A Delaunay-Based Region-Growing Approach to Surface Reconstruction from Unorganized Points

Chuan-Chu Kuo [1] and Hong-Tzong Yau [2]

[1] National Chung Cheng University, ckuo@pcc-server.ccu.edu.tw
[2] National Chung Cheng University, imehty@ccunix.ccu.edu.tw

ABSTRACT

This paper presents a Delaunay-based region-growing (*DBRG*) surface reconstruction algorithm that holds the advantages of both Delaunay-based and region-growing approaches. The proposed *DBRG* algorithm takes a set of unorganized sample points from the boundary surface of a three-dimensional object and produces an orientable manifold triangulated model with a correct geometry and topology that is faithful to the original object. Compared with the traditional Delaunay-based approach, the *DBRG* algorithm requires only one-pass Delaunay computation and needs no Voronoi information because it improves the non-trivial triangle extraction by using a region-growing technique. Compared with the traditional region-growing methods, the proposed *DBRG* algorithm makes the surface reconstruction more systematic and robust because it inherits the structural characteristics of the Delaunay triangulation, which nicely complements the absence of geometric information in a set of unorganized points. The proposed *DBRG* algorithm is capable of handling surfaces with complex topology, boundaries, and even non-uniform sample points. Experimental results show that it is highly efficient compared with other existing algorithms.

Keywords: Surface reconstruction, Delaunay triangulation, Region growing

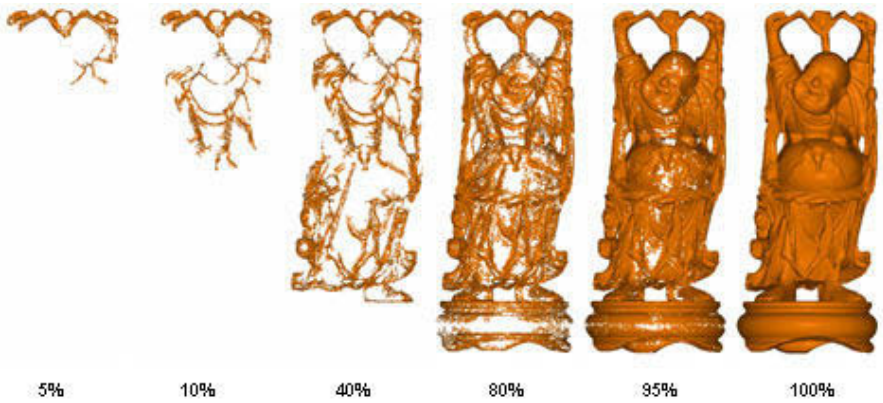


Fig. 1. A demonstration of the growing process of a large complex Buddha model that consists of 542546 points. Bottom line represents the achieved percentage of the growing progress.

1. INTRODUCTION

Surface reconstruction from a finite set of unorganized points have attracted much attention in the past decade and are becoming increasingly important in geometric modeling and related applications such as reverse engineering, computer vision, computer graphics and virtual reality. In the past, there were several algorithms that have been proposed to reconstruct surfaces with arbitrary topology. We distinguish them into three main categories: Delaunay-based, implicit surface and region-growing approaches.

For the Delaunay-based approach, the typical procedure consists of two steps: 1) First, it builds a geometric structure from a finite set of unorganized sample points via the Delaunay triangulation or Voronoi diagram; 2) it then extracts a collection of facets from such a geometric structure to approximate the actual surface. The early reconstruction method of Boissonnate [1], the well-

known alpha shapes of Edelsbrunner et al. [2], the crust and power crust algorithms of Amenta et al. [3],[4], the co-cone algorithm of Dey et al. [5],[6], and the reconstruction algorithm of Yau and Kuo [7],[8] generally fall into this category. The main advantage of the Delaunay-based approach is that the structural characteristics of the Delaunay triangulation and Voronoi diagram nicely complement the absence of geometric information in a set of unorganized points. Accordingly, it is more systematic and robust than other approaches. To our knowledge, algorithms with theoretical guarantees all fall into this category [4-7]. However, the expensive computation of the Delaunay triangulation was always denounced in the past. Besides, the extraction process based on the Delaunay/Voronoi structures is not simple and trivial. Consequently, the Delaunay-based method does not only require the Delauany triangulation but also the Voronoi diagram [3],[4]; it even needs multiple Delaunay computations [3],[4],[7],[8], which are quite time-consuming.

In a different approach, the region-growing method begins by initiating a triangle as an initial region and iterates to attach new triangles only on the region's boundaries. Such approach includes the early surface-based algorithm of Boissonnate [1], the graph-based approach of Mencl and Muller [9], the ball-pivoting algorithm (BPA) of Bernardini et al. [10], the projection-based triangulating techniques of Gopi and Krishnan [11], the interpolant reconstruction method of Petitjean and Boyer [12], the advancing-front algorithm of Hung and Menq [13], and recently the greedy Delaunay-based algorithm of Cohen-Steiner [14]. Although these methods are extremely fast due to keeping the Delaunay computation off [1],[9-13], there exists a common drawback that the reconstruction quality heavily depends on the user-defined parameters, which vary with the sampling density and cannot be assigned easily. In addition, the region-growing method may still leave behind small holes after the growing procedure when poor data (for example, noises) exists, therefore it cannot guarantee the creation of a closed manifold model if no appropriate hole-filling post-processing procedure is used. Among these algorithms, only [14] grows a triangulated surface from the Delaunay triangulation, which is similar to our proposed approach in this paper. However, in [14], the growing criterion, the smallest ball passing through the vertices of a Delaunay triangle and enclosing no sample points, proposed by [14], is exactly borrowed from the concepts of the pivoting ball [10] and the empty ball [12]; therefore, it does not only require the Delaunay triangulation but also the Voronoi edge, which is a disadvantage of the Delauny-based approach.

For the implicit surface approach, a signed distance function defined from sample points is first defined and computed, and then uses a zero-set of the signed distance function to construct an approximate triangulated surface with topology as the actual surface. Such approach has been applied to the surface reconstruction problem by Hoppe et al. [15], Curless and Levoy [16], Bajaj et al. [17], and Boissonnate and Cazals [18]. Recently, Beatson et al. [19] fits a radius basis function (RBF) to the signed distance function and uses RBF to polygonize sample points to create a triangulated surface. Compared with the Delaunay-based and the region-growing methods, the implicit surface approach approximates rather than interpolates sample points and therefore limits its applications only to computer graphics and virtual reality. In CAD/CAM and coordinate metrology applications [8],[20], however, accuracy of model representation is sometimes more important.

In this paper, we present a Delaunay-based region-growing (*DBRG*) algorithm that maintains the advantages of both Delaunay-based and region-growing approaches. The proposed *DBRG* algorithm first computes a Delaunay triangulation of a set of sample points, and then performs a region growing process to "grow" into a triangulated surface from the Delaunay structure. The growing process starts from initiating a triangle as an initial region and iterates to attach new triangles only to the region's boundary edges. In each iteration, the region expands by incrementally adding new triangles. Figure 1 demonstrates the growing process of a large complicated Buddha model, which is resulted from our proposed *DBRG* algorithm. Figure 2 illustrates the idea and overall procedure of our *DBRG* algorithm. Although the Delaunay computation was considered time-consuming in the past, but recently it has been improved quite substantially. In our experience, the computational geometry algorithms library *CGAL* [21] is now able to compute the 3D Delaunay triangulation of 15000 points per sec on a 1.2 GHz AMD CPU, which is extremely fast and is adopted in our implementation.

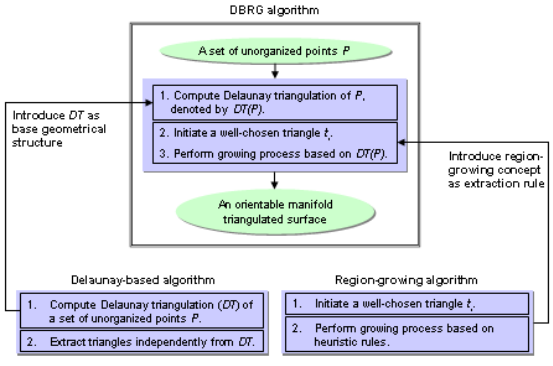


Fig. 2. The idea and overall procedure of the *DBRG* algorithm.

Compared with the traditional Delaunay-based approach, the proposed *DBRG* algorithm extracts triangles "incrementally" instead of "independently" from the Delaunay structure; therefore it can control the extraction easily and only requires single Delaumay computation, making the whole process more efficient. Compared with the traditional region-growing approach, the proposed *DBRG* algorithm inherits the structural characteristics of the Delaunay triangulation, which nicely complements the absence of geometric information in a set of unorganized points, making the surface reconstruction more systematic and robust. In other words, the proposed *DBRG* algorithm provides a systematic way to grow the triangulated surface, which is better than the traditional region-growing methods that rely on heuristic rules. Compared with the implicit surface approach, the proposed *DBRG* algorithm interpolates rather than approximates the sample points, therefore its output is guaranteed to pass through the original sample points, which is important in CAD/CAM and coordinate metrology applications [8],[20].

2. DATA STRUCTURES

The procedure of the *DBRG* algorithm, as shown in Figure 2, is to select the triangles from the Delaunay triangulation of sample points and incrementally insert them into the growing triangulated surface. In the proposed *DBRG* algorithm, therefore, there are two main data structures needed for storing the Delaunay triangulation and the growing triangulated surface.

2.1 Delaunay triangulation

The well-known Delaunay triangulation and its duality, Voronoi diagram, are becoming increasingly important and have found extensive applications in various fields [22]. A three-dimensional Delaunay triangulation, denoted by *DT*, is a tetrahedralization that divides the convex hull of sample points into a collection of tetrahedrons. A three-dimensional *DT* consists of four entities: vertices, edges, triangles and tetrahedrons. One of most important properties is that a three-dimensional *DT* is unique under the assumption that no 5 sample points can be found on a common sphere, which is one of the advantages of using *DT* in surface reconstruction from unorganized points.

In our implementation, we use the famous *CGAL* (Computational Geometry Algorithms Library) [21] to compute the three-dimensional *DT*. Therefore, a 3D-triangulation data structure provided by CGAL is adopted as the data structure of *DT* in this paper. With such data structure, all tetrahedrons and triangles incident to a given edge or vertex can be readily acquired, which provides a quick access to the necessary geometrical information in our region-growing operations.

2.2 Growing triangulated surface

During the region-growing process, an appropriate geometrical data structure is required for constructing and manipulating the growing triangulated surface, especially for quickly inserting triangles and ascertaining their local geometrical and topological correctness. A triangulated surface represents the boundary surface of a real object using a collection of triangles, which is a kind of boundary-representation (*B-Rep*) geometrical model. In the past, various data structures have been proposed for boundary representation in solid modeling, which include vertex-based, edge-based and face-based structures depending on which entity is given a central role. The edge-based data structures, such as winged-edge [23] and half-edge [24], have become most widely accepted due to the ability to represent both planar as well as curved polyhedral models.

However, the triangulated surface is the simplest one among boundary representations. In this paper, a simpler but practical *B-rep* data structure that explicitly represents triangle-edge and edge-vertex topologies is adopted to facilitate the manipulation of the triangulated surface during the growing process. Figure 3 illustrates the graph scheme of the data structure used in this paper with nodes corresponding to triangles, edges and vertices. Links between these nodes express topological or connectivity information. In this data structure, the triangulated surface is represented as a list of triangles. Each triangle is composed of three edges oriented in counterclockwise direction with respect to its normal, and each edge is composed of two vertices with each one storing the x, y and z coordinates. To efficiently manipulate the triangulated surface, each edge also records a list of triangles using this edge as one of its edges, and each vertex records a list of edges using this vertex as one of its endpoints. An edge is a boundary edge if its list of triangles only contains one. A boundary triangle means that at least one of its edges is a boundary edge. A manifold model is supposed to have two triangles at most in the list of triangles for each edge. With these explicit topological relations, it is easy to examine the orientation compatibility between adjacent triangles and access to the one-ring neighborhood of a vertex, possibly in terms of adjacent vertices or incident edges or faces. Figure 4 gives illustrations of these accesses. Figure 4(a) demonstrates that two adjacent triangles are compatible in their orientations since their common edge is in the opposite direction with respect to their normal, and Figure 4(b) illustrates the one-ring neighborhood of a vertex $\boldsymbol{v}$, which include adjacent vertices v_i, incident edges e_i and incident triangles t_i, $i = 1,\ldots,7$. In addition, we design a hash table for the list of

triangles to meet the dynamic triangulated surface, especially to facilitate the quick addition, deletion and access to triangles during the region-growing process.

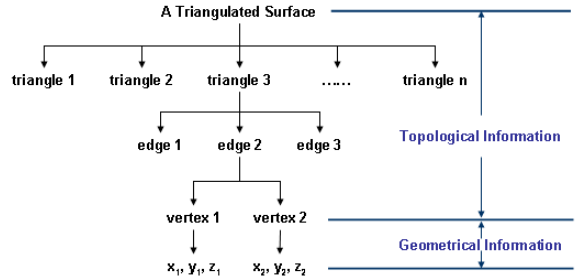
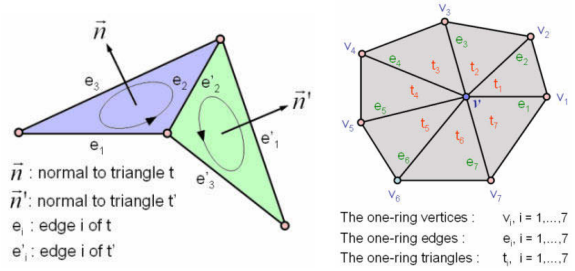


Fig. 3. The graph scheme of the data structure used in this paper.



(a) (b)

Fig. 4. Illustrations of: (a) the orientation compatibility between two adjacent triangles; (b) the one-ring neighbor of a vertex $\boldsymbol{v}$, which include adjacent vertices, incident edges and triangles.

3. THE PROPOSED DBRG ALGORITHM

As shown in Figure 2, our *DBRG* algorithm is composed of three procedures: Delaunay computation, initial triangle selection and region-growing operation. Let P denote the set of input sample points, $DT(P)$ the Delaunay triangulation of P, S the growing triangulated surface and ∂S the set of boundary edges of S. Notice that $S \subset DT(P)$since S is a collection of triangles chosen from $DT(P)$. Besides, in S and $DT(P)$, let v denote a vertex, e an edge, t a triangle and $r(t)$ the circumradius of t. Outlined as follows are the complete steps of the proposed *DBRG* algorithm.

Step 1: Compute $DT(P)$.
Step 2: Initialize the growing triangulated surface.
- *Step 2.1*: From $DT(P)$, select a triangle as an initial triangle t_{init} and add t_{init} into S.
- *Step 2.2*: For each boundary edge $e \in \partial S$, determine its candidate triangle $ct(e)$.
- *Step 2.3*: For each $ct(e)$, if $ct(e) \neq NULL$, assign a local smooth degree (*LSD*) to $ct(e)$ and put $ct(e)$ into a priority queue Q with the priority determined by *LSD*.

Step 3: Perform the region-growing process.
- *Step 3.1*: From Q, select a candidate triangle with top priority as the most credible candidate triangle *cct* and remove *cct* from Q. Examine the local geometry and topology of *cct* in S.
- *Step 3.2*: If *cct* is correct in both topology and geometry in S, insert *cct* into S and update ∂S. Otherwise, go to *step 3.1*.
- *Step 3.3*: For each new edge $ne \in \partial S$, determine its candidate triangle $ct(ne)$ with corresponding $LSD(ct(ne))$ and put $ct(ne)$ into Q with the priority determined by $LSD(ct(ne))$.
- *Step 3.4*: Iterate *steps 3.1, 3.2* and *3.3* until Q is empty.

With a good program for $DT(P)$, only steps 2 and 3 require detailed elaborations.

3.1 Initializations

3.1.1 Initial triangle

In step 2.1, a triangle is chosen from $DT(P)$ as an initial triangle to start the region-growing process. The first procedure is to pick up a vertex v_i with the maximum z coordinate from $DT(P)$ and then compute the circumradius for each incident triangle of v_i. A triangle with the minimum circumradius is selected as an initial triangle t_{init} and added into S. The normal to t_{init} has to be oriented as its z component is positive. In other words, the angle between the normal to t_{init} and the positive z direction must be less than $\pi/2$. At this moment, the three edges of t_{init} are all the boundary edges ∂S of S.

3.1.2 Candidate triangle

For a boundary edge $e \in \partial S$, it is associated with a candidate triangle $ct(e)$, which is determined as follows. With the 3d-triangulation data structure described in section 2.1, a set of incident triangles of e can be easily obtained from $DT(P)$. Let $T_{incident}(e)$ be the set of incident triangles of e and $t_{ncident}(e)$ one of $T_{incident}(e)$. Besides, let $t_b(e)$ represent the triangle using e as one of its edges in S ($t_b(e) \in S$ and $t_b(e) \in T_{incident}(e)$) and $\theta(t_{incident}(e), t_b(e))$ the dihedral angle between $t_b(e)$ and $t_{incident}(e)$, which $t_{incident}(e) \neq t_b(e)$. Notice that $t_b(e)$ and $t_{incident}(e)$ must be compatible in the computation of the dihedral angle. However, in the 3D Delaunay triangulation, the value of $\theta(t_{incident}(e), t_b(e))$ might be extremely large due to the existence of a sliver tetrahedron. A silver tetrahedron is a tetrahedron whose four vertices lie close to a plane and whose orthogonal projection to that plane is a convex quadrilateral with no short edge [25]. If a silver tetrahedron is encountered and therefore the value of $\theta(t_{incident}(e), t_b(e))$ is too large, saying greater than $5\pi/6$, then the $t_{incident}(e)$ cannot be selected as a candidate triangle $ct(e)$ to avoid the situation that the growing direction turns back and therefore creates a wrong geometry. Therefore, let θ_{sliver} denote a user-defined

parameter for being a threshold to avoid generating such wrong geometry, especially when $t_b(e)$ and $t_{ncident}(e)$ are the triangles of a sliver tetrahedron. The candidate triangle *ct(e)* associated with *e* is defined as follows.

$$ct(e) = \{ t \in T_{incident}(e),\ t \neq t_b(e) \mid r(t) \text{ is minimum and } \theta(t, t_b(e)) < \theta_{sliver} \}$$

In the existing surface reconstruction algorithms, different values of the similar silver parameter like θ_{sliver} are selected for dealing with sliver tetrahedrons [3],[5],[14]. In our implementation, θ_{sliver} is set to $5\pi/6$. Experiments show the value of θ_{sliver} is not critical. From step 2.2, a set of candidate triangles $T_{ct}(\partial S)$ are obtained by traversing the boundary edges. Figure 5 illustrates a three-dimensional example determining a candidate triangles *ct(e)* associated with $e \in \partial S$. In Figure 5, *S* is represented by the green region, ∂S the bold edges with counterclockwise directions and *e* is one of ∂S. From *DT(P)*, it is easy to access to the incident triangles of *e*, namely $t_1,\ldots,t_6$, and let $T_{incident}(e) = \{t_1,\ldots, t_6\}$. Notice that $t_b(e) = t_1$. The red triangle t_2 is the candidate triangle *ct(e)* since its value of *r(t)* is obviously minimal among $T_{incident}(e)$ and $\theta_t < \theta_{sliver}$.

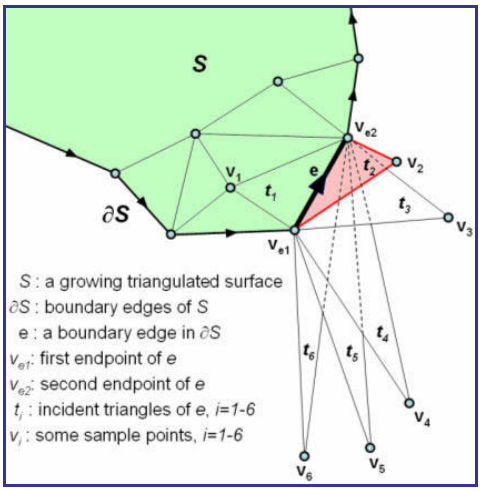


Fig. 5. A three-dimensional illustration of determining the candidate triangle for a boundary edge.

3.1.3 Local smooth degree

In each iteration during the growing process, only one of the candidate triangles $T_{ct}(\partial S)$ can be selected and inserted into *S*. To determine such candidate triangle, a greedy heuristic that follows [14] is used in this paper. The basic idea is that the most credible candidate triangle should be inserted into *S* first. In [14], the *c(t)* with the smallest value of $\theta(c(t), t_b(e))$ is inserted into *S* first. In other words, such *c(t)* would generate smoother surface in that local region and therefore it has higher credibility. In this paper, however, a more comprehensive parameter, *local smooth degree* (*LSD*), is defined to measure the credibility of a candidate triangle, which may represent a local smooth degree more appropriately for a triangulated surface. The definition of *LSD* is described as follows. Let nt_i be the *i*th neighboring triangle of *t* and $\theta(t, nt_i)$ the dihedral angle between *t* and nt_i. The *LSD* of *t* is defined as follows.

$$LSD(t) = \frac{1}{n}\sum_{i=1}^{n}\cos\left(\theta(t, nt_i)\right)$$

where *n* is the number of neighboring triangles of *t* and $1 \le n \le 3$.

From the definition of *LSD*, it is obvious that a candidate triangle with a higher value of *LSD* is more credible than one with a lower value of *LSD*. In step 2.3, we assign the local smooth degree, *LSD*, to each candidate triangle *ct(e)* and put *ct(e)* into a priority queue *Q* with the priority determined by *LSD(ct(e))*. Therefore, the candidate triangle with top priority in *Q* is the most credible candidate triangle.

3.2 Region Growing

During the region-growing process, new triangles are incrementally attached to ∂S. The determination of these new triangles and their incremental attaching and updating operations are clarified as follows.

3.2.1 Topological and geometrical examinations

As described in section 3.1.3, a candidate triangle with top priority in *Q* is selected as the most credible candidate triangle. Let *cct* denotes the most credible candidate triangle. Before *cct* is inserted into *S*, its local topology and geometry in *S* are examined to ensure that *S* is an orientable manifold model all the time.

- Topological examination

  A topological examination mainly checks that inserting a *cct* would not create a non-maniflod or a non-orientable manifold. An edge attaching operation is created when a *cct* is attached to ∂S. As shown in Figure 5, if t_2 is a *cct*, t_2 would be attached to *e* that is a common edge shared with t_1. With the above-mentioned appropriate data structure for the growing triangulated surface, this orientation examination is easily accomplished by checking the orientation compatibility between t_1 and t_2, as illustrated in Figure 4(a). Moreover, the edge attachment also needs to avoid creating a non-manifold, such as an edge shared by more than two triangles, as shown in Figure 6(a). In addition, the vertex status must be checked simultaneously for avoiding the creation of the neck vertex, as shown in Figure 6(b). If an incorrect topology is detected, it means that *cct* is an invalid triangle in *S*.

- Geometrical examination
  A geometrical examination of *cct* is performed after ensuring no topological incorrectness. Let nt_i be the *i*th neighboring triangle of *cct* in *S* and $\theta(cct, nt_i)$ the dihedral angle between *cct* and nt_i. The geometrical constraint is: $cos(\theta(cct, nt_i)) > cos(\theta_{sliver})$ for any existing neighboring triangle nt_i of *cct*. If the dihedral angle violates this geometrical constraint, it means that *cct* is an invalid triangle.

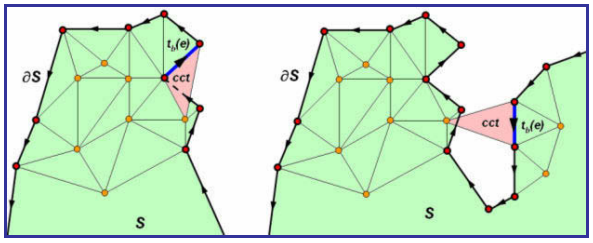


Fig. 6. Two typical incorrect topological operations of the most credible candidate triangle *cct*.

3.2.2 *Update and iteration*

Through the above-mentioned examinations, if *cct* is a valid triangle (both topology and geometry are correct in *S*), it is inserted into *S*. Otherwise, the growing procedure is interrupted and goes back to step 3.1 to select a new *cct*. After inserting *cct* into *S*, an update operation for ∂S is followed, as described in step 3.2. In other words, some new boundary edges are created and inserted into ∂S; some current boundary edges become non-boundary and must be removed from ∂S. In step 3.3, indeed, it is an update of *Q* by executing step 2.2 and 2.3 but only for new boundary edges. Continuing steps 3.1, 3.2 and 3.3 until *Q* is empty, *S* is the resulting triangulated surface of the *DBRG* algorithm.

3.3 Demonstrations

Figure 7 demonstrates the growing process of a torus model growing from 1 triangle to 2016 triangles. The blue line represents boundary edges and the dark region shows the interior of the torus model. A complete solid model (no blue line) is reconstructed when the number of triangles is 2016. A larger and more complicated Buddha model is demonstrated in Figure 1. From the growing process, it is obvious that the regions with high curvature are reconstructed first. Indeed, for the Buddha model, the sampling density is mostly higher in the regions with high curvature than with low curvature. Therefore, the regions with high curvature really have low value of *r(t)* and will be reconstructed first in the *DBRG* procedure.

4. OPEN SURFACE REONSTRUCTION

An open surface has apparent boundaries. In our *DBRG* algorithm, the growing process should be terminated on the boundaries if an open surface is reconstructed. Therefore, for the open surface reconstruction, the boundary detection is a critical procedure in the *DBRG* algorithm. Since the growing process described in section 3 uses candidate triangles to grow the triangulated surface, the selection rule of candidate triangles is directly related to the boundary detection. Through observations, during the growing process, the value of circumradii of candidate triangles will increase rapidly on the boundaries. This phenomenon helps quantitatively control the growing process on the boundaries. Let $r_a(S)$ represent the average of circumradius of all triangles in *S*. A candidate triangle *ct(e)* associated with a boundary edge *e* is discarded when $r(ct(e)) > M\ r_a(S)$, which *M* is a multiplication factor provided by users. In other words, for an open surface, the region-growing process is terminated when all candidate triangles are discarded. Figure 8 demonstrates the growing process of a hypersheet, in which open surface boundaries are detected and preserved.

Fig. 7. The growing process of a torus model. The number represents the number of triangles during the growing process.

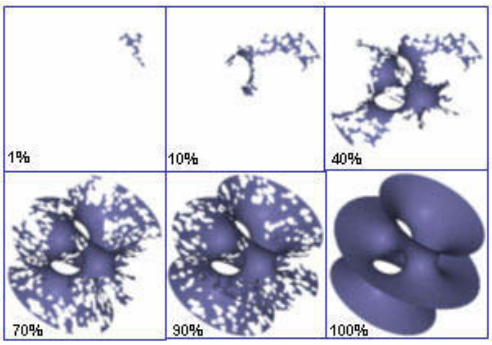


Fig. 8. The growing process of a hypersheet model. The number represents the achieved percentage of the growing progress

5. SMALL HOLES FILLING

In practice, S might still have small holes in regions with poor sample points such as noises. The small holes are retained when the boundary edges are not created by the open surface boundary detection. Let H denote a set of small holes and ∂H the boundary edges of H. Notice that ∂H is different from ∂S since ∂S is the open surface's boundaries that are detected and preserved by discarding the candidate triangles (section 4) and ∂H survived due to the existence of invalid candidate triangles (section 3.2.1). In this paper, we propose an *explicit recursive hole-filling method* as a post-processing step to fill such small holes. The hole-filling procedure is similar to the above-mentioned region-growing process, which incrementally creates new triangles and attaches them to ∂H in sequence.

As an illustrative example shown in Figure 9, our hole-filling procedure is addressed as follows. First, a small hole h is selected from H. Let ∂h and ∂v respectively represent a set of boundary edges and vertices of h. Second, ∂v is divided into two groups of vertices, namely concave vertices $V_{concave}$ and convex vertices V_{convex}. Let v_{i-1}, v_i and v_{i+1} be three sequential vertices on ∂v, $\vec{n}_{vi}$ the unit normal to v_i, which is computed by averaging the normal to incident triangles of v_i, and $\vec{n}_{cross} = \overrightarrow{v_{i-1}v_i} \times \overrightarrow{v_i v_{i+1}}$. A vertex is labeled as "concave" when the dot product of $\vec{n}_{vi}$ and $\vec{n}_{cross}$ is smaller than *zero*. An example is shown in Figure 9(a). Let v_1, v_2 and v_3 be three sequential vertices on ∂v. The vertex v_2 is labeled as "convex" because the dot product of $\vec{n}_{v2}$ and $\vec{n}_{cross}$ is greater than *zero*. In Figure 9(b), $V_{concave}$ are colored by blue points. Third, for each $v_{concave} \in V_{concave}$, its candidate triangle is created by $v_{concave}$ and its two adjacent vertices on ∂v, which would be used to fill this small hole. With these candidate triangles, finally, an incremental two-edge attaching procedure is performed following the steps 2.3 and 3.1-3.4, described in the *DBRG* algorithm. A difference, however, is that when the candidate triangle of $v_{concave}$ is invalid, all the filling triangles and the boundary triangles of ∂h are deleted from S and hence a new and larger hole h' is generated. Then, the hole-filling procedure is recursive to fill h' instead of h. In practice, such recursive deletion might lose some geometrical features of a real object. But it is reasonable that the regions with poor sample points are cannot be easily reconstructed except when those poor sample points are filtered out. Indeed, these deletions act like a noise-filtering process. Finally, H is filled and triangulized by repeating the hole-filing procedure for each small hole $h \in H$. In Figure 9(b), a small hole h, shown in Figure 9(a), is filled by the red triangles. Notice that t_1 is attached on ∂h earlier than t_7 since $r(t_1) < r(t_7)$.

Figure 10(a) is a complex dragon toy model, of which the sample points are obtained by merging six sets of data acquired from a laser range scanner. By using our *DBRG* algorithm, a resulting triangulated surface with some small holes colored by blue edges is shown in Figure 10(b). Through the *explicit recursive hole-filling* process, a complete solid model is reconstructed in Figure 10(c).

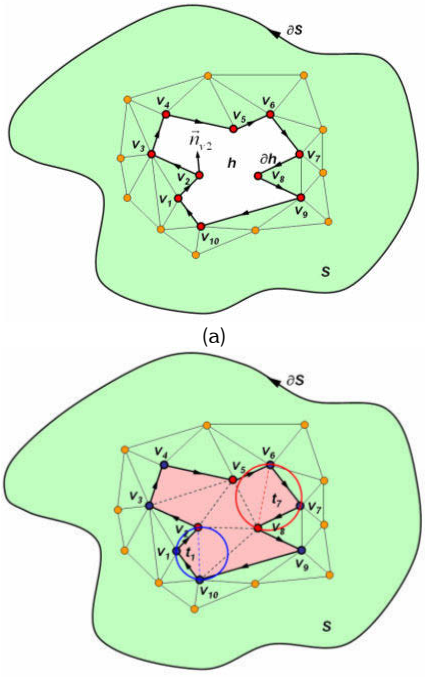
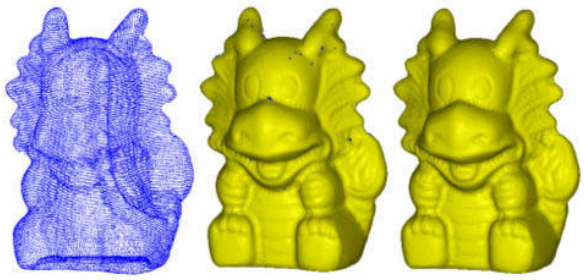


Fig. 9. An illustration of filling a small hole.



(a) (b) (c)

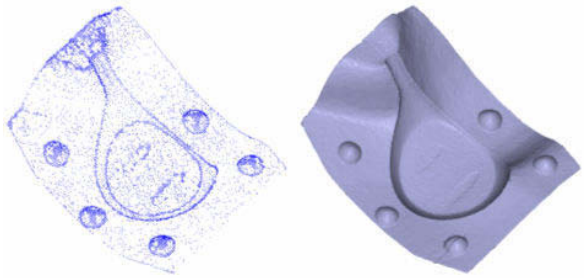
Fig. 10. A practical example of filling small holes. (a dragon toy model)

6. TREATMENT OF POOR DATA

6.1 Non-uniform distribution

The ability of handling sample points with uneven distribution is an important issue to the general surface reconstruction algorithm. Since the sampling requirement actually depends on the level of detail of the geometry, our DBRG algorithm is capable of dealing with the sample points with non-uniform distribution. Figure 11 demonstrates the surface reconstruction of a

golf model having the sample points with uneven distribution. Figure 11(a) is a non-uniform distribution case. Figure 11(b) shows the reconstruction from the non-uniform distribution case.

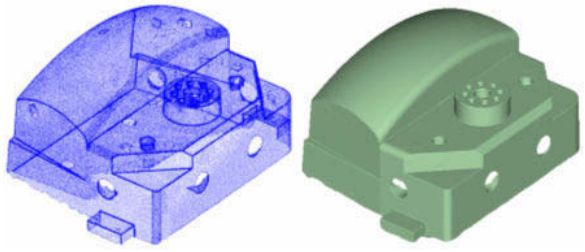


(a) (b)

Fig. 11. The reconstruction from the sample points with uneven distribution. (a golf mold)

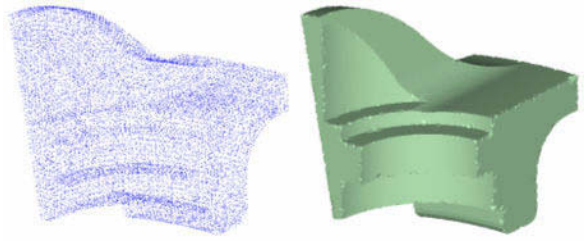
6.2 Sharp edges

Many existing surface reconstruction algorithms suffer from the fact that sharp edge features cannot be easily reconstructed. In practice, our *DBRG* algorithm can handle the reconstruction of sharp edges or corners only under the condition that the sampling requirement is satisfied, such as shown in Figure 12. Otherwise, as shown in Figure 13, a fan disk is reconstructed, but a post-processing step is needed to rectify the sharp edge features.



(a) (b)

Fig. 12. The reconstruction with clean and clear features. (a machine part)



(a) (b)

Fig. 13. The reconstruction with poor sharp features. (a fan disk model)

7. IMPLEMENTATION AND EXPERIMENTS

The proposed *DBRG* algorithm has been implemented in C++. We use the computational geometry algorithms library *CGAL* [21] to compute three-dimensional Delauny triangulation (*DT*). The *CGAL* provides several choices of number types [26]. We use floating point arithmetic, which is the fastest one provided by *CGAL*. In addition, Delaunay hierarchy method provided by *CGAL 2.4* is adopted by us for more efficient computation of Delaunay triangulation.

Model		*n*	*v*	*t*	DT(*n*)	RG	HF	TRT	Resulted Model
Closed surface	Bunny	34834	34834	69664	1.94	2.41	0.00	**4.35**	
	Golf club	209772	209451	418898	14.06	12.91	0.09	**27.06**	
	Buddha	543524	533523	1067070	43.68	36.12	0.24	**80.45**	
	Blade	882954	875656	1751540	75.03	60.40	2.08	**137.51**	
Open surface	Hyper sheet	6752	6746	12614	0.34	0.49	0.00	**0.83**	
	Nascar	20621	20620	40747	1.05	1.19	0.00	**2.34**	
	Lamp	23563	23497	46338	1.26	1.29	0.00	**2.55**	
	Bottle	186295	185941	370833	13.20	14.40	0.00	**27.60**	

n: number of input points; ***v***: number of output points; ***t***: number of output triangles;
DT(*n*): running time of Delaunay triangulation of *n* points (using *CGAL* 2.4);
RG: running time of region-growing process (sec);
HF: running time of hole-filling process (sec);
TRT: total running time (sec);
Platform: windows XP on a PC with 1.2 GHz AMD CPU and 512 MB memory (except with 768MB memory for the blade model)

Tab. 1. Experimental data of several surface reconstructions.

Table 1 demonstrates the experimental results of the closed and open surfaces using the proposed *DBRG* algorithm. The results show that the proposed *DBRG* algorithm is extremely fast compared with other existing algorithms. Notice that most of running time of hole-filling process is *zero*, even the maximum value is smaller than *2.1* second. The reason is that the hole-filling efficiency is extremely fast due to only few small holes. In addition, the ability of handling larger models is also demonstrated in Table 1, such as the blade model (882954 points).

8. CONCLUSION

A *DBRG* (***D****elauany-****B****ased* ***R****egion-****G****rowing*) surface reconstruction algorithm that holds the advantages of both Delaunay-based and region-growing approaches is presented. The input of the *DBRG* algorithm is an unorganized point set *P* and the output is an orientable manifold triangulated surface *S* that has a correct topology and geometry and is faithful to the original object. The *DBRG* algorithm uses the three-dimensional *DT(P)* as a combinatorial structure that consists of vertices, edges, triangles and tetrahedrons, and then incrementally extracts a collection of triangles from *DT(P)* as the resulting triangulated surface *S*. Such incremental extraction is carried out by using a region-growing approach that begins from initializing a triangle

as an initial region and iterates to attach new triangles only on the region's boundary edges. The output of the *DBRG* algorithm might contain small holes due to noises. An *explicit recursive hole-filling method* is developed for filling such small holes.

9. ACKNOWLEDGMENTS

The authors would like to thank the National Science Council in Taiwan for the support of this project. We thank the developed team of *CGAL* for publishing the source code. Thanks also go to the Cyberware, Inc. for providing the data of the golf club; Hughes Hoppe for the hyper sheet and the automobile exterior surface (nascar); the Stanford Data Repository for the bunny; the Buddha and the blade.

10. REFERENCES

[1] Boissonnat, J.-D., Geometric structures for three-dimensional shape representation, ACM Trans. Graph., Vol. 3, No. 4, 1984, pp 266-286.

[2] Edelsbrunner, H. and Mucke, E. P., Three-dimensional alpha shapes, ACM Trans. Graph., Vol. 13, No. 1, 1994, pp 43-72.

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

[4] Amenta, N., Choi, S. and Kolluri, R. V., The power crust, Proceedings of Sixth ACM Symposium on Solid modeling and applications, Ann Arbor, Michigan, USA, 2001, pp 249-266.

[5] Dey, T. K., Giesen, J., Leekha, N. and Wenger, R., Detecting boundaries for surface reconstruction using co-cones, Intl. J. Comput. Graph. & CAD/CAM, Vol. 16, 2001, pp 141-159.

[6] Dey, T.K. and Goswami, S., Tight Cocone: A water-tight surface reconstructor, Proceedings of Eighth ACM Symposium on Solid modeling and applications, Seattle, Washington, USA, 2003, pp 127-134.

[7] Yau, H. T., Kuo, C. C. and Yeh, C.H., Extension of surface reconstruction algorithm to the global stitching and repairing of STL models, Computer-Aided Design, Vol. 36, No. 5, 2002, pp 477-486.

[8] Kuo, C. C. and Yau, H. T., Reconstruction of virtual parts from unorganized scanned data for automated dimensional inspection, Journal of Computing and Information Science in Engineering (JCISE), Transactions of the ASME, Vol. 3, No. 1, 2003, pp 76-86.

[9] Mencl, E. and Müller, H., Graph-based surface reconstruction using structures in scattered point sets, Proceedings of CGI'98 (Computer Graphics International), 1998, pp 298–311.

[10] Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G., The ball-pivoting algorithm for surface reconstruction, IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 4, 1999, pp 349-359.

[11] Gopi, M. and Krishnan, S., A fast and efficient projection-based approach for surface reconstruction, International Journal of High-Performance Computer Graphics, Multimedia and Visualization, Vol. 1, No. 1, 2000, pp 1-12.

[12] Petitjean, S. and Boyer, E., Regular and non-regular point sets: properties and reconstruction, Comput Geom Theory Appl., Vol. 19, 2001, pp 101-126.

[13] Huang, J. and Menq, C. H., Combinatorial manifold mesh reconstruction and optimization from unorganized points with arbitrary topology, Comp.-Aided Design, Vol. 34, No. 2, 2002, pp 149-165.

[14] David, C.-S., A greedy Delaunay based surface reconstruction algorithm, Res. Rep., INRIA, 2002.

[15] Hoppe, H., DeRose, T., Duchamp, T., McDonald, J. and Stuetzle, W., Surface reconstruction from unorganized points, SIGGRAPH '92, pp 71-78.

[16] Curless, B. and Levoy, M., A volumetric method for building complex models from range images, SIGGRAPH '96, 1996, pp 303-312.

[17] Bernardini, F., Bajaj, C., Chen, J. and Schikore, D., Automatic Reconstruction of 3D CAD Models from Digital Scans, Intl. J. on Comp. Geom. and Appl., Vol. 9, No. 4-5, 1999, pp 327-370.

[18] Boissonnat, J.-D. and Cazals, F., Smooth surface reconstruction via natural neighbor interpolation of distance functions, Proceedings of 16th ACM Symposium on Computational Geometry, 2000, pp 223-232.

[19] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C. and Evans, T. R., Reconstruction and representation of 3D objects with radial basis functions, SIGGRAPH'01, 2001, pp 67-76.

[20] Yau, H. T. and Kuo, C. C., Virtual CMM and virtual part for intelligent dimensional inspection, 2002 Japan-USA Symposium on Flexible Automation, 2002, pp 1289-1296.

[21] http://www.cgal.org/

[22] Aurenhammer, F., Voronoi diagrams - a survey of a fundamental geometric data structure, ACM Computing Surveys, Vol. 23, No. 3, 1991, pp 345-405.

[23] Baumgart, B. G., A polyhedron representation for computer vision, National Computer Conference, AFIPS, Anaheim, CA, 1975, pp 589–596.

[24] Weiler, K., Edge-based data structures for solid modeling in curved-surface Environments, IEEE Computer Graphics and Applications, Vol. 5, No. 1, 1985, pp 21-40.

[25] Edelsbrunner, H. and Guoy, D., An experimental study of sliver exudation, Engin. with Computers, Vol. 18, 2002, pp 229-240.
[26] Boissonnat, J.-D., Devillers, O., Teillaud, M. and Yvinec, M., Triangulations in CGAL, Proceedings of 16th ACM Symposium on Computational Geometry, 2000, pp 11-18.