Robust Geometric Computation Based on Topological Consistency

Kokichi Sugihara

Department of Mathematical Information Science and Technology,
University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-0033, Japan
sugihara@simplex.t.u-tokyo.ac.jp

Abstract. The paper surveys a method, called the "topology-oriented method", for designing numerically robust geometric algorithms. In this method, higher priority is placed on the consistency of the topological structures of geometric objects than on numerical values. The resulting software is completely robust in the sense that inconsistency never arizes no matter how large numerical errors take place. The basic idea of this method and typical examples are shown.

1 Introduction

Quite a large number of "efficient" algorithms have been proposed to solve geometric problems. However, those algorithms are fragile in general in the sense that, if we implement them naively, they easily fail due to numerical errors [31,7,5,15,16]. Theoreticians design algorithms on the assumption that there is no numerical error or degeneracy, but in real computation both numerical errors and degeneracy arise frequently. This gap between the ideal world and the real world causes a serious problem of instability in actual geometric computation.

To overcome this difficulty, many approaches have been proposed. To simplify the situation, we can classify these approaches into three groups according to how much they rely on numerical computation. These three groups are shown in Fig. 1. The horizontal axis in this figure represents the amount of reliability of numerical values assumed in the design of robust algorithms; numerical values are more reliable in the right than in the left.

The first group is the "exact-computation approach", in which numerical computation are carried out in sufficiently high precision [41,29,21,23,24,35,30,1] [10,40,45]. The topological structure of a geometric object can be decided by the signs of the results of numerical computations. If we restrict the precision of the input data, these signs can be judged correctly in a sufficiently high but still finite precision. Using this principle, the topological structures are judged correctly as if the computation is done exactly. In this approach, we need not worry about misjudgement and hence theoretical algorithms can be implemented rather straightforward.

In this approach, degenerate situations are recognized exactly, and hence exceptional branches of processing for degenerate cases are necessary to complete

V.N. Alexandrov et al. (Eds.): ICCS 2001, LNCS 2073, pp. 12–26, 2001.

small	**reliance values numerical values**	**large**
⟵		⟶
topology-oriented approach	**tolerance-approach** ε**-geometry interval-arithmetic**	**exact-computation approach**

Fig. 1. Three groups of approaches to robustness.

the algorithms. However, such exceptional branches can be avoided by a symbolic perturbation scheme [6,35,44].

Another disadvantages of this approach is the computation cost. The computation in this approach is expensive, because multiple precisions are used. To decrease the cost, acceleration schemes are also considered. A typical method is a lazy evaluation scheme, in which computation is first done in floating-point arithmetic, and if the precision turns out to be insufficient, then they switch to multiple precision [1,4,10,32,40]. Another method is the use of modular arithmetic instead of the multiple precision [2,3,18].

The second group of approaches relies on numerical computation moderately. They start with the assumption that numerical computation contains errors but the amount of the errors is bounded. Every time numerical computation is done, the upper bound of the error is also evaluated. On the basis of this error bound, the result of computation is judged to be either reliable or unreliable, and the only reliable result is used [25,8,9,12,16,31,34]. This approach might be natural for programmers in order to cope with numerical errors, but it makes program codes unnecessarily complicated because every numerical computation should be followed by two alternative branches of processing, one for the reliable case and the other for the unreliable case. Moreover, this approach decreases the portability of the software products, because the amount of errors depends on computation environment.

The third group of approaches is the "topology-oriented approach", which does not rely on numerical computation at all. In this approach, we start with the assumption that every numerical computation contains errors and that the amount of the error cannot be bounded. We place the highest priority on the consistency of topological properties, and use numerical results only when they are consistent with the topological properties, thus avoiding inconsistency [42,43,20] [28,17,22,26,27,38,39].

In this paper, we concentrate on the third approach, i.e., the topology-oriented approach, and survey the basic idea and several examples.

2 Instability in Geometric Computation

First, we will see by an example how unstable the geometric computation is. Suppose that we are given a convex polyhedron Π and a plane H, and that

we want to cut the polyhedron by the plane and to take one part off. For this purpose we need to classify the vertices of Π into two groups, the vertices above H and those below H. Consider the situation where H is very close to and almost parallel to one face of Π, as shown in Fig. 2(a). Then, the classification of the vertices are easily violated by numerical errors. Hence, it can happen that, as shown in Fig. 2(b), a pair of mutually opposite vertices on the face are judged below H and the other vertices on the face are judged above H. This situation is inconsistent, because this situation implies that the face should meet H at two lines, while in Euclidean geometry two distinct planes can meet at most at one line. Such an inconsistent classification of the vertices usually causes the algorithm to fail.

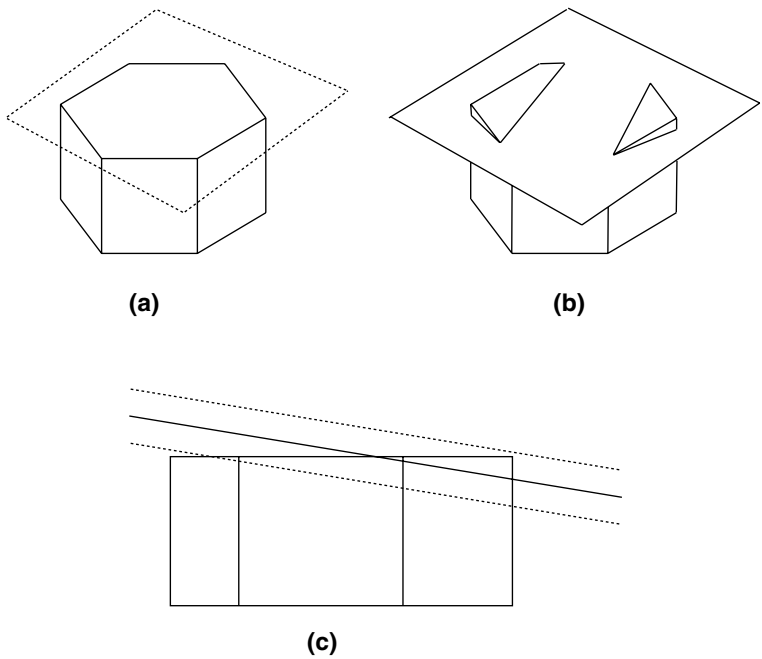


Fig. 2. Inconsistency caused by numerical errors in cutting a polyhedron by a plane.

A conventional method to circumvent this difficulty is to fix a certain small number ε, call a *tolerance*, and to consider two geometric elements to be at the same position if their distance is smaller than ε. Indeed, inconsistency can be avoided in many cases by this method. However, this is not complete; inconsistency still can happen.

Fig. 2(c) shows an example where the above method does not work. This is the picture of the scene in Fig. 1(a) seen in the direction parallel both to the top face of Π and the cut plane H. The pair of broken lines shows the region in

which the distance to H is smaller than ε. In this particular example, five of the vertices on the top face are judged to be exactly on H whereas the other vertex is judged below H. This is a contradiction, because in Euclidean geometry three or more noncollinear points being on H implies that the other vertices are also on H; hence all the vertices should be on the cut plane H.

3 Robustness and Consistency

Let P be a geometric problem, and let f be a theoretical algorithm to solve P. By a "theoretical" algorithm, we mean an algorithm that is designed assuming precise arithmetic, namely, one whose correctness is based on the assumption that no numerical error takes place in the computation.

The algorithm f can be considered a mapping from the set $\Xi(P)$ of all possible inputs to the set $\Omega(P)$ of all possible outputs. Each input $X \in \Xi(P)$ represents an instance of the problem P, and the corresponding output $f(X) \in \Omega(P)$ is a solution of the problem instance.

Both the input and the output can be divided into the "combinatorial and/or topological part" ("topological part" for short) and the "metric part." We represent the topological part by a subscript T and the metric part by a subscript M. More specifically, the input X is divided into the topological part X_{T} and the metric part X_{M}, and the output $f(X)$ is divided into the topological part $f_{\mathrm{T}}(X)$ and the metric part $f_{\mathrm{M}}(X)$.

For example, suppose that P is the problem of cutting a convex polyhedron by aplane. Then the topological part X_{T} of the input consists of the incidence relations among the vertices, the edges and the faces of the given polyhedron, and the metric part X_{M} consists of the equation of the cutting plane and the list of the three-dimensional coordinates of the vertices and/or the list of equations of the planes containing the faces. The topological part $f_{\mathrm{T}}(X)$ of the output consists of the incidence relations among the vertices, the edges and the faces of the computed polyhedron, and the metric part $f_{\mathrm{M}}(X)$ of the output consists of the list of the three-dimensional coordinates of the vertices of the computed polyhedron.

For another example, suppose that P is the problem of constructing the Voronoi diagram for a finite number of given points in the plane. Then the topological part X_{T} of the input consists of a single integer to represent the number N of points, and the metric part X_{M} is the set of the n pairs of coordinates of the points: $X_{\mathrm{T}} = \{n\}$ and $X_{\mathrm{M}} = \{x_1, y_1, \ldots, x_n, y_n\}$. The topological part $f_{\mathrm{T}}(X)$ of the output is the planar graph structure consisting of the Voronoi vertices and the Voronoi edges, and the metric part $f_{\mathrm{M}}(X)$ consists of the coordinates of the Voronoi vertices and the directions of the infinite Voronoi edges.

Let $\widetilde{f}$ denote an actually implemented computer program to solve P. The program $\widetilde{f}$ may be a simple translation of the algorithm f into a programming language, or it may be something more sophisticated aiming at robustness. The program $\tilde{f}$ can also be considered a mapping from the input set to the output

set. However, in actual situations, the program runs in finite-precision arithmetic, and consequently the behavior of $\widetilde{f}$ is usually different from that of f.

The program $\widetilde{f}$ is said to be *numerically robust* (or *robust* for short) if $\widetilde{f}(X)$ is defined for any input X in $\Xi(P)$. In other words, $\widetilde{f}$ is robust if it defines a total (not partial) function from $\Xi(P)$ to a superset $\widetilde{\Omega}(P)$ of $\Omega(P)$, i.e., if the program always carries out the task, ending up with some output, never entering into an endless loop nor terminating abnormally.

The program $\widetilde{f}$ is said to be *topologically consistent* (or *consistent* for short) if $\widetilde{f}$ is robust and $\widetilde{f}_{\mathrm{T}}(X) \in \Omega_{\mathrm{T}}(P)$ for any $X \in \Xi(P)$. In other words, $\widetilde{f}$ is consistent if the topological part $\widetilde{f}_{\mathrm{T}}(X)$ of the output coincides with the topological part $f_{\mathrm{T}}(X')$ of the correct solution of some instance X' (not necessarily equal to X) of the problem P.

Our goal is to construct $\widetilde{f}$ that is at least robust and hopefully consistent.

4 Basic Idra of the Topology-Oriented Approach

4.1 Basic Idea

In this section we suppose that exact arithmetic is not available and hence numerical computation contains errors. Fig. 3(a) shows how a conventional algorithm fails. Let $S = \{J_1, J_2, \ldots, J_n\}$ be the set of all the predicates that should be checked in the algorithm. Whether those predicates are true or not is judged on the basis of numerical computations. Since numerical computations contain errors, some of the predicates may be judged incorrectly, which in tern generate inconsistency and the algorithm fails.

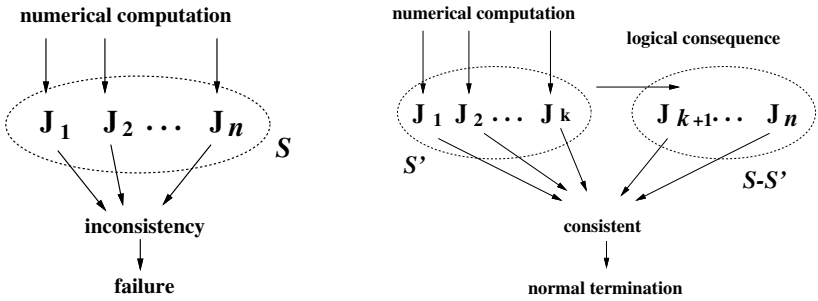


Fig. 3. Basic idea of the topology-oriented approach.

Numerical errors are inevitable in computation, but still we want to avoid inconsistency. To this goal, we first try to find a maximal subset, say S', of predicates that are independent from each other, as shown in Fig. 3(b), where "independent" means that the truth values of any predicates in S' do not affect

the truth values of the other predicates in this subset. The other predicates are dependent in the sense that their truth values are determined as the logical consequence of the truth values of the predicates in S'.

Once we find such a subset S', we evaluate the predicates in S' by numerical computation, and adopt the logical consequences of them as the truth values of the other predicates, i.e., the predicates in $S - S'$. Since the predicates in S' are independent, any assignment of the truth values to the predicates in S' does not generate inconsistency. Moreover, since we adopt the logical consequences of these truth values as the truth values of the predicates in $S - S'$, we never come across inconsistency.

We cannot guarantee the correctness of the truth values in S' because we have numerical errors, but once we believe the results of numerical computations, we can construct a consistent world. This is the basic idea for avoiding inconsistency. In the following subsections we will show how this idea works using typical example problems.

5 Examples

5.1 Cutting a Convex Polyhedron by a Plane

Let Π be a convex polyhedron in a three-dimensional space, and H be a plane. We consider the problem of cutting Π by H and taking one part off. Theoretically this problem is not difficult. What we have to do is to classify the vertices of Π into those above H and those below H. Once we classify them, we can determine the topological structure of the resulting polyhedron.

However, a naive implementation of this algorithm is not robust, as we have already seen in section 2.

To attain numerical robustness, we concentrate on the topological part of the algorithm. From the topological point of view, the vertices and the edges of Π form a planar graph, say G. As shown in Fig. 4, to cut Π by H, we first find the vertices that are above H (the vertices with black circles in Fig. 4(b)), next generate new vertices on the edges connecting the vertices above H and those below H (the vertices with white circles in Fig. 4(b)), generate a new cycle connecting them (the broken lines in Fig. 4(b)), and finally remove the substructure inside the cycle (Fig. 4(c)).

Let V_1 be the set of vertices of G that are judged above H, and let V_2 be the set of vertices that are judged below H. Since Π is convex, the next property holds.

Proposition 1. The subgraph of G induced by V_1 and that induced by V_2 are both connected.

We place higher priority on this property; we employ numerical results only when they do not contradict against this property. In this way we can construct a numerically robust algorithm [38].

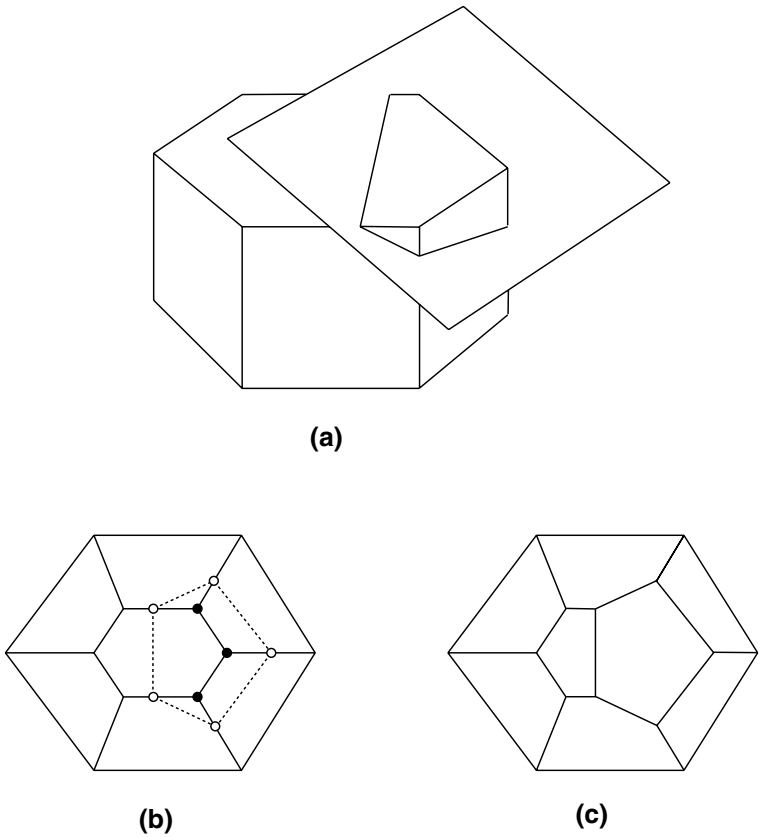
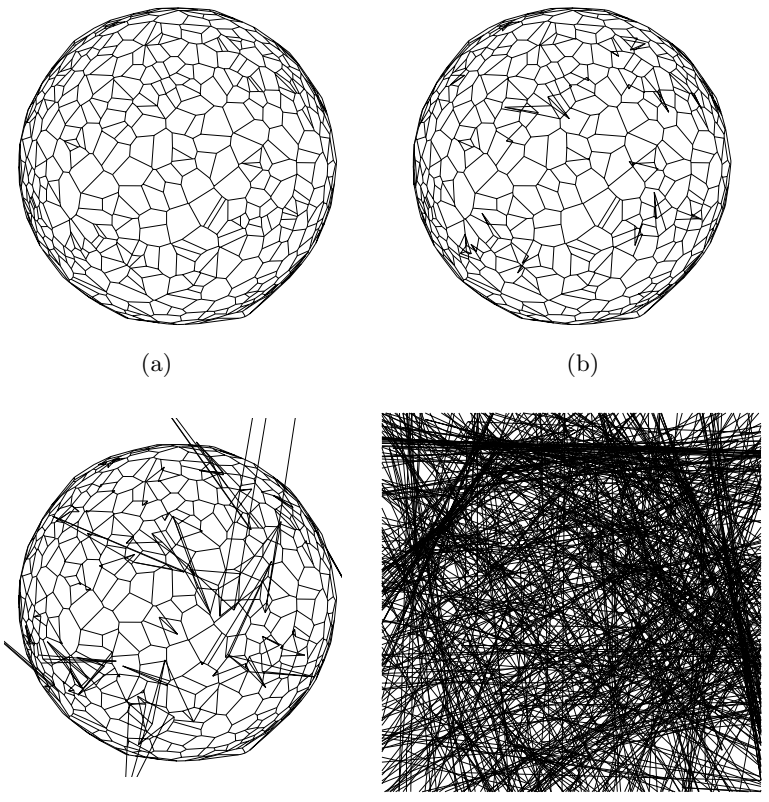


Fig. 4. Topological aspect of the cut operation.

An example of the behavior of this algorithm is shown in Fig. 5. Fig. 5(a) is the output of the algorithm when a cube is cut by 500 planes that are tangent to a common sphere. This problem is not difficult; a naively implemented software may also be able to give the same output. However, our algorithm is designed so that it never fails even if numerical computation contains large errors. To see this property, artificial errors were added to all the floating-point computations in the algorithm using random numbers. Then, the output becomes as shown in Fig. 5(b). Some part of the output is not correct. However, what is important is that although the algorithm made misjudgements, it carries out the task, ending up with some output. When we added larger artificial errors, the output becomes as shown in Fig. 5(c). As the extremal case, when we replaced all the floating-point computations by random numbers, then the output was as shown in Fig. 5. This output is of course nonsense, but an important thing is that topological inconsistency never arises in this algorithm and always some output is given. If we see Fig. 5(d), (c), (b), (a) in this order, we can say that the output of

the algorithm converges to the correct answer as the precision in computation becomes higher.



(a) (b)

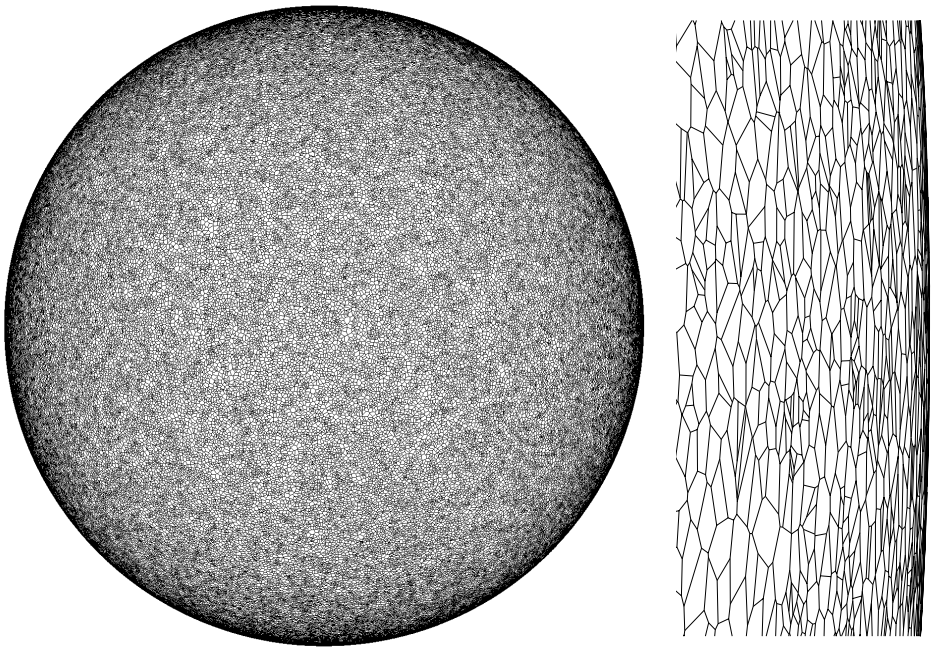
(c) (d)

Fig. 5. Behavior of the topology-oriented algorithm for cutting a polyhedron by a plane.

Another example of the output of this algorithm is shown in Fig. 6. Fig. 6(a) is the result of cutting a cube by 10^5 planes touching a common sphere, and Fig. 6(b) is a magnified picture of the left portion. This example also shows the robustness of the algorithm.

5.2 Construction of Voronoi Diagrams

Let $S = \{\mathrm{P}_1, \mathrm{P}_2, \ldots, \mathrm{P}_n\}$ be a set of finite number of points in the plane. The region $R(S; \mathrm{P}_j)$ defined by $R(S; \mathrm{P}_i) = \{\mathrm{P} \in \mathbf{R}^2 \mid d(\mathrm{P}, \mathrm{P}_i) < d(\mathrm{P}, \mathrm{P}_j),\ j =$



(a) (b)

Fig. 6. Cutting a cube by 10^5 planes.

$1, \ldots, i-1, i+1, \ldots, n\}$ is called the *Voronoi region* of P_i, where $d(\mathrm{P}, \mathrm{Q})$ represents the Euclidean distance between the two points P and Q. The partition of the plane into Voronoi region $R(S; \mathrm{P}_i)$, $i = 1, 2, \ldots, n$, and their boundaries is called the *Voronoi diagram* for S.

In the incremental algorithm, we start with the Voronoi diagram for a few points, and modify it by adding the other points one by one. An increment step proceeds in the following way. Suppose that we have already constructed the Voronoi diagram for k points, and now want to add the $(k+1)$-th point. To modify the Voronoi diagram, we first find a cyclic list formed by the perpendicular bisectors between the new point and the neighboring old points, and next remove the substructure inside this cycle.

Though this procedure is theoretically simple, it is numerically unstable because the sequence of bisectors does not necessarily form a cycle in imprecise arithmetic, particularly when the input points are degenerated [42,43]. To construct a robust algorithm, we can use the following property.

Proposition 2. If a new point is inside the convex hull of the old points, the substructure to be removed is a tree in a graph theoretical sense.

We place higher priority on this property than on numerical values, and thus can construct a numerically robust algorithm for the Voronoi diagram [42,43]. Fig. 7(a) is an example of the output of this algorithm. Though the points were highly degenerate, the algorithm constructed the globally correct Voronoi diagram. If we magnify the central portion of this figure by 10^4, 10^5 and 10^6 respectively, we can see small disturbance, as shown in Fig. 7(b), (c) and (d). However, it should be noted that such disturbance never makes the algorithm to clash, because the algorithm always maintains topological consistency of the data structure.

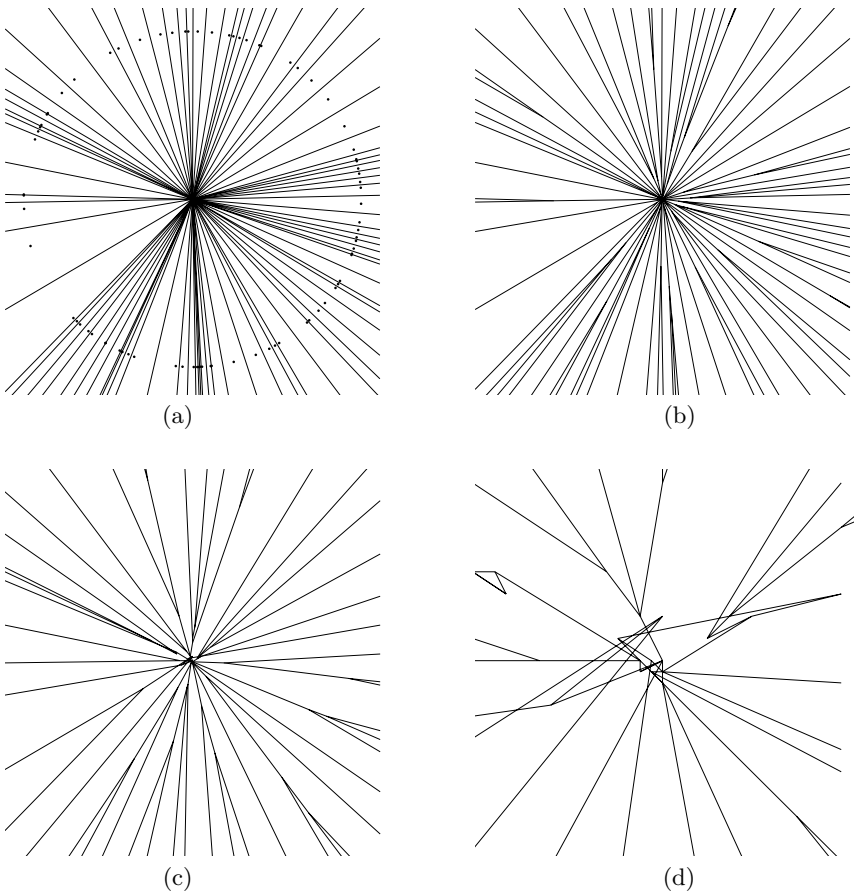


Fig. 7. Voronoi diagram for highly degenerate set of points

Other applications of the topology-oriented method include the divide-and-conquer construction of the two-dimensional Voronoi and Delaunay diagrams [28], the incremental construction of the three-dimensional Voronoi and Delau-

nay diagrams [20,19], the incremental construction of the Voronoi diagram for polygons [17], the gift-wrapping construction of the three-dimensional convex hull [39], the divide-and-conquer construction of the three-dimensional convex hull [26,27], the intersection of half spaces in the three-dimensional space [38], and other applications [36,37].

6 Discussions

Here we consider some general properties of the topology-oriented algorithms.

Robustness

A topology-oriented algorithm is completely robust in the sense that it does not require any minimum precision in numerical computation. All possible behavior is specified by the topological skeleton, and therefore even if numerical precision is very poor (or even if all the results of numerical computation are replaced by random numbers), the algorithm still carries out the task and generates some output.

Topological Consistency

Whether the algorithm is topologically consistent depends on the chosen set Q of purely topological properties. The topology-oriented implementation guarantees that the output satisfies all the properties in Q. In general, however, Q gives only a necessary condition for the output to belong to the set $\Omega(P)$ of all the possible solutions of the problem P; it does not necessarily give a sufficient condition. This is because the purely topological characterization of the solution set is not known for many geometric problems, and even if it is known, it is usually time-consuming to check the conditions (note that Q should contain only those properties that can be checked efficiently).

Hence, topological consistency can be attained for a limited number of problems. A trivial example is the problem of constructing a convex hull in the plane. For this problem, any cyclic sequence of three or more vertices chosen from the input points can be the solution of a perturbed version of the input, so that topological consistency can be easily attained.

More nontrivial examples arise in the class of problems related to convex polyhedra. The topological structures of convex polyhedra can be characterized by Steinitz's theorem, which says that graph G is a vertex-edge graph of a convex polyhedron if and only if G is a 3-connected planar graph with four or more vertices [33]. Because of this theorem we can see that the algorithm in Section 5.1 is topologically consistent. Actually we can prove that if the input graph G is a 3-connected planar graph, then the output G' is also a 3-connected planar graph. Hence, the output of this algorithm is the vertex-edge graph of some polyhedron, that is, the output is the vertex-edge graph of the solution of some instance of the problem though it is not necessarily the given instance.

For the two-dimensional Voronoi diagram for points, necessary and sufficient conditions are known [13,14]. However, these conditions require much time to

check, and hence cannot be included in Q. Actually the algorithm in Section 5.2 uses only a necessary condition, and hence it is not topologically consistent.

Convergence

If the input to the algorithm is not degenerate, the output converges to the correct solution as the computation becomes more and more precise, because the correct branch of the processing is chosen with sufficiently high precision. However, the speed of convergence cannot be stated in a unifying manner, because it depends on the individual problems and on the implementation of numerical computation.

The situation is different for degenerate input. If the algorithm is topologically consistent, the output converges to an infinitesimally perturbed version of the correct solution. In any high precision, the true degenerate output cannot be obtained, because degenerate cases are not taken into account in the topology-oriented approach. For example, suppose that the cutting plane H goes through a vertex of the polyhedron Π. Then our algorithm classifies the vertex either above H or below H, and decides the topological structure accordingly. As a result, the output may contain the edges whose lengths are almost 0.

7 Concluding Remarks and Open Problems

We have seen the topology-oriented approach to the robust implementation of geometric algorithms, and also discussed related issues. Since we can separate the topological-inconsistency issue from the error-analysis issue completely, the algorithm designed in this approach has the following advantages:

(1) No matter how large the numerical errors are that may take place, the algorithm never fails; it always carries out the task and gives some output.

(2) The output is guaranteed to satisfy the topological properties Q used in the topological skeleton of the algorithm.

(3) For a nondegenerate input, the output converges to the correct solution as the precision in computation becomes higher.

(4) The structure of the algorithm is simple because exceptional branches for a degenerate input are not necessary.

However, in order to use the output for practical applications we still have many problems to be solved.

The topology-oriented approach might give output that contains numerical disturbance particularly when the input is close to degeneracy. Such disturbances are usually very small but not acceptable for some applications. Hence, to rewrite the application algorithms in such a way that they can use numerically disturbed output of the topology-oriented algorithms is one of main future problems related to this approach.

This work is supported by the Grant-in-Aid for Scientific Research of the Japan Ministry of Education, Science, Sports, and Culture, and the Torey Science Foundation.

References

1. M. Benouamer, D. Michelucci and B. Peroche: Error-free boundary evaluation using lazy rational arithmetic—A detailed implementation. *Proceedings of the 2nd Symposium on Solid Modeling and Applications*, Montreal, 1993, pp. 115–126.
2. H. Brönnimann, I. Z. Emiris, V. Y. Pan and S. Pion: Computing exact geometric predicates using modular arithmetic with single precision. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, Nice, June 1997, pp. 1-182.
3. H. Brönnimann and M. Yvinec: Efficient exact evaluation of signs of determinants. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, Nice, June 1997, pp. 166-173.
4. K. L. Clarkson: Safe and effective determinant evaluation. *Proceedings of the 33rd IEEE Symposium on Foundation of Computer Science*, pp. 387-395.
5. D. Dobkin and D. Silver: Recipes for geometric and numerical analysis—Part I, An empirical study. *Proceedings of the 4th ACM Annual Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 93–105.
6. H. Edelsbrunner and E. P. Mücke: Simulation of simplicity—A technique to cope with degenerate cases in geometric algorithms. *Proceedings of the 4th ACM Annual Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 118–133.
7. D. A. Field: Mathematical problems in solid modeling—A brief survey. G. E. Farin (ed.), *Geometric Modeling—Algorithms and New Trends*, SIAM, Philadelphia, 1987, pp. 91–107.
8. S. Fortune: Stable maintenance of point-set triangulations in two dimensions. *Proceedings of the 30th IEEE Annual Symposium on Foundations of Computer Science*, Research Triangle Park, California, 1989, pp.94–499.
9. S. Fortune: Numerical stability of algorithms for 2D Delaunay triangulations. *International Journal of Computational Geometry and Applications*, vol. 5 (1995), pp. 193-213.
10. S. Fortune and C. von Wyk: Efficient exact arithmetic for computational geometry. *Proceedings of the 9th ACM Annual Symposium on Computational Geometry*, San Diego, 1993, pp. 163–172.
11. D. H. Greene and F. Yao: Finite resolution computational geometry. *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, Toronto, October 1986, pp. 3-152.
12. L. Guibas, D. Salesin and J. Stolfi: Epsilon geometry—Building robust algorithms from imprecise computations. *Proc. 5th ACM Annual Symposium on Computational Geometry* (Saarbrücken, May 1989), pp. 208–217.
13. T. Hiroshima, Y. Miyamoto and K. Sugihara: Another proof of polynomial-time recognizability of Delaunay graphs. *IEICE Transactions on Fundamentals*, Vol. E83-A (2000), pp. 627-638.
14. C. D. Hodgson, I. Rivin and W. D. Smith: A characterization of convex hyperbolic polyhedra and of convex polyhedra inscribed in the sphere. *Bulletin of the American Mathematical Society*, vol. 27 (1992), pp. 6-251.
15. C. M. Hoffmann: The problems of accuracy and robustness in geometric computation. *IEEE Computer*, vol. 22, no. 3 (March 1989), pp. 31-1.

16. C. M. Hoffmann: *Geometric and Solid Modeling*. Morgan Kaufmann Publisher, San Mateo, 1989.
17. T. Imai: A topology-oriented algorithm for the Voronoi diagram of polygon. *Proceedings of the 8th Canadian Conference on Computational Geometry*, 1996, pp. 107–112.
18. T. Imai: How to get the sign of integers from their residuals. *Abstracts of the 9th Franco-Japan Days on Combinatorics and Optimization*, 1996, p. 7.
19. H. Inagaki and K. Sugihara: Numerically robust algorithm for constructing constrained Delaunay triangulation. *Proceedings of the 6th Canadian Conference on Computational Geometry*, Saskatoon, August 19, pp. 171-176.
20. H. Inagaki, K. Sugihara and N. Sugie, N.: Numerically robust incremental algorithm for constructing three-dimensional Voronoi diagrams. *Proceedings of the 6th Canadian Conference Computational Geometry*, Newfoundland, August 1992, pp. 3–339.
21. M. Karasick, D. Lieber and L. R. Nackman: Efficient Delaunay triangulation using rational arithmetic. *ACM Transactions on Graphics*, vol. 10 (1991), pp. 71–91.
22. D. E. Knuth: *Axioms and Hulls*. Lecture Notes in Computer Science, no. 606, Springer-Verlag, Berlin, 1992.
23. G. Liotta, F. P. Preparata and R. Tamassia: Robust proximity queries — An illustration of degree-driven algorithm design. *Proceedings of the 13th Annual ACM Symposium on Computational Geometry*, 1997, pp. 156-165.
24. K. Mehlhorn and S. Näher: A platform for combinatorial and geometric computing. *Communications of the ACM*, January 1995, pp. 96-102.
25. V. Milenkovic: Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artificial Intelligence*, vol. 37 (1988), pp. 377-01.
26. T. Minakawa and K. Sugihara: Topology oriented vs. exact arithmetic—experience in implementing the three-dimensional convex hull algorithm. H. W. Leong, H. Imai and S. Jain (eds.): *Algorithms and Computation, 8th International Symposium, ISAAC'97* (Lecture Notes in Computer Science 1350), (December, 1997, Singapore), pp. 273–282.
27. T. Minakawa and K. Sugihara: Topology-oriented construction of three-dimensional convex hulls. *Optimization Methods and Software*, vol. 10 (1998), pp. 357–371.
28. Y. Oishi and K. Sugihara: Topology-oriented divide-and-conquer algorithm for Voronoi diagrams. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 57 (1995), pp. 303–3.
29. T. Ottmann, G. Thiemt and C. Ullrich: Numerical stability of geometric algorithms. *Proceedings of the 3rd ACM Annual Symposium on Computational Geometry*, Waterloo, 1987, pp. 119–125.
30. P. Schorn: Robust algorithms in a program library for geometric computation. Dissertation submitted to the Swiss Federal Institute of Technology (ETH) Zürich for the degree of Doctor of Technical Sciences, 1991.
31. M. Segal and C. H. Sequin: Consistent calculations for solid modeling. *Proceedings of the ACM Annual Symposium on Computational Geometry*, Baltimore, 1985, pp. 29–38.
32. J. R. Shewchuk: Robust adaptive floating-point geometric predicates. *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, Philadelphia, May 1996, pp. 1-150.
33. E. Steinitz: *Polyheder und Raumeinteilungen*. Encyklopädie der mathematischen Wissenchaften, Band III, Teil 1, 2. Hälfte, IIIAB12, pp. 1-139.

34. A. J. Steward: Local robustness and its application to polyhedral intersection. *International Journal of Computational Geometry and Applications*, vol. (1994), pp. 87-118.
35. K. Sugihara: A simple method for avoiding numerical errors and degeneracy in Voronoi diagram construction. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E75-A (1992), pp.68–477.
36. K. Sugihara: An intersection algorithm based on Delaunay triangulation. *IEEE Computer Graphics and Applications*, vol. 12, no. 2 (March 1992), pp. 59-67.
37. K. Sugihara: Approximation of generalized Voronoi diagrams by ordinary Voronoi diagrams. *Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing*, vol. 55 (1993), pp. 522-531.
38. K. Sugihara: A robust and consistent algorithm for intersecting convex polyhedra. *Computer Graphics Forum*, EUROGRAPHICS'94, Oslo, 1994, pp. C-45–C-54.
39. K. Sugihara: Robust gift wrapping for the three-dimensional convex hull. *Journal of Computer and System Sciences*, vol.9 (1994), pp. 391–407.
40. K. Sugihara: Experimental study on acceleration of an exact-arithmetic geometric algorithm. *Proceedings of the 1997 International Conference on Shape Modeling and Applications*, Aizu-Wakamatsu, 1997, pp. 160–168.
41. K. Sugihara and M. Iri: A solid modelling system free from topological inconsistency. *Journal of Information Processing*, vol. 12 (1989), pp. 380–393.
42. K. Sugihara and M. Iri: Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic. *Proceedings of the IEEE*, vol. 80 (1992), pp. 71–1484.
43. K. Sugihara and M. Iri: A robust topology-oriented incremental algorithm for Voronoi diagrams. *International Journal of Computational Geometry and Applications*, vol. (1994), pp. 179–228.
44. C. K. Yap: A geometric consistency theorem for a symbolic perturbation scheme. *Proceedings of the 4th Annual ACM Symposium on Computational Geometry*, Urbana-Champaign, 1988, pp. 1–142.
45. C. K. Yap: The exact computation paradigm. D.-Z. Du and F. Hwang (eds.): *Computing in Euclidean Geometry, 2nd edition*. World Scientific, Singapore, 1995, pp.52–492.
46. X. Zhu, S. Fang and B. D. Brüderlin: Obtaining robust Boolean set operations for manifold solids by avoiding and eliminating redundancy. *Proceedings of the 2nd Symposium on Solid Modeling and Applications*, Montreal, May 1993, pp. 7-154.