グラフィクスとCAD 33－4
（1988. 7. 22）

ウィンド・クリッピング問題の計算複雑さ

浅野　哲夫
大阪電気通信大学工学部応用電子工学科

コンピュータ・グラフィックスにおける最も基本的な問題の一つに、任意に指定された軸平行な長方形の窓に含まれる全ての線分を列挙するという問題（ウィンドウ・クリッピング問題）がある。この問題に対して従来から多数のアルゴリズムが提案されているが、問題の計算複雑さについては余り研究されていないようである。本報告では、ウィンドウ・クリッピング問題の計算複雑さについて考察し、記憶量を線分数に比例する程度に限定したとき、どのようなアルゴリズムでも最低線分数の平方根に比例する程度の時間は必要であるという結果を得た。そのほか、ウィンドウ・クリッピング問題に対する効率の良いデータ構造についても検討している。

Computational Complexity of Window Clipping Problem

Tetsuo Asano

Department of Applied Electronics,
Osaka Electro-Communication University
18-8 Hatsu-cho, Neyagawa, 572 Japan

One of the most fundamental problem in computer graphics is to enumerate all the line segments that intersect an arbitrary axis-parallel rectangular window, known as the window clipping problem. The purpose of this paper is to investigate the computational complexity of the problem and to present an algorithm and a data structure for solving the problem efficiently. Some pessimistic result is shown, that is, the lower bound of the time required to answer such a query is proportional to the square root of the number of line segments if storage proportional to the number of line segments is available.

1. Introduction

Due to the recent remarkable development of the technology in computer graphics geometrical data processing has become more and more important. To meet the requirement a number of efficient algorithms and data structures suitable for storing geometrical information has been proposed. In fact a number of commercial line drawing programs are available.

One of the most fundamental tasks in those programs are to define a new line segment by specifying two endpoints on the screen by a mouse cursor or other equipment and to delete or move a line segment by specifying a point near a target line segment. To be precise, we would like to find a line segment which intersects a small fixed size square or a small circle of fixed radius. Such a task is known as line clipping. Another important task is to clip line segments by a (large) axis-parallel rectangular window, that is, to enumerate all the line segments that intersect an arbitrary specified window, which is known as window clipping. Those two problems are similar to each other, that is, both of them are to find intersections with a query rectangle. The difference is the size of a query rectangle.

A number of algorithms have been proposed for the window clipping problem. The first most-commonly used one is the Cohen-Sutherland algorithm [NS79]. Then, Cyrus-Beck algorithm [CB78], Liang-Barsky algorithm [LB84], Sobokow-Pospiril-Yang algorithm [SPY86], and Nicholl-Lee-Nicholl algorithm[NLN87] followed. The purposes of all those algorithms were to reduce the number of arithmetic operations to determine whether a line segments a query rectangle and to do some experiments on the computation time. From a standpoint of asymptotic analysis of computation time all those algorithms can be said to be exhaustive methods. In this paper we are interested in how much we can reduce the number of line segments tested asymptotically.

The first efficient algorithm was presented by M.H. Overmars[Ov85a] in 1985. Assuming that there is no intersection among given line segments, he presented an algorithm which can answer a query in O(k + log n) time, where k is the number of line segments reported, and which builds a data structure in O(n log n) time using O(n) space for fixed sized windows and O(n log n) space for arbitrary sized windows. He also presented an algorithm for dynamic version of the problem, i.e., which answers a query in $O(k + \log^2 n)$ time, inserts a new line segment in $O(\log^2 n)$ time and deletes a line segment in O(log n) time using O(n log n) storage. The data structure used is a BB[a]-tree, similar to an interval tree. However, in an ordinary situation his assumption that there is no pair of line segments intersecting each other is not acceptable. Then, is there any algorithm which can answer a query in poly-log time, say in $O(\log^2 n)$? It will be shown that the answer is "No" if only O(n) or O(n log n) space is allowed. Then, we present an algorithm with sublinear time and a data structure for it.

Unfortunately, in a practical situa-tion the assumption of no intersection among given line segments is not acceptable. No efficient algorithm has been known for the general case. In this paper we present an algorithm which enumerates all the line segments inter-secting a query rectangular window in $O(k + \sqrt{n}\log^3 n)$ time among n given line segments using O(n log n) storage. Then, based on the recent results by Chazelle[Ch87] we show that these bounds for space and query time are optimal up to polylog-factors. That is, we prove a half planar query whose lower bound has been proved to be $O(k + \sqrt{n})$ can be reduced to a window clipping query in linear time and space.

2. The Case of No Intersection

If there is no intersection among line segments, we can construct an algorithm for answering a window clipping query in poly-log time using linear storage. Such an algorithm was presented by Overmars[Ov85a, Ov85b]. In his algorithm two data structures are used; one for storing endpoints of line segments and the other for open line segments. Using the first data struc-ture we can find all those line segments having their endpoints within a query rectangular window. The other is for enumerating all line segments that cross a horizontal or vertical side of the window.

The first data structure is realized by a range tree proposed by Bentley and Friedman [BF79]. Using the data structure, we can store a set of n points in the plane using O(n log n) storage such that range queries can be answered in $O(k + \log^2 n)$ time, where k is the output size. Moreover, we can insert a new point in $O(\log^2 n)$ time and delete a point in O(log n) time.

For the second data structure for storing a set of line segments so that those line segments crossing at least one side of a query rectangular window can be found efficiently, Overmars [Ov85a] proposed a segment-tree like data structure which he called BB[a]-tree.

Here we show that an ordinary segment tree is enough to enumerate all the line segments crossing a side of a query window. Since a window consists of four sides, two vertical and two horizontal, what we have to do is to show that given an arbitrary line segment s, which is horizontal or vertical, we can find line segments intersecting s in an efficient way. For simplicity we assume that there is no vertical or horizontal line segment.

We build a horizontal segment tree by projecting all line segments onto the x-axis, that is, a line segment is represented by its horizontal interval. A line segment s is stored in every node v such that the x-interval covers that corresponding to v while it does not cover that of the parent of v. In each node line segments associated with it are stored in a balanced binary tree in the sorted order of their y-coordinates of intersections with left boundary of the interval corresponding to the node. Similarly a vertical segment tree is built.

Given an axis-parallel rectangular window, we perform segment-intersection query for four sides of the window. For a horizontal side h we use the vertical

segment tree. First we find a leaf node v associated with the y-coordinate of h and enumerate all the ancestors of v. At each such node we first locate the leftmost endpoint of h and then locate the rightmost one. Since line segments are stored in a balanced binary tree with x-coordinates of their intersections with top horizontal boundary of the interval corresponding to the node as the key. Then, it is easy to to see that we can locate them in O(log n) time. Thus, we can enumerate all the line segments intersecting an arbitrary specified horizontal line segment in O(k + log n) time even in the worst case, where k is the output size. Since there are O(log n) nodes to be examined, a total query-answering time is O(k + $\log^2 n$). Since we used a segment tree we need storage proportional to n log n.

3. An Algorithm for Window Clipping

In the preceding section we showed that we can answer a window clipping query in poly-log time under the assumption that there is no intersection among given line segments. In practical situations, however, such assumption is not acceptable. In this section we present a window-clipping algorithm without the assumption.

An algorithm to be presented is based on the range tree and the above-stated segment-tree data structure. Here we are concentrated on the problem of enumerating all line segments intersecting an arbitrary specified vertical line segment, which is a side of a query window. As before, we project each line segment onto the x-axis to represent each line segment by it x-interval.

Given a vertical line segment s, we first find a set of nodes of the segment tree whose x-interval contains the x-coordinate of the query line segment s. The problem is to enumerate all line segments among those ones stored in each such node that intersect s. The difficulty is here is that we cannot use a binary search on the set of line segments since the order of segments change at their intersections.

The idea is to view line segments in each node as infinite lines. Then, we use a duality transformation from a line to a point and from a point to a line (see, for example, a survey paper[LP84]). Using the transformation, line segments are mapped to points in the plane. A query vertical line segment is mapped into two parallel lines. What is required to find is to enumerate all points between the two parallel lines. The problem is almost the same as that known as half-planar search, for which an algorithm with linear storage and sublinear query time is known. First such algorithm is due to Willard [Wi82], the computational complexity of which is recently improved into linear storage and O($\sqrt{n} \cdot \log^2 n$) query time by Welzl[We88] using a partition-tree data structure.

Using the data structure, we can enumerate all point between arbitrary specified parallel lines in O(k + $\sqrt{n}$ $\log^2 n$) time using linear storage. Thus, the total query time would be O(k + $\sqrt{n}$ $\log^3 n$) time, where k is the output size.

4. Lower Bound for Window Clipping Problem

In this section we consider the

lower bound for the window clipping problem. We consider the following three problems.

Problem P1: Let S be a set of line segments in the plane. Given an arbitrary vertical line segment s, report all the line segments intersecting s.

Problem P2: Let S' be a set of lines in the plane. Given a vertical half line $(x_0, y_0) - (x_0, +\infty)$, report all the lines of S' intersecting the half line.

Problem P3: Let V be a set of points in the plane. Given an arbitrary line L, report all the points of V below L.

Problem P3 is known as a half-planar search problem. Very recently, Chazelle [Ch87] showed that the lower bound for a half-planar query is $O(\sqrt{n})$ time if only linear space is available to store given points.

As is easily seen, Problem P2 is easier than Problem P1. It is also seen that Problem P2 is equivalent to Problem P3 under duality transformation. Thus, we can conclude that Problem P1 is at least as difficult as Problem P3 for which $O(\sqrt{n})$ lower bound is known in its time complexity, that is, the lower bound of the time required to answer a query in Problem P1 is $O(\sqrt{n})$ if only linear space is allowed. So is the lower bound of the query time for the window clipping problem.

References

[BF79] J.L. bentley and J.H. Friedman, "Data Structures for Range Searching", ACM Comput. Surveys, 11, pp.397-409, 1979.

[CB78] M. Cyrus and J. Beck, "Generalised Two- and Three-Dimensional Clipping", Computers and Graphics, vol. 3, no. 1, pp.23-28, 1978.

[Ch87] B. Chazelle, "Polytope Range Searching and Integral Geometry", Proc. IEEE SYmposium on Foundations of Computer Sciences, 1987.

[DE84] D. P. Dobkin and H. Edelsbrunner, "Space Searching for Intersecting Objects", Proc. IEEE Symposium on Foundations of Computer Science, pp. 387-392, 1984.

[LB84] Y.D. Liang and B.A. Barsky, "A New Concept and Method for Line Clipping", ACM Transaction on Graphics, vol. 3, no. 1, pp.1-12, 1984.

[LP84] D.T. Lee and F.P. Preparata, "Computational Geometry -- A Survey", IEEE Trans. on COMPUTERS, vol. C-33, no. 12, pp.1072-1101, 1984.

[NLN87] T.M. Nicholl, D.T. Lee and R.A. Nicholl, "An Efficient New Algorithm for 2-D Line Clipping: Its Development and Analysis", Proc. ACM Symposium on Computer Graphics, pp.253-262, 1987.

[NS79] W.M. Newman and R.F. Sproull, "Principles of Interactive Computer Graphics", 2nd ed., McGraw-Hill, New York, 1979.

[Ov85a] M. H. Overmars, "Range Searching in a Set of Line Segments", Proc. 1st ACM Symposium on Computational Geometry, pp.177-185, 1985.

[Ov85b] M. H. Overmars, "Geometric Data Structures for Computer Graphics", Tech. Report RUU-CS-85-13, University of Utrecht, 1985.

[SPY86] M.S. Sobkow, P. Pospisil and Y.-H. Yang, "A Fast Two-Dimensional Line Clipping Algorithm", University of Saskatchewan Technical Report, 86-2.

[SS68] R.F. Sproull and I.E. Sutherland, "A Clipping Divider", FJCC 1968, Thompson Books, Washington, D.C., pp.765-775.

[We88] E. Welzl, "Partition Trees for Triangle Counting and Other Range Searching Problems", ACM Symposium on COmputational Geometry, 1988.

[Wi82] D. Willard, "Polygon Retrieval", SIAM J. Comput., 11, pp.149-165, 1982.