アルゴリズム 16－5
（1990．7．16）

C-oriented Polygonの交差探索問題について

譚 学厚，　平田 富夫，　稲垣 康善

名古屋大学　工学部

多辺形の集合がc-orientedと呼ばれるのは，含まれる辺の方向が定数種類しかないときをいう．本稿ではこのような多辺形に対し交差探索問題を調べる．交差探索問題とは，n個の多辺形の集合Sに対して，質問多辺形qが与えられる時，pと交わるSの要素を列挙する問題である．われわれのアルゴリズムは$O(n\ log\ n)$の記憶領域と前処理時間を要し，qと交わるSのt個の多辺形を$O(log\ n\ +\ t)$時間で報告する．さらに，このアルゴリズムをSが動的な場合にも拡張する．

The Intersection Searching Problem for C-oriented Polygons

Xue-Hou TAN,　Tomio HIRATA　and Yasuyoshi INAGAKI

Faculty of Engineering, Nagoya University, Chikusa-ku, Nagoya 464, Japan

A set of polygons is called c-oriented if the edges of all polygons have a constant number of previously defined orientations. The intersection searching problem is examined for such objects: Given a set S of c-oriented polygons and a c-oriented query polygon q, report all the polygons in S that intersect q. Our algorithm requires $O(n\ log\ n)$ space and preprocessing time and reports the t polygons of S that intersect q in $O(log\ n\ +\ t)$ time, where n is the cardinality of S. Furthermore, the algorithm is extended to the case where S is a dynamic set of polygons.

1 Introduction

Polygon intersection searching is one of the most fundamental problems in computational geometry. A considerable amount of effort has been devoted to this problem. Most of the results obtained concern two basic types of polygons, i.e. rectangles with edges parallel to the coordinate axes and polygons whose edges are of arbitrary orientations. In [9], Güting (1984) considered the problem on a set of "c-oriented polygons" and gave the efficient algorithms in a static as well as a dynamic setting. C-oriented polygons are those whose bounding edges have a constant number c of orientations. Since then, many improvements have been made on the complexities for both rectangles and arbitrarily oriented polygons, but nothing for c-oriented polygons. Here we reconsider the c-oriented version of the polygonal intersection searching problem, and devise new algorithms with better time and space bounds.

The notion of c-orientedness was introduced by Güting [8], which leads to a study on "c-oriented" geometry later [3, 8, 9, 10, 15, 16, 18, 19]. Motivations for the study are two-fold: One is to bridge the gap in complexity between orthogonal objects and arbitrary objects; the other is that there really exist some applications where objects are finitely oriented, for example, new chip designs use not only 0^0 and 90^0 lines but also 45^0 and 135^0 lines.

2 C-oriented Polygonal Intersection Searching

The general polygonal intersection searching problem is defined as follows: Given a set of simple polygons, each with a bounded number of edges, find the polygons whose intersection with a query polygon of the same type is nonempty.

In this paper we study the c-oriented version of the polygonal intersection searching problem. First, we define the notion of "orientation". An orientation is a vector with a positive y component which passes through the origin. We describe an orientation α as the (counterclockwise) angle from the x axis. See Fig. 1. A polygon is called *c-oriented* if each edge of the polygon is parallel to one of c possible orientations chosen from the collection $C = \{\alpha_1, \alpha_2, \cdots, \alpha_c\}$. We will treat the collection $\{\alpha_1, \alpha_2, \cdots, \alpha_c\}$ as fixed for the rest of this paper and restrict the input polygons (query polygons and queried polygons) of our algorithm to be finitely oriented with respect to this collection.

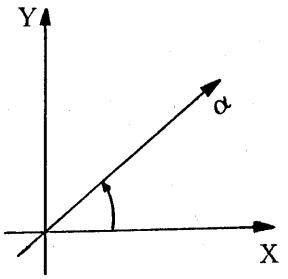

Fig. 1. Orientation α defined in the coordinate system.

As is well known [4], polygonal intersection searching can be reduced to three subproblems:

(1) polygonal range searching

(2) polygonal point enclosure searching

(3) segment intersection searching

In the following the c-oriented versions of the three subproblems are respectively solved. Multiple reports caused by singular cases can be easily avoided using extra care. Combining these results (see Theorem 2, Theorem 3 and Theorem 4), we obtain:

Theorem 1 *Given a set S of n c-oriented polygons and a c-oriented query polygon q, the t polygons in S intersecting q can be reported in $O(\log n + t)$ time using $O(n \log n)$ space and preprocessing time.*

This improves Güting's result [9] of $O(log^2 n + t)$ time, $O(n \log n)$ space and $O(n \log^2 n)$ preprocessing time. For comparison, let us consider the complexities of the best known algorithms developed for rectangles and arbitrarily oriented polygons. Combining the recent results for the three problems [1, 13, 17], we can solve the general problem in $O(\log n + t)$ time, $O(n^4)$ space and $O(n^4 \log n)$ preprocessing time. When polygons are restricted to rectangles, the problem can be solved in $O(\log n + t)$ time, $O(n \log n / \log \log n)$ space and $O(n \log n)$ preprocessing time [1].

2.1 Polygonal range searching

In this section we study c-oriented polygonal range searching : Given a set S of n points and a c-oriented query polygon q, report all the points of S that lie in q.

There exist two kinds of solutions for the general problem (with an arbitrarily oriented query polygon). The problem can be solved either in optimal $O(\log n)$ query time using $O(n^2)$ space [13] or in $O(n^{0.695})$ query time and optimal $O(n)$ space [6]. For a query rectangle, the enclosed points can be reported in $O(\log n)$ query time using $O(n \log n / \log \log n)$ space [1].

To solve the range searching problem, the key is how to decompose query polygon q. Here we decompose q into rectangles and triangles (with one vertical edge). See Fig. 2. Using the plane sweep method, q is divided into triangles and trapezoids (with two vertical edges), and then, each trapezoid is refined into one rectangle and at most two (right) triangles. It is obvious that a polygon with k edges can be partitioned in $O(k \log k)$ time into $O(k)$ rectangles and triangles.

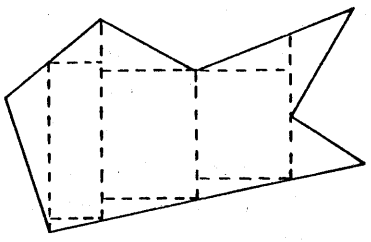

Fig. 2. A polygon divided into rectangles and triangles.

The problem is now solved by determining for each constituent rectangle or triangle the points that fall in it, and hence, reduced to two subproblems: rectangular range searching and triangular range searching. The first problem is well studied and can be solved in $O(log\ n)$ query time and $O(n\ log\ n\ /log\ log\ n)$ space [1]. To solve the triangular range searching problem, the c-orientedness becomes crucial. Each triangle (with one vertical edge) is characterized by the two orientations of its bottom and top edges, which leads to a classification of the resulting triangles of q into $(c\ -\ 1) * (c\ -\ 2)$ classes by the combination of edge orientations (We assume without loss of generality that some two orientations are parallel to the coordinate axes). All the triangles in a class have the important property that whose edges are parallel to three fixed orientations. When queries are restricted to this kind of triangles, the problem is called *homothetic range searching.* Chazelle and Edelsbrunner [2] have developed a so-called *priority search dag* that supports such range searching in $O(log\ n)$ query time, $O(n)$ space and $O(n\ log\ n)$ preprocessing time. Thus the point set S is organized into $(c\ -\ 1) * (c\ -\ 2)$ different priority search dags, one for each triangle class. With each triangle of query polygon q, we search the appropriate structure for the points that lie in the triangle.

Theorem 2 summarizes the results on c-oriented polygonal range searching.

Theorem 2 *Given a set S of n points and a c-oriented query polygon q with a bounded number of edges, the t points of S lying in q can be reported in $O(log\ n\ +\ t)$ time with $O(n\ log\ n\ /log\ log\ n)$ space and $O(n\ log\ n)$ preprocessing time.*

Taking c into account, the space requirement is $O(c^2\ n\ +\ n\ log\ n\ /log\ log\ n)$. The query time is independent of c. This improves Güting's result of $O(log^2 n\ +\ t)$ time and $O(c^2\ n\ log\ n)$ space. In practical applications, c must be rather small to make our solution efficient.

2.2 Polygonal point enclosure searching

C-oriented polygonal point enclosure searching is defined as follows: Given a set S of n c-oriented polygons and a query point q, report all the polygons in S which enclose q.

For the general problem, Tan et al. [17] obtain the result of $O(log\ n)$ query time and $O(n^2)$ space. In contrast, the optimal solution of $O(log\ n)$ query time and $O(n)$ space for the rectangular version of the problem is already given by Chazelle [1].

Let us see what kind of result we can obtain for the c-oriented problem. Again, all the given polygons are divided into rectangles and triangles. The problem is reduced to two subproblems: rectangular point enclosure searching and triangular point enclosure searching. With the *hive-graph* [1], rectangular point enclosure searching can be solved in $O(log\ n)$ query time and $O(n)$ space. To solve the second problem, the set of triangles is split into $(c\ -\ 1) * (c\ -\ 2)$ disjoint subsets according to the combination of edge orientations. Many subsets may be empty. Note that the cardinality of the union of all subsets is $O(n)$. On each subset we consider the point enclosure searching problem. Let $S_{\alpha,\beta}$ denote the subset whose bottom edge and top edge have orientations α and β respectively. See Fig. 3 (Suppose that the vertical edge is the right edge). A triangle in $S_{\alpha,\beta}$ can then be denoted by a triple $(x_0,\ \overline{\alpha_0},\ \overline{\beta_0})$, where $\overline{\theta}$ is the orientation

perpendicular to orientation θ. Query point $q = (x_q, \overline{\alpha_q}, \overline{\beta_q})$ is enclosed by a triangle of $S_{\alpha,\beta}$ if and only if the following conditions jointly hold:

$$x_q \leq x_0, \qquad \overline{\alpha_0} \leq \overline{\alpha_q}, \qquad \overline{\beta_q} \leq \overline{\beta_0}.$$

These conditions are trivially transformed into a well-known relation "$\prec$" of dominance between two 3-dimensional points, that is

$$(x_q, -\overline{\alpha_q}, \overline{\beta_q}) \prec (x_0, -\overline{\alpha_0}, \overline{\beta_0}).$$

Thus, triangular point enclosure searching becomes the *dominance searching* problem in three dimensions.

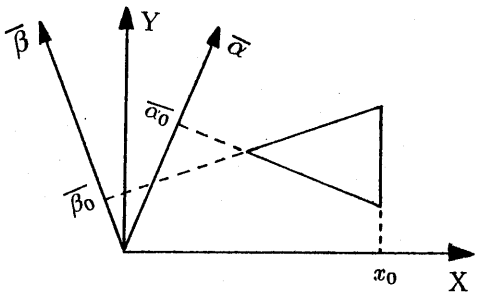

Fig. 3. A triangle in the $(x, \overline{\alpha}, \overline{\beta})$ coordinate system.

Gabow et al. [7] have given a solution to 3-dimensional dominance searching with $O(log\ n)$ query time, $O(n\ log\ n)$ space and preprocessing time. While Chazelle and Edelsbrunner [2] trade space for query time and obtain a result of $O(log^2 n)$ query time, $O(n)$ space and $O(n\ log^2 n)$ preprocessing time. Either structure of them can be employed. For c-oriented polygonal point enclosure searching, $(c-1)*(c-2)$ such structures are used. Given a query point q, search all the structures for the points dominating it. This completes the solution of c-oriented polygonal point enclosure searching.

Theorem 3 *Given a set S of n c-oriented polygons with a bounded number of edges and a query point q, the t polygons of S enclosing q can be reported in $O(log\ n + t)$ $(O(log^2 n + t))$ time using $O(n\ log\ n)$ $(O(n))$ space and $O(n\ log\ n)$ $((n\ log^2 n))$ preprocessing time.*

Taking c into account, the query time is $O(c^2\ log\ n + t)$ $((c^2\ log^2 n + t))$. The space requirement is independent of c. This improves Güting's result of $O(c^2 log^2 n + t)$ time and $O(n\ log\ n)$ space.

2.3 Segment intersection searching

C-oriented segment intersection searching is: Given a set S of n c-oriented segments and a c-oriented query segment q, report all the segments in S that intersect q. Chazelle [1] has given a solution of $O(log\ n)$ query time and $O(n^4)$ space for arbitrarily oriented segments and an optimal solution of $O(log\ n)$ query time and $O(n)$ space for orthogonal segments.

As Güting [9] did, we can simply reduce the c-oriented segment intersection searching problem to several instances of the orthogonal problem. Using the hive-graphs, we obtain:

Theorem 4 *Given a set S of n c-oriented segments and a c-oriented query segment q, the t segments of S intersecting q can be reported in $O(\log n + t)$ time using $O(n)$ space and $O(n \log n)$ preprocessing time.*

Güting uses the layered segment-binary trees instead of the hive-graphs, which requires to increase the space bound by a factor *log n*. Again, the hidden multiplicative function of c in our result is quadratic (see [9] for details).

3 The dynamic Solution

We now consider the intersection searching problem on a dynamic set of c-oriented polygons. Most of the data structures described in Section 2 can not be applied in the dynamic case. We take a different approach, which is mostly borrowed from Güting [9]. Güting's algorithm mainly benefits from the standard data structures, such as, *segment trees, range trees, priority search trees and interval trees* [12, 14].

For dynamic triangular range searching, $(c-1)*(c-2)$ range-priority search trees can be used, one for each class of query triangles. Consider the class of query triangles whose bottom edge and top edge have orientations α and β respectively (see also Fig. 3). All points in S are stored in the top level range tree over the x coordinates and the set of points associated with each node is then organized into a priority search tree over the $(\overline{\alpha}, \overline{\beta})$ coordinates. To retrieve the points of S lying in a query triangle $(q_x, q_{\overline{\alpha}}, q_{\overline{\beta}})$, one can perform two consecutive searches with the query ranges $([-\infty, q_x])$ and $([q_{\overline{\alpha}}, +\infty], [-\infty, q_{\overline{\beta}}])$. It follows that the range-priority search tree permits the retrieval of the enclosed points in $O(log^2 n)$ query time using $O(n \log n)$ space. Furthermore, any point can be inserted or deleted in $O(log^2 n)$ time. Range-priority search trees also suit for dynamic rectangular range searching. For a query rectangle, say, described by a quadruple $(q_{x1}, q_{x2}, q_{y1}, q_{y2})$, the query ranges become $([q_{x1}, q_{x2}])$ and $([q_{y1}, +\infty], [-\infty, q_{y2}])$.

As Güting shown [9], the solution for c-oriented range searching can be inverted for c-oriented point enclosure searching. These two problems are dual; only the roles of query objects and queried set are exchanged. This means that segment-priority search trees can be applied for both dynamic triangular point enclosure searching and rectangular point enclosure searching, with the same bounds as in the range searching case.

The dynamic solution of orthogonal segment intersection searching is already given by McCreight [11]. The space requirement is $O(n)$ and the query time is $O(log^2 n)$. The data structure is actually an interval-priority search tree. Combining the results on the three subproblems, we obtain:

Theorem 5 *Let S be a dynamic set of n c-oriented polygons. It is possible to represent S in a data structure with $O(n \log n)$ space and $O(log^2 n)$ update time, so that the t polygons in S intersecting a c-oriented query polygon can be reported in $O(log^2 n + t)$ time.*

This reduces the space requirement of Güting's algorithm by a factor $log\ n$.

4 Applications

The solution of c-oriented polygonal intersection searching has many applications, not only in 2-dimensional space but also in 3-dimensional space. The first example is to report all intersecting pairs in a set of c-oriented polygons. The problem can be solved by the *repeated searching* method. As an immediate consequence of Theorem 5, we can report the t intersecting pairs in a set of n c-oriented polygons in $O(n\ log^2 n\ +\ t)$ time and $O(n\ log\ n)$ space (see [9] for details). Using the plane sweep method, Tan et al. [18] present an optimal algorithm of $O(n\ log\ n\ +\ t)$ time and $O(n)$ space. In three dimensions, the efficient algorithms have been given for the *hidden line removal problem* [10] and the *translation problem* [16] considered on a set of c-oriented objects, that is, whose bounding faces have only a constant number of orientations. It is expected that the methods developed in this paper can also be applied to other c-oriented problems.

References

[1] B. Chazelle, Filtering search: a new approach to query-answering, *SIAM J. Comput.* **15**(1986), 703-724.

[2] B. Chazelle and H. Edelsbrunner, Linear space data structures for two types of range search, *Discrete Comput. Geometry* **2**(1987), 113-126.

[3] J.Culberson and G.J.E.Rawlins, Turtlegons: Generating simple polygons from sequences of angles, in *Proceedings, ACM Symp. Comput. Geometry*(1985), pp. 305-310.

[4] H. Edelsbrunner, D. G. Kirkpatrick and H. A. Maurer, Polygonal intersection searching, *Inform. Process. Lett.* **14**(1982), 74-79.

[5] H. Edelsbrunner and H. A. Maurer, On the intersection of orthogonal objects, *Inform. Process. Lett.* **13**(1981), 177-181.

[6] H. Edelsbrunner and E. Welzl, Halfplanar range search in linear space and $O(n^{0.695})$ query time, *Inform. Process. Lett.* **23**(1986), 289-293.

[7] H. N. Gabow, J. L. Bentley and R. E. Tarjan, Scaling and related techniques for geometry problems, in *Proceedings, 16th Annu. ACM Symp. Theory of Computing* (1984), pp. 135-143.

[8] R. H. Güting, Stabbing c-oriented polygons, *Inform. Process. Lett.* **16**(1983), 35-40.

[9] R. H. Güting, Dynamic c-oriented polygonal intersection searching, *Inform. Control* **63**(1984), 143-163.

[10] R. H. Güting and Th. Ottmann, New algorithms for special cases of hidden line elimination problem, *Comput. Vision Graphics Image Process.* **40**(1987), 188-204.

[11] E. M. McCreight, Priority Search Trees, *SIAM J. Comput.* **14** , 1985, 257-276.

[12] K. Mehlhorn, "Data Structures and Algorithms 3: Multi-dimensional Searching and Computational Geometry," Springer-Verlag, Berlin, 1984

[13] M.S.Paterson and F.F.Yao, Point retrieval for polygons, *J. Algorithms* **7**(1986), 441-447.

[14] F.P.Preparata and M.I.Shamos, *Computational Geometry*, Springer-Verlag, 1985.

[15] G.J.E.Rawlins and D.Wood, Optimal computation of finitely oriented convex hulls, *Inform. Computation* **72**(1987), 150-166.

[16] X. -H. Tan, T. Hirata and Y. Inagaki, On translating a set of c-oriented objects in three dimensions, Tech. Rep., Engineer. Dept., Nagoya University, 1989.

[17] X. -H. Tan, T. Hirata and Y. Inagaki, Spatial point location and its applications, to present in SIGAL International Symposium on Algorithms, Tokyo, 1990.

[18] X. -H. Tan, T. Hirata and Y. Inagaki, Reporting intersections of c-oriented polygons, Tech. Rep., Engineer. Dept., Nagoya University, 1990.

[19] P.Widmayer, Y.F.Wu and C.K.Wang, Distance problems in computational geometry for fixed orientations, in *Proceedings, ACM Symp. Comput. Geometry*(1985), pp. 186-195.