Numerically Robust Algorithm for Constructing Constrained Delaunay Triangulation

Hiroshi Inagaki† Kokichi Sugihara††

†Toyota National College of Technology, JAPAN
e-mail: inagaki@tctice.toyota-ct.ac.jp

††Faculty of Engineering, University of Tokyo, JAPAN
e-mail: sugihara@simplex.t.u-tokyo.ac.jp

Abstract

In our previous paper, we proposed a numerically robust algorithm for constructing the three-dimensional Voronoi diagram. In that algorithm, the higher priority is placed on the consistency of the topological structure than on numerical values, so that, no matter how large numerical errors may take place, the algorithm never comes across topological inconsistency and thus can always complete its task.

In this paper, we show that our algorithm can be applied not only to the construction of the ordinary Delaunay triangulation but also to the construction of the constrained Delaunay triangulation. The Delaunay triangulation is the dual diagram of the Voronoi diagram, and the constrained Delaunay triangulation is its generalization. Both triangulations have many fields of applications. In order to construct the constrained Delaunay triangulation, we invent numerical judgments on the way to the construction of the Voronoi diagram in such a way that they satisfy the constraints. In spite of the invention of numerical judgments, our algorithm works well, because it is based on topologically consistent process.

Moreover we implemented the algorithm as a computer program, and verified the validity of the algorithm by computational experiments.

keywords: Voronoi diagram, constrained Delaunay triangulation, topological consistency, numerical robustness

1 Introduction

The Delaunay triangulation is the dual diagram of the Voronoi diagram, and it gives a triangulation of the convex hull for a finite set of points. The definition of the Delaunay triangulation can be generalized in such a way that prescribed edges are used in the triangulation. Such a triangulation is known as the constrained Delaunay triangulation [1][2][3]. This triangulation can be applied to many fields, such as mesh generation, interpolation, path planning and computer graphics [2][3][4].

In computational geometry, some efficient algorithms for constructing Voronoi diagrams and Delaunay triangulations have been proposed. However, these algorithms were designed on the assumption that numerical errors do not take place, and hence computer programs based on these algorithms often fail because of topological inconsistency due to computational errors.

This kind of the problem arises not only in the construction of these diagrams but also in many other geometric processings. Recently, the importance of numerical robustness has become widely recognized, and many approaches have been proposed [5][6][7][8].

In our previous paper [8], we proposed a numerically robust algorithm for constructing the three-dimensional Voronoi diagram. In the present paper, we modify the algorithm so that it can be applied not only to the construction of the ordinary Delaunay triangulation but also to the construction of the constrained Delaunay triangulation.

2 Algorithm for Voronoi Diagrams

Among many algorithms for the construction of the Voronoi diagram, a rather sophisticated implementation of the incremental-type algorithm is most practical [9]. It runs, on the average, in $O(n)$ time for n generators. The incremental-type algorithm starts with a simple Voronoi diagram for several generators, and modifies it by adding new generators one by one.

However, the incremental-type algorithm as well as other algorithms is unstable because of computational errors when degeneracy takes place or when the situation is very close to degeneracy.

In our previous papers [7][8], we re-designed the incremental-type algorithm so as to make it work well in finite-precision environment. In that algorithm, the higher priority is placed on the consistency of the topological structure than on numerical values, so that, no matter how large numerical errors may take place, the algorithm will never come across topological inconsistency and thus can always complete its task. The detail of that algorithm is described in [7] (two-dimensional case) and in [8] (three-dimensional case).

3 Algorithm for Constrained Delaunay Triangulations

3.1 Constrained Delaunay Triangulation

Let P be a set of a finite number of points in the plane, and C be a set of line segments whose terminal points belong to P and which do not intersect except at their terminal points. For any points p, q, r in P, we say that p is *visible* from the line segment $\overline{qr}$ if, for any point s in $\overline{qr} - \{q, r\}$ (i.e., s is a point on $\overline{qr}$ other than the terminal points), the line segment $\overline{ps}$ does not intersect any line segment in $C - \{\overline{qr}\}$. For two points p and q in P, the line segment $\overline{pq}$ is called a *constrained Delaunay edge* either if $\overline{pq}$ belongs to C, or both if $\overline{pq} - \{p, q\}$ does not intersect any element in C and if there is a circle passing through p and q that contains no point in P, visible from $\overline{pq}$, in its interior. All the constrained Delaunay edges give a partition of the convex hull of P into triangles. This partition is called the *constrained Delaunay triangulation* for P with the constraint set C. (This definition is valid only for the nondegenerate case; for the strict definition, see [2][3][4].) The *constrained Voronoi diagram* is defined as the dual of the constrained Delaunay triangulation.

In the three-dimensional space, on the other hand, a triangulation satisfying given constraints does not necessarily exist. If it exists, the three-dimensional constrained Delaunay triangulation is defined similarly.

3.2 Topologically Consistent Algorithm

The basic idea in our algorithm for constructing the constrained Delaunay triangulation is as follows. On the way to the construction of the ordinary Voronoi diagram, we invent the numerical judgments, so that the generators under constraint are adjacent. As a result, the obtained diagram may be far from the true Voronoi diagram. However, that diagram is still a planar graph; hence, it defines the dual diagram. That dual diagram is exactly the constrained Delaunay triangulation which satisfies given constraints. For example, let us consider the input data shown in Figure 1. The solid circles represent the generators, and the dashed line represents the constraint. Figure 2 shows the diagram obtained as a result of the invention mentioned above. In this figure the self-intersection can be seen, and hence it is far from the ordinary Voronoi diagram. Figure 3 shows the dual of the diagram in Figure 2. We can see that this diagram is the triangulation that satisfies the given constraint.

In spite of the invention of numerical judgments, our algorithm works well, because it is based on topologically consistent process. According to that process, numerical data, such as the locations of the Voronoi points, are considered to be not necessarily correct, but still no inconsistency takes place from the topological point of view.

The idea can be applied to both two-dimensional and three-dimensional constrained Delaunay triangulations. In the three-dimensional case, however, the solution which satisfies given constraints does not always exist. If the solution does not exist, the output of our algorithm includes overlapping tetrahedra. We can check this situation by comparing the volume of the convex hull of P with the total sum of the volumes of the tetrahedra; if they differ, the constrained Delaunay triangulation does not exist.

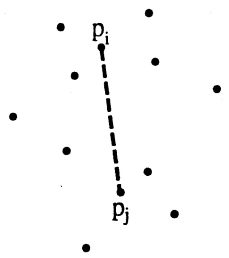

Figure 1. Generators and the constraint

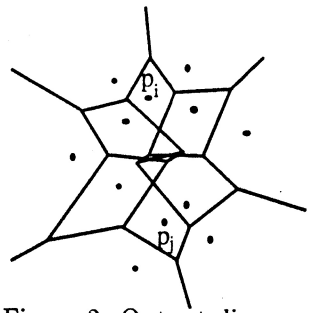

Figure 2. Output diagram including self-intersection

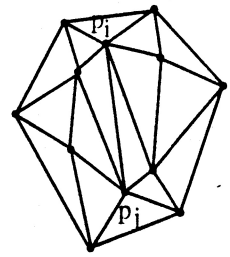

Figure 3. Dual diagram

4 Computational Experiments

We implemented our new algorithm for constructing the three-dimensional constrained Delaunay triangulation as a computer program. In this section, we show some outputs of this program.

4.1 Simple Example

In the first experiment, we execute our program for simple input data, and confirm its validity. Five generators are arranged on the vertices of the hexahedron shown in Figure 4.

First, we placed no constraint. As the output of the program, we obtained the Voronoi diagram shown in Figure 5. The output is represented in such a way that the three-dimensional structure can be perceived when the right picture is seen by the left eye, and the left picture by the right eye. From this Voronoi diagram, the Delaunay triangulation shown in Figure 6 is derived. In order to see the state of the decomposition easily, we illustrate the Delaunay tetrahedra separately in Figure 7 (each element is slightly moved). From this figure, we can see that the hexahedron is cut vertically into three pieces.

Next, for the same locations of generators, we placed the constraint that the triangle face in the middle of the hexahedron should be used (See Figure 8; the shaded triangle represents the constraint face). Figure 9 shows the obtained Voronoi diagram under the constraint. In this figure, we can find self-intersection caused by the constraint. From this Voronoi diagram, the dual diagram shown in Figure 10 is derived. In order to see the state of the decomposition easily, we illustrate the tetrahedra separately in Figure 11 (the same representation as Figure 7). From this figure, we can see that the hexahedron is cut horizontally into two pieces. Therefore the triangulation satisfies the given constraint.

4.2 Constrained Surface of Polyhedron

In the second experiment, the surface of a polyhedron was used as the constraint. We gave all vertices of the polyhedron shown in Figure 12 as generators, and all faces of the polyhedron as the constraint. Furthermore, we added 100 generators randomly inside and outside this polyhedron. Consequently, the generators are arranged as shown in Figure 13. In this figure, solid lines represent the constraint faces.

Now, we show the output for this input. Figures 14 and 15 show examples of the Voronoi regions (we do not show the whole Voronoi diagram because it is too complicated to understand). The Voronoi region shown in Figure 14 is near the constraint faces, so that it includes great disturbance due to the constraint faces. On the other hand, the Voronoi region shown in Figure 15 is far from the constraint faces, so that it seems to include no disturbance.

The constrained Delaunay triangulation derived from that Voronoi diagram is the triangulation where all of the faces of the polyhedron are chosen. In order to verify this fact, all the Delaunay tetrahedra whose centroid is out of the polyhedron were removed. Figure 16 shows the result of that operation (in order to see easily, only the visible side of the remaining triangulation is drawn). Judging from the result, all the faces of the polyhedron are chosen in the triangulation, so that the output satisfies the given constraint.

4.3 More Complicated Example

In the third experiment, the input size is much larger, and moreover the constraint faces are more complicated. The polyhedron used as the constraint is shown in Figure 17. We put all the vertices of the polyhedron into the generator set. Furthermore we add 1,500 generators inside and outside the polyhedron randomly. Consequently, the generators are arranged as shown in Figure 18.

Now, we show the output for these generators and the constraint. The output of our program is shown in Figure 19. In this figure, we removed the Delaunay tetrahedra which are outside the constraint polyhedron in the same way as in Section 4.2. Judging from the result, we can see that the output satisfies the given constraint.

5 Summary

This paper presents a new method for constructing the constrained Delaunay triangulation in the three-dimensional space. The main idea of our method is invention of numerical judgments in the construction so that the output has the desired topological structure. Note that our algorithm is constructed on the basis of the topologically consistent process; therefore it works well in spite of the invention of numerical results.

Besides we implemented the algorithm for constructing three-dimensional constrained Delaunay triangulation, and verified the advantage of it by computational experiments. The experiments show that our algorithm can be applied to many fields, such as mesh generation, interpolation, path planning and computer graphics.

This work is partly supported by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan (Grant No. 05780271), and The Naito Research Grant.

References

[1] Preparata, F. P. and Shamos, M. I.: "Computational Geometry — An Introduction", Springer Verlag, New York (1985).

[2] Aurenhammer, F.: "Voronoi Diagrams — A Survey of a Fundamental Geometric Data Structure", ACM Computing Survey, **23**, No. 3, pp. 345 - 405 (1991).

[3] Okabe, A., Boots, B. and Sugihara K.: "Spatial Tessellations — Concepts and Applications of Voronoi Diagrams", John Wiley & Sons, Chichester (1992).

[4] Bern, M. and Eppstein, D.: "Mesh Generation and Optimal Triangulation" in Du. D.-Z. and Hwang, F. K. (eds.) : "Computing in Euclidean Geometry", World Scientific Publishing Co., pp. 23 - 90 (1992).

[5] Hoffman, C., Hopcroft, J. and Karasick, M.: "Towards Implementing Robust Geometric Computations", Proc. 4th ACM Annual Conference on Computational Geometry, pp. 106 - 117 (1988).

[6] Milenkovic, V.: "Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic", Artificial Intelligence, **37**, pp. 377 - 401 (1988).

[7] Sugihara, K. and Iri, M.: "Construction of the Voronoi Diagram for One Million Generators in Single-Precision Arithmetic", Proc. IEEE, **80**, pp. 1471 - 1484 (1992).

[8] Inagaki, H., Sugihara, K. and Sugie, N.: "Numerically Robust Incremental Algorithm for Constructing Three-Dimensional Voronoi Diagrams", Proc. Fourth Canadian Conference on Computational Geometry, pp. 334 - 339 (1992).

[9] Ohya, T., Iri, M. and Murota, K.: "Improvements of the Incremental Method for the Voronoi Diagram with Computational Comparison of Various Algorithms", J. Operation Research Society of Japan, **27**, pp. 306 - 336 (1984).

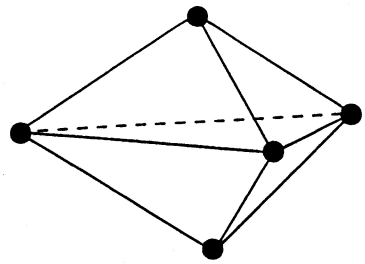

Figure 4. Input data without constraint

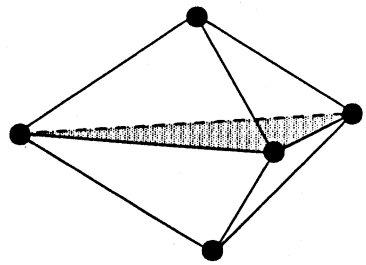

Figure 8. Constraint face

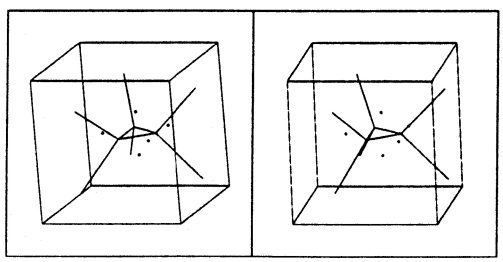

Figure 5. Voronoi Diagram without Constraint

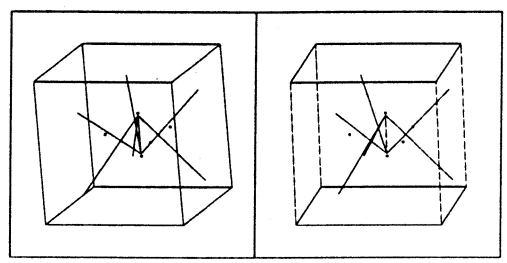

Figure 9. Voronoi diagram under constraint

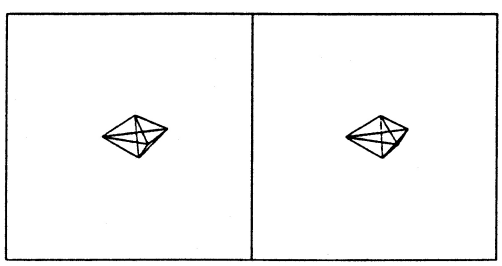

Figure 6. Delaunay triangulation without constraint

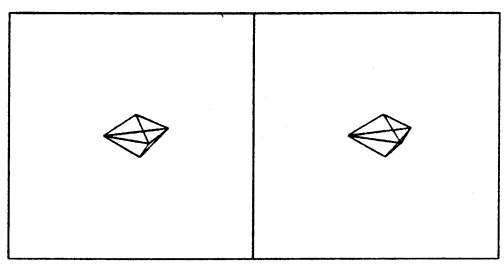

Figure 10. Delaunay triangulation under constraint

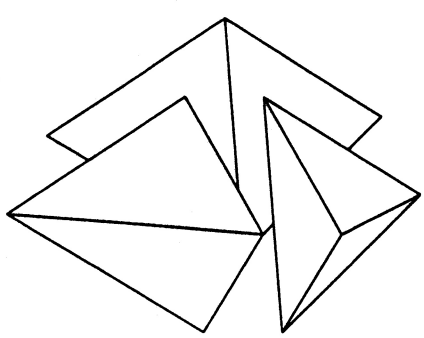

Figure 7. Decomposion without constraint

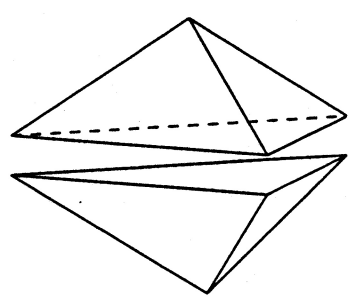

Figure 11. Decomposion under constraint

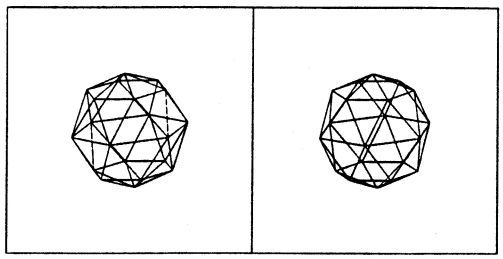

Figure 12. Polyhedron used as the constraint

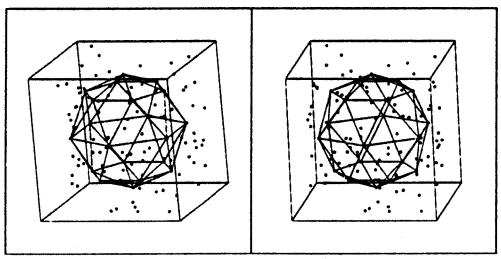

Figure 13. Input data

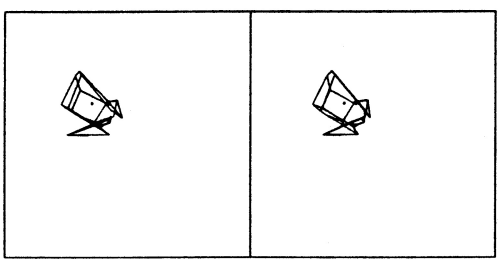

Figure 14. Voronoi region including self-intersection

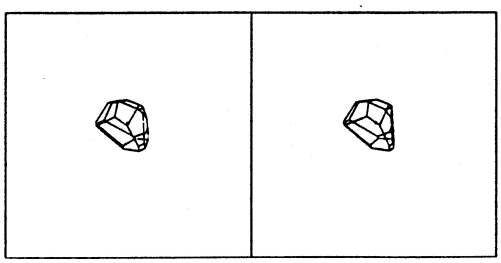

Figure 15. Voronoi region not including self-intersection

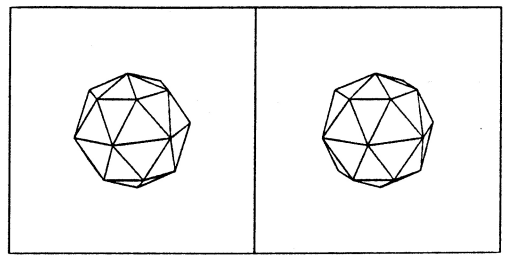

Figure 16. Remaining polyhedron

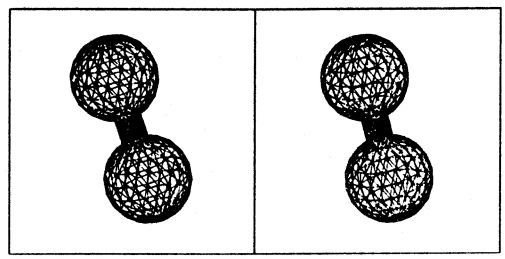

Figure 17. Polyhedron used as the constraint

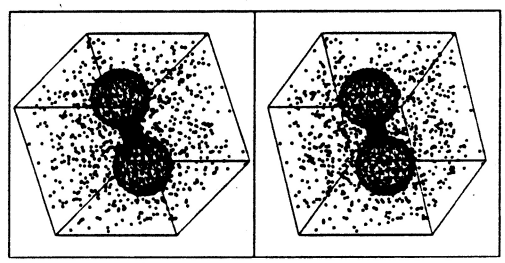

Figure 18. Input data

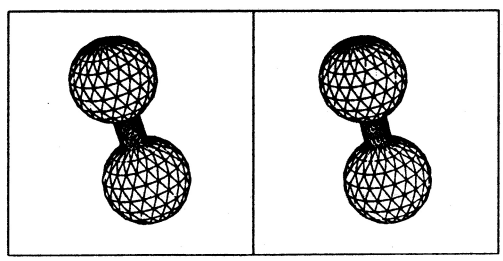

Figure 19. Remaining polyhedron