3D Visualization of Complex Surfaces Using Dynamic Delaunay Tessellation

M.L. Gavrilova and J. Pivovarov

Dept. of Computer Science, University of Calgary
Calgary, AB, Canada T2N1N4
{marina,juraj}@cpsc.ucalgary.ca

Abstract. The need for robust and reliable rendering mechanism is very high in the areas of computer simulation, biological modeling and computer graphics. Specific applications, such as modeling of growing forms, require tailored approaches to visualization. The paper describes the software developed for maintaining dynamic changes and visualization of complex 3D surfaces, utilizing the Delaunay tessellation as an underlying data structure. Some qualitative analysis of visualization is presented and the algorithm for skeletonization based on the developed model is discussed.

1 Introduction

The need for robust and reliable rendering mechanism in the areas of computer simulation, biological modeling and computer graphics is hard to underestimate. Due to particular specifics of each of the applications and expected outcomes requiring visualization, numerous techniques have been developed so far. For instance, a number of data structures such as TIN (Triangulated Irregular Networks), ROAM (Real-Time Optimally Adapting Meshes), quadtree-based meshes were developed for terrain visualization in GIS (Geographical Information Systems) [2,3,5,7]. Smoothing, culling and error metrics algorithms are applied to improve visual features of the model and achieve a more realistic representation [7,11].

In the area of image processing, the algorithms for investigating connectivity problems and background skeletonization have been intensively investigated. They rely on a number of techniques, ranging from peeling to building medial axis using Voronoi diagram based methods [1]. These methods often serve as basic algorithms for research on template matching and morphing, which in turn require efficient visualization techniques for algorithm verification.

In solid modeling, a number of problems that typically addressed are: how to represent a model in the computer, how to construct or reconstruct a surface from discrete point data, and how to extract features of a model. None of these are possible to address without appropriate visualization techniques [8,9,13].

In the following, we describe the specifics of the software developed for 3D dynamic Delaunay tessellation visualization and discuss the variety of problems that this software will assist in solving.

2 Relevant Research

Connections have been growing recently between the modeling community and the computational geometry. Many questions in molecular modeling can be understood geometrically in terms of arrangements of spheres in two and three dimensions. Problems include computing properties of such arrangements such as their volume and topology, testing intersections and collisions between molecules (particles), finding offset surfaces (related to questions of accessibility of molecule subregions to solvents such as water), data structures for computing interatomic forces and performing molecular dynamics simulations, and computer graphics algorithms for rendering models of growing forms accurately and efficiently (taking advantage of their special geometric structure). One of the very relevant areas is morphogenesis. Methods for the analysis of and comparison of growth forms are very important in this area, while the 3D methods seem to still remain an underexplored field [16].

One of most commonly used data structures, possessing a number of important properties from both geometrical and application-oriented perspectives and can be applied to the above problem, is the generalized Voronoi diagram and its dual, Delaunay triangulation (tessellation). Shortly, the structure represents the proximity information of a set of entities (points, objects) in a general d -dimensional space. The subdivision of a space by Voronoi regions provides a convenient background for point location, nearest-neighbor queries and studying of the specific physical properties of the modeled systems, such as distribution of voids in molecular biology, grow patterns in growing systems modeling and properties of clusters in granular materials simulation [12,13]. The corresponding triangulation (tessellation) possesses properties of being regular, satisfying min-max criterion as well as a number of other local optimization criteria, which makes it a desired choice for surface representation, meshing and model visualization.

For instance, a solvent-accessible triangulated surface generator for molecular graphics and boundary element applications was developed by Zauhar [16]. A 3D Delaunay tessellation was built to study and visualize the structure of fluids by Medvedev [13]. The algorithm for the rapid display of three-dimensional Delaunay meshes on standard raster displays, without the use of special purpose graphics hardware, was devised by Karasick [10]. Method for visualization of 3D surfaces and curves, based on Delaunay property of the underlying tessellation, was devised by [Baker]. In this paper, we describe the first prototype implementation of the software that allows to build and dynamically maintain the 3D model of the growing entity (such as coral, for instance), compute the medial axis of the model and perform fast visualization.

3 Preliminaries

Consider a set of objects S in R^d .

Definition 1. A *generalized Voronoi diagram (GVD)* for a set of objects S in R^d is the collection of generalized Voronoi regions

$$GVor(P) = \{\mathbf{x} \mid d(\mathbf{x}, P) \leq d(\mathbf{x}, Q), \forall Q \in S - \{P\}\},$$

where $d(\mathbf{x}, P)$ is the distance function in R^d between point $\mathbf{x} \in R^d$ and object $P \in S$.

Thus, a generalized Voronoi region of a generalized Voronoi diagram of n sites is obtained as the intersection of $n - 1$ half-spaces. The boundary of the generalized Voronoi region consists of up to $n - 1$ facets, where each facet is a $d - 1$ dimensional face. A 1-dimensional face is called a generalized Voronoi edge, and a 0-dimensional face is called a generalized Voronoi vertex.

Definition 2. A *generalized Delaunay triangulation* is the dual of the generalized Voronoi diagram obtained by joining all pairs of sites whose generalized Voronoi regions share a common generalized Voronoi edge.

Definition 3. A *generalized Delaunay tessellation (DT)* corresponding to the generalized Voronoi diagram is a collection of d -dimensional simplexes such that for each generalized Voronoi vertex

$$\mathbf{v} = Vor(\mathbf{p}_1) \cap Vor(\mathbf{p}_2) \cap \dots \cap Vor(\mathbf{p}_{d+1})$$

there is a simplex $(\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{d+1})$ in the generalized Delaunay tessellation.

Fig. 1 illustrates the above concepts in 2D space.

Fig. 1. Point set (left) and corresponding Delaunay Triangulation (right).

In 3D, the Delaunay tessellation contains edges (1-dimensional) and facets (2-dimensional triangles). The Voronoi dual of a facet of DT is a segment (a Voronoi edge). For each edge of DT connecting two sites, the dual is a Voronoi facet, which is a part of the bisecting plane in 3D separating the two sites.

4 Software for Dynamic 3D Delaunay Tessellation

VisualizedT is a program that visualizes the 3D Delaunay tessellation. The latest version of the program was compiled with Microsoft Visual C++, although it also runs under UNIX environment. The 3D Delaunay tessellation is build by utilizing the incremental construction algorithm, first presented in [6]. The DETRI software is used for building the initial 3D Delaunay tessellation [4,14]. The state of the art

program is written to enable dynamic changes in the Delaunay tessellation, computing statistics and performing 3D visualization was created (Fig. 2).

Fig. 2. Edges of DT of a torus (left), surface of DT (right).

The original DETRI program was made to run in two modes, one of them is for computing the tessellation from scratch, and the other mode is for reading the cached tessellation from a binary file, and then doing some reporting with it. The program creates an initial state; one that is highly dependent upon static sizes of the inputs. To alleviate this assumption, we made the software more "object-oriented". The algorithm state was abstracted and stored in a C structure. Some of the functions were changed to work with this state, rather than a set of global variables. This has the advantage too that more than one model could be manipulated. Finally, the state can be left around as long as is necessary while someone "interacts" with the model by adding points etc.

In the original DETRI code, the initialization happened, then the 3D Tessellation was computed, then the clean up occurred. Thus it was difficult to run just the "creation" portion of the software in order to be left with a model to work with. In the new organization of an abstracted algorithm state, the algorithm can be paused before cleanup takes place, again to facilitate interaction with the model.

4.1 Description of Data Formats

Using the modified code for Delaunay tessellation involves running the program twice. The first time it is run, the 3D Delaunay tessellation is computed and saved in a binary file format. By convention, the set of points are stored in an ascii file named without an extension. The binary file produced is saved in a file named as the point set file, but with a dt extension. Ex: `file --> file.dt`. The second time the program is run, it reads the binary file containing the tessellation and creates an ascii file outputting one or more of the tetrahedra, triangles, edges and vertices. More specifically, the following four types of files are used:

file

This ASCII file contains a sequence of 3D points "x y z" on each line.

file.DT.fl

This ASCII file has a different format based on which options are specified in the -C "print ..." command-line option. Basically, it contains a mish-mash of tetrahedrons, triangles, edges and vertices.

- A tetrahedron is displayed as "T a b c d" where a, b, c, d are 1-based indexes into the list of points.
- A triangle is displayed as "f a b c" where a, b, c are 1-based indices into the list of points.
- An edge is displayed as "e a b" where a, b are 1-based indices into the list of points.
- Finally a vertex is "v a" where a is a 1-based index into the list of points. All the above objects are part of the 3D Delaunay Tessellation of the 3D Point Set.

file.info

This ASCII file has some stats from the DT computation. It contains statistics of how many tetrahedrons, triangles, edge and faces were found, time requires per each operation, a number of comparisons performed.

file.voronoi

This ASCII file contains the Voronoi edges of the dual Voronoi graph. The format is one line segment per line "x1 y1 z1 x2 y2 z2" where one segment exists for each pair of tetrahedrons that share a face. A circumscribing sphere is computed for each tetrahedron T1, T2 with centers $c_1 = (x_1, y_1, z_1)$ and $c_2 = (x_2, y_2, z_2)$.

Fig. 3. Running the VisualizeDT program.

4.2 VisualizeDT Program

The VisualizeDT program uses information from the files **file**, **file.DT.fl** and **file.Voronoi** in order to create visual representation of Voronoi diagram and Delaunay tessellation (see Fig. 3).

The VisualizeDT program has 4 modes of viewing (see Fig 4):

- the set of points, highlighted (mode 1),
- the edges in the tessellation (mode 2),
- the triangles/faces of the tessellation (mode 3) and
- the dual Voronoi edges of the tessellation (mode 4).

Fig. 4. Four modes of viewing in VisualizeDT program.

5 Dynamic Update of Delaunay Tessellation

The algorithm has been modified so that it can add extra points dynamically to the point set (see Fig. 5). The detailed algorithm description follows.

Algorithm for dynamic modification of the Delaunay tessellation and Voronoi diagram

1. (*Preprocessing*) Construct the Delaunay tessellation (using DETRI code).
2. (*Main loop*) Insert a new point in the Delaunay tessellation.
 - 2.1 Perform a point location to determine the tetrahedron T for a new point to be inserted. The inserted point splits T onto four new tetrahedra. This results in 8 new quadrilaterals (four based on faces and four on edges).
 - 2.2 Compute the inscribed sphere for the eight new quadrilaterals, using the function *Inscribed_sphere*.
 - 2.3 If the inscribed sphere is not empty, i.e. contains Voronoi sites of the original set S , *GOTO* 2.4
 Else GOTO 2.5
 - 2.4 Perform the *swap* operation. In 3D, the operation involves swapping the facet for an edge or an edge for a facet. For each of the newly introduced quadrilaterals (based on new facets and new edges), perform *Inscribed_sphere* test. *GOTO* 2.3
3. Update the corresponding Voronoi diagram. Compute dual Voronoi edges for all new DT facets and edges, using coordinates of the obtained inscribed spheres to determine the Voronoi vertices.

The complexity of the algorithm in the worst-case is $O(n^3)$. But on average, only a constant number of swaps is required at each step, i.e. $O(1)$ per insertion.

Fig. 5. Spherical surface (left), extra point dynamically added (right).

6 Algorithm for Building a 2D Medial Axis Based on a Voronoi Diagram

An algorithm to build the 2-dimensional medial axis for any randomly shaped object in the plane was developed and implemented. The algorithm also allows visualizing the Delaunay triangulation, Voronoi diagram and medial axis in 2D. The algorithm outline is as follows:

1. Compute the coordinates of a Voronoi edge for a given DT edge: find two adjacent triangles, find two corresponding Voronoi vertices.

2. Assign a weight to each DT edge XY using either approach a or b :

Approach a : use the edge length.

Approach b : compute the shortest path between X and Y along the edges of the input curve.

3. Start removing DT edges (and corresponding Voronoi edges) with smaller weights first. Use visualization to evaluate the termination condition.

The program then displays the Voronoi diagram, Delaunay triangulation and the medial axis. The medial axis shape can be drawn interactively or read from a file. In Fig. 6, the resulting medial axis is identified by blue font, the Delaunay triangulation edges by yellow font. The implementation of a 3D version of the algorithm is currently on its way.

Fig. 6. 2D Medial axis and Delaunay triangulation.

7 Conclusions

The efficient visualization of growing surfaces is an area, highly relevant to morphogenesis. The presented methods for visualization of 3D surfaces in the presence of dynamic updates are the first step on the way to a better understanding of complex biological structures, and will provide an efficient framework for testing new developments in that area.

References

- [1] Baker, T.J. (1989) Automatic Mesh Generation for Complex Three-Dimensional Regions Using a Constrained Delaunay Triangulation, *Engng. Comput.*, 5 p. 161.
- [2] Blow, J (2000) Terrain Rendering at High Levels of Detail, in *Proceedings of the Game Developers' Conference*, San Jose, California.
- [3] De Berg, M. and Dobrint, K. (1995) On levels of detail in terrains, *Proc. 11th ACM Symposium on Computational Geometry* June 1995.
- [4] DETRI code [www. Geom..umn.edu/software/cglist/GeomDir/#DETRI](http://www.Geom.umn.edu/software/cglist/GeomDir/#DETRI)
- [5] Duchaineauy, M, Wollinsly, M. (1997) ROAMing Terrain: Real-Time Optimally Adapting Meshes, *IEEE Visualization '97 Proceeding*.
- [6] Edelsbrunner, H. and Shah, N. (1996) Incremental topological flipping works for regular triangulations, *Algorithmica*, vol. 15, pp. 223–241.
- [7] Hesse, M and Gavrilova, M. (2003) Quantitative Analysis of Culling Techniques for Real-time Rendering of Digital Elevation Models, *WSCG*.
- [8] Kaandorp, J.A. and Sloot, P.M.A. (1997) Parallel simulation of accretive growth in 3-dimensions. *BioSystems* 44(3):181–192.
- [9] Kaandorp, J.A. and Sloot, P.M.A. (2000) Growth and form of sponges and corals in a moving fluid. In *Pattern formation in biology, vision and dynamics*, pp. 24–41, World Scientific.
- [10] Karasick, D. Lieber, L. R. Nackman, and V. T. Rajan (1997). Visualization of Three-Dimensional Delaunay Meshes. *Algorithmica*, 19(1-2):114–128.
- [11] Lindstrom, P and Pascucci, V. (2001) Visualization of Large Terrains Made Easy, *IEEE Visualization 2001*, California, October 21–26, 2001
- [12] Luchnikov, V.A, Gavrilova, M.L., Medvedev, N.N., Voloshin V.P. (2002) The Voronoi-Delaunay Approach for the Free Volume Analysis of a Packing of Balls in a Cylindrical Container, *Future Generation Computer Systems*, 18, pp. 673–679, Elsevier.
- [13] Medvedev, N.N. (2000) Voronoi-Delaunay method for non-crystalline structures, *SB Russian Academy of Science*, Novosibirsk.
- [14] Mücke, E., Saias, I. and Binahi Zhu (1996) Fast Randomized Point Location Without Preprocessing in Two- and Three-dimensional Delaunay Triangulations. *12th Annual Symp. on Comp. Geometry*.
- [15] Prusinkiewicz, P (1994) Visual Models of Morphogenesis. In *Artificial Life*, 1(1/2):67–74.
- [16] Zauhar, R.J. (1995) SMART: A solvent-accessible triangulated surface generator for molecular graphics and boundary element applications, *J. Computer-Aided Molecular Design* 9, 149–159.