Appl. Math. Lett. Vol. 2, No. 2, pp. 203–206, 1989
Printed in Great Britain.

0893-9659/89 $3.00 + 0.00

Two Design Principles of Geometric Algorithms in Finite-Precision Arithmetic

KOKICHI SUGIHARA and MASAO IRI

Department of Mathematical Engineering and Information Physics
Faculty of Engineering, University of Tokyo

Abstract The paper presents two new approaches to the design of geometric algorithms which are expected to work correctly in finite-precision arithmetic.

INTRODUCTION

Geometric algorithms have intensively been studied in computational geometry, and a number of "efficient" algorithms have been proposed for various types of geometric problems. However, in designing those algorithms, emphasis has been placed only on the time-complexity point of view, other aspects, numerical problems in particular, being considered not so seriously. Geometric algorithms are usually designed in the world where there is no numerical error, so that the correctness of the algorithms is proved only in that world. In real computers, on the other hand, every real number is represented in finite precision, so that numerical errors occasionally arise in representation and computation of geometric data unless special care is taken. Hence, straightforward translation of a geometric algorithm into a programming language will not necessarily result in a practical computer program. Numerical errors often cause such a program to fail; for example, to end abnormally, to fall in an endless loop, or to generate a wrong output. In this sense there is a great gap between "theoretically correct" algorithms and "practically valid" computer programs.

Recently, importance of numerical facets of geometric problems have been recognized, and several approaches were proposed for getting around the numerical problems. They include approaches in which geometric data are represented in terms of integer grid points [1] or of geometric intervals [2]. It seems, however, that those approaches stem from the resignation that numerical errors are inevitable in finite-precision arithmetic.

In this letter we present two new approaches to the problem of numerical unstableness. The first approach is to construct a closed world in which topological structures of geometric objects are always determined precisely even in finite-precision computation; this approach is applied to solid modelling. The second approach is to avoid topological inconsistency by placing higher priority to logical consequence than to numerical judgement; this approach is applied to the construction of Voronoi diagrams.

CONSTRUCTION OF AN ERROR-FREE WORLD

Numerical data given in a fixed number of digits can represent only a finite number of different objects, so that only a finite number of different values can appear when a fixed set of algebraic operations are applied to those data. The distinction among these values, therefore, can be judged exactly in certain finite-precision computation. This principle has recently been utilized for constructing efficient algorithms in several fields

This research is supported in part by the Grant in Aid for Scientific Research of the Ministry of Education, Science and Culture of Japan (Grant No. 62580017).

AML 2:2-G

of mathematics such as linear programming [3][4][5] and symbolic computation [6]. The same principle can be utilized for constructing a closed world in which all the topological relations among a given set of geometric objects can be exactly decided in finite-precision computation. As an example of such a closed world, we will propose a solid modelling system in which no topological error arises in the course of generating polyhedral solid models through set-theoretic combination of primitive solids [7].

The system can preserve topological consistency because of the following conventions.

First, basic operations are restricted to the form

$$A \leftarrow A * M(P),$$

where A denotes a solid to be generated, P a primitive solid, M a motion, and $*$ a set-theoretic operation such as union and intersection. At each step, we choose a primitive P, move it to an appropriate position and posture, and combine it with the current solid A; starting with an empty set A, we continue this process until we get the desired solid.

Secondly, all the fundamental metric data of a solid are represented by the coefficients of face equations

$$a_i x + b_i y + c_i z + d_i = 0,$$

with respect to the coordinate system to which the solid A is fixed, in such a way that the coefficients are integers satisfying $-L \leq a_i, b_i, c_i \leq L$ and $-L^2 \leq d_i \leq L^2$ for a certain large number L. A primitive solid is assumed to be a polyhedron each of whose vertices is incident to exactly three faces, so that even if numerical errors take place in the course of the coordinate transformation for the motion, the relevant vertices are consistently defined as the intersections of three of the transformed faces. The coefficients of the equations of the transformed faces are approximated by integers, which are considered as the most fundamental metric data.

Thirdly, all the computation for deciding the topological structures of the generated solids are reduced to finding a relative configuration of four planes. It can be shown that in order for the modelling system to decide precisely such relative configurations in finite-precision computation, the necessary and sufficient precision in computation is *five* times the precision in which the fundamental metric data are represented.

A two-dimensional version of the system was implemented in a computer program, in which plane figures are generated through set-theoretic combination of simple polygons. All the fundamental metric data are represented by the coefficients of edge equations, and all the computations for deciding the topological structures are reduced to finding a relative configuration of three lines, so that exact topological structures in a figure are computed in *four* times the precision of the fundamental data. Among many examples which we have computed with this program, there is the union of 1000 triangles generated in such a way that vertices of distinct triangles came close to one another, for which a traditional-type program could not construct the union properly [7].

GIVING THE HIGHEST PRIORITY TO TOPOLOGICAL CONSISTENCY

Our second approach is to place higher priority to topological structures than to numerical results to avoid inconsistency. Misjudgement on topological structures due to numerical errors often generates inconsistency, but not always because one misjudgement may be consistent with another misjudgement. Therefore, even in a world where misjudgement due to numerical errors cannot be avoided, we can get around inconsistency if we consider topological structures more important than numerical results. This principle is exemplified by our new algorithm for constructing Voronoi diagrams for points given on a plane.

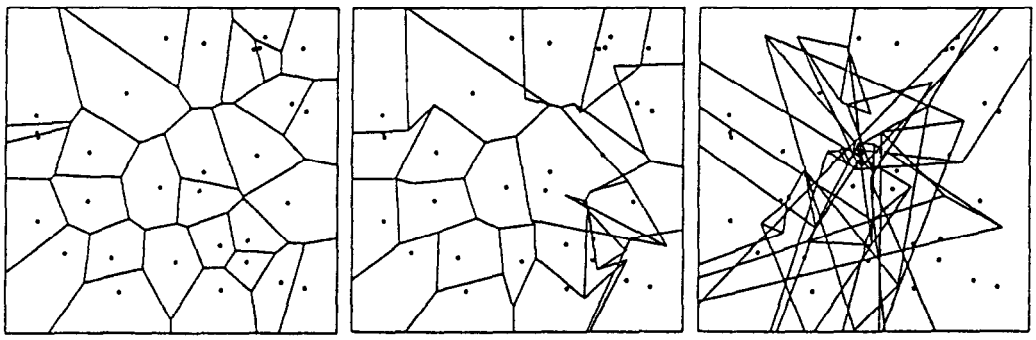

Fig. 1. Construction of the Voronoi diagram in finite precision for 30 generators located at random in the unit square: (a) Output of the program in which floating-point computation were carried out in single precision; (b) Output of the program in which small random numbers were added to the results of floating-point computations; (c) Output of the program in which all the floating-point computations were replaced by random numbers.

For two points p and q, $d(p,q)$ denotes the Euclidean distance between p and q. For a finite set $P = \{p_1, \ldots, p_n\}$ of points on a plane, the region defined by

$$V(p_i) = \{p \mid d(p,p_i) < d(p,p_j) \text{ for any } j(\neq i)\}$$

is called the *Voronoi region* of p_i, and the partition of the plane into Voronoi regions $V(p_1), \ldots, V(p_n)$ is called the *Voronoi diagram* for P. An element of P is called a *generator* of the Voronoi diagram, and an edge and a vertex of the diagram are called a *Voronoi edge* and a *Voronoi vertex*, respectively.

A Voronoi diagram possesses a number of properties, both purely topological and nontopological; for example, from the topological point of view

(P1) *a Voronoi diagram is a planar graph (of degree three in almost all cases) embedded in the plane, partitioning the plane into as many regions as the generators,* and
(P2) *two Voronoi regions do not share more than one boundary line segment,*

while, from the more quantitative point of view, the Voronoi regions are convex polygons and Voronoi edges do not intersect except at their terminal vertices. In finite-precision arithmetic, it is difficult to guarantee the nontopological properties completely, while the topological properties can usually be guaranteed by means of combinatorial computation unless it requires too much time in computation.

Among various algorithms proposed for constructing Voronoi diagrams [8][9][10], an incremental-type algorithm [11] is most practical because, on the average, it runs in $O(n)$ time for n generators. Here we re-design this algorithm so as to make it work well in finite-precision environment [12].

In our new algorithm we give the highest priority to the fact that the Voronoi diagram is a degree-three diagram with the properties (P1) and (P2); at each step of the addition of a new generator, a current diagram is changed into another without violating (P1) and (P2). In other words, the main task in the addition of the ith generator is "to convert a degree-three planar diagram with $i-1$ cells to a degree-three planar diagram with i cells in such a way that the ith cell does not touch another along two or more boundary lines." Of course, this specification only does not define a unique construction of the latter diagram, but we use the results of floating-point computation to choose among the possible diagrams satisfying (P1) and (P2) the one that appears to be the

closest to the true Voronoi diagram.

The algorithm designed in this way uses floating-point numbers only to clear up the ambiguity in choosing the topological structure of the diagram, and hence does not come up with inconsistency; however poor the precision of computation may be, the algorithm always carries out its task, ending up with some diagram that satisfies at least (P1) and (P2). Such an output can be considered as an approximation to the Voronoi diagram, and it converges to the true Voronoi diagram as the precision in computation becomes higher in the sense that, for any positive ε, there exists a precision such that the output given by computation in any higher precision is ε-retractable to the true Voronoi diagram.

Our algorithm was implemented in a FORTRAN program. Fig. 1 shows the outputs of this program for 30 generators located at random in a unit square. In all the cases (a), (b) and (c) the program carried out its task without coming across inconsistency and gave outputs that satisfied (P1) and (P2).

The present program is not only robust against numerical errors, but it possesses also other remarkable features. First, the average time complexity is linear in the number of generators, since our modification does not affect the time complexity of the incremental-type algorithm originally proposed in [11]. Second, the program is much simpler in structure than the conventional, because it need not take care of special treatment for degenerate cases. Recall that, in finite-precision arithmetic, we cannot discern whether degeneracy takes place or not; even if a certain numerical result suggests degeneracy, it means no more than that we are very close to a degenerate case. Hence, in a finite-precision world, all cases may be considered as nondegenerate.

REFERENCES

1. D. H. Greene and F. Yao, *Finite-resolution computational geometry,* Proceedings of the 27th ACM Annual Symposium on Foundations of Computer Science, Toronto, 1986, 143–152.
2. J. U. Turner, *Accurate solid modeling using polyhedral approximations,* Computer Graphics and Applications 8 (1988), 14–28.
3. L. G. Khachian, *Polynomial algorithms in linear programming,* USSR Computational Mathematics and Mathematical Physics 20 (1980), 53–72.
4. N. Karmarkar, *A new polynomial-time algorithm for linear programming,* Combinatorica 4 (1984), 373–395.
5. M. Iri and H. Imai, *A multiplicative barrier function method for linear programming,* Algorithmica 1 (1986), 455–482.
6. A. K. Lenstra, *Polynomial factorization by root approximation,* J. Fitch (ed.), "Proceedings of the International Symposium on Symbolic and Algebraic Computation", Lecture Notes in Computer Science, no. 174, Springer-Verlag, Berlin, 1984, 272–276.
7. K. Sugihara and M. Iri, *An approach to error-free solid modelling* (in Japanese), Transactions of Information Processing Society of Japan 28 (1987), 962–974. (An English version is also available: K. Sugihara and M. Iri, *A solid modelling system free from topological inconsistency* (preprint), Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1988).
8. M. I. Shamos and D. Hoey, *Closest-point problems,* Proceedings of the 16th IEEE Annual Symposium on Foundations of Computer Science, 1975, 151-162.
9. D. T. Lee and B. J. Schachter, *Two algorithms for constructing a Delaunay triangulation,* International Journal of Computer and Information Sciences 9 (1980), 219–242.
10. S. Fortune, *A sweepline algorithm for Voronoi diagrams,* Proceedings of the 2nd ACM Annual Symposium on Computational Geometry, Yorktown Heights, 1986, 313–322.
11. T. Ohya, M. Iri and K. Murota, *Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms,* Journal of the Operations Research Society of Japan 27 (1984), 306–336.
12. K. Sugihara and M. Iri, *Geometric algorithms in finite-precision arithmetic,* Abstracts of the 13th International Symposium on Mathematical Programming, Tokyo, 1988, WE/3K2, 196. (The text presented was printed as *Research Memorandum* RMI 88-10, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1988.)

Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 7-3-1 Hongo, Bunkyo-ku, Tokyo 113, Japan.