and at the corners, the algorithm will form an edge where two advancing frontiers meet, and a corner where three or more frontiers meet. Otherwise, the meeting frontiers would still approximate the edges and corners. Experimental results show that the method presented in this paper is effective for constructing meshes from non-uniformly distributed 3D points lying on surfaces with discontinuities.

References

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

[2] J. D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, 1998.

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

[4] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry, Algorithms and Applications*. Springer-Verlag, 1997.

[5] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. on Information Theory*, 29:551–559, 1983.

[6] H. Edelsbrunner and E. P. Mucke. Three-dimensional alpha shapes. *ACM TOG*, 13:43–72, 1994.

[7] T.-P. Fang and L. A. Piegl. Delaunay triangulation using a uniform grid. *IEEE Computer Graphics and Applications*, 13(3):36–47, 1993.

[8] P. J. Green and R. R. Sibson. Computing Dirichlet tessellations in the plane. In *Comp. J.*, pages 168–173, 1978.

[9] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of Delaunay and Voronoi diagrams. *Algorithmica*, 2:381–413, 1992.

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

[11] C. L. Lawson. Transforming triangulation. *Discrete Math*, 3:365–372, 1972.

[12] C. Oblonsek and N. Guid. A fast surface-based procedure for object reconstruction from scattered points. *Computer Vision and Image Understanding: CVIU*, 69(2):185–195, 1998.