*ACM Symposium on Computational Geometry, May 1996*

# Maintenance of the Set of Segments Visible from a Moving Viewpoint in Two Dimensions

Sherif Ghali A. James Stewart
Department of Computer Science
University of Toronto
{ghali,jstewart}@dgp.toronto.edu

## 1 Introduction

Consider a viewpoint moving amongst a set of non–intersecting line segments in the plane. We would like to compute efficiently the set of segments visible at successive positions along the viewpoint trajectory. This problem can be solved in either the off–line or the on–line setting. The latter case, in which the updates of the viewpoint are known only as they occur, is more interesting for interactive systems.

This video illustrates a simple algorithm to solve this problem. For a scene containing $m$ visual discontinuities, each discontinuity crossing is performed in time $\mathcal{O}(\log^2 m)$ using space $\mathcal{O}(m)$. The space bound is an important consideration, as $m$ can become very large in real applications.

Pocchiola [Poc90] maintains each update in the view in logarithmic time for a viewpoint moving amidst a set of objects in the plane (off–line version). For $n$ objects, his algorithm computes the arrangement of $2n$ curves, which takes space quadratic in $n$. Bern et al. [BDEG90] address a three–dimensional version of the problem where the trajectory is a straight–line segment and forms part of the input. They show that it is possible in time $\mathcal{O}((n^2 + p) \log n)$ (where $p$ depends on the scene) to find the locations along the segment where the topology of the visible surfaces changes.

## 2 Outline of the approach

A *scene* consists of a set of $n$ segments and a viewpoint in the plane. We take the following steps to solve the problem:

- Build the visibility graph for the $n$ segments. The nodes in the visibility graph are the segment endpoints. Two nodes are joined by an edge if and only if the line segment connecting the corresponding endpoints does not intersect the interior of any scene segment. The line supporting the two endpoints is called a *discontinuity line*.

  Ghosh and Mount [GM87] construct the visibility graph in time $\mathcal{O}(m + n \log n)$ and space $\mathcal{O}(m)$ for a graph with $m$ edges. A simpler algorithm that reports the graph in time $\mathcal{O}(m \log n)$ and space $\mathcal{O}(n)$ has been described by Overmars and Welzl [OW88].

- Given the set of discontinuity lines, preprocess it into a data structure that a viewer traversing the scene can query to determine whether any discontinuity has been crossed.

- As long as no discontinuity line is crossed, the set of visible segments remains unchanged. If one is crossed, the data structure has to be updated and a segment might need to be inserted or removed from the set.

### 2.1 Detecting the intersection of the viewpoint trajectory with a discontinuity line

Consider the arrangement of the discontinuity lines that are defined by the visibility graph. Let $\mathcal{A}$ be the cell in which the viewpoint lies. As the viewpoint moves but remains inside $\mathcal{A}$, the set of visible segments remains unchanged. We want to compute a description of the boundary of $\mathcal{A}$ in order to intersect it with the viewpoint trajectory with each position increment. However, we do not want to construct the whole arrangement, since it would require storage quadratic in $m$.

We avoid computing the arrangement as follows: Compute the dual $S'$ of the set of discontinuity lines $S$ and the dual $V'$ of the viewpoint $V$ (the dual of line $ax + by = 1$ is point $(a, b)$ and vice versa). $V'$ divides the set of points $S'$ into two sets $S'_1$ and $S'_2$. Construct $C_1$, the convex hull of $S'_1$, and $C_2$, the convex hull of $S'_2$. Then the lines bounding $\mathcal{A}$ correspond, in the dual, to points on the boundaries of the two convex hulls. However, not all points on the boundaries of the convex hulls correspond to lines bounding the viewpoint cell. Figure 1 show this construction.

We store the two convex hulls in the data structure of Overmars and van Leeuwen [OvL80]. This structure permits dynamic maintenance of the convex hull as points are inserted

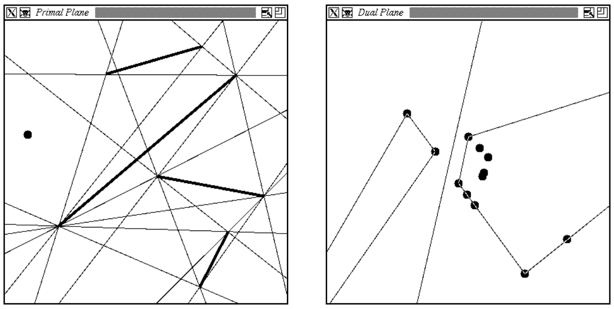


Figure 1: *Left:* The scene with segments in bold and discontinuity lines in thin, as well as the viewpoint. *Right:* The duals of the discontinuity lines, separated by the dual of the viewpoint. The convex hull of each separate set of points is shown. Note that some points on the convex hull fall outside the window.

and deleted at a cost of $\mathcal{O}(\log^2 m)$ time per operation, for a convex hull of $m$ points.

At each increment of the viewpoint position, we check whether the line segment connecting the old and the new viewpoint locations intersects the boundary of $\mathcal{A}$, as follows: A movement of the viewpoint $V$ from its old to its new location corresponds in the dual to a rotation of $V'$ about some point. We find the common tangents between that point and each convex hull in time $\mathcal{O}(\log m)$ and pick the tangent corresponding to the minimum rotation. The viewpoint crosses the boundary of $\mathcal{A}$ at discontinuity line $\ell$ if and only if the tangent touches the convex hull at the dual point, $\ell'$, and $\ell$ separates the old and new viewpoint locations.

As the viewpoint crosses a discontinuity line $\ell$ in the primal plane, the point $\ell'$ in the dual changes from one side to the other of the line $V'$. In this case, the point $\ell'$ must be removed from one convex hull and inserted into the other, which takes $\mathcal{O}(\log^2 m)$ time.

### 2.2 Updating the Set of Visible Segments

An initial view of the input segments is computed in $\mathcal{O}(n \log n)$ time by sweeping radially around the initial viewpoint. The output will be a circular list of the visible segments. This list is only updated if the viewpoint crosses a discontinuity line.

The set of visible segments is *not always* updated when the viewpoint crosses a discontinuity line (although the convex hulls must be updated to maintain the description of the current cell). This occurs in one of the following two situations: Either the viewpoint crosses a discontinuity line between the two endpoints that gave rise to that discontinuity, or the two endpoints are occluded from the viewpoint by a segment. These situations are easily detected.

## 3 Conclusion

We have described an on–line algorithm and data structure to maintain the set of segments visible from a viewpoint moving in a planar scene. The data structure requires only $\mathcal{O}(m)$ space in a scene of $m$ discontinuity lines, an important consideration in real applications. (In architectural scenes, we expect $m$ to be linear in the size of the scene because of the high degree of endpoint–to–endpoint connectivity among scene segments.) The algorithm takes time $\mathcal{O}(\log m)$ per increment in viewpoint location and time $\mathcal{O}(\log^2 m)$ per crossing of a discontinuity line.

### The Video

The algorithm was implemented in C++ on a SPARCstation under Solaris. The NTSC signal was recorded on a Sony BVU–950 $3/4''$ tape recorder. This was done at the Dynamic Graphics Project at the University of Toronto.

### Acknowledgement

We are grateful to the LEDA team at Max–Planck for making a robust library of data structures [MN95] publicly available, resulting in simplified development of this code. For the video, we appreciate the suggestions of the reviewing committee, and the technical help provided by Michiel van de Panne.

### REFERENCES

[BDEG90] M. Bern, D. Dobkin, D. Eppstein, and R. Grossman. Visibility with a moving point of view. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 107–117, 1990.

[GM87] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. In *Proc. 28th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 11–19, 1987. (also in SIAM J. Computing 1991, vol. 20).

[MN95] K. Mehlhorn and S. Näher. LEDA: a platform for combinatorial and geometric computing. *CACM*, 38:96–102, 1995.

[OvL80] M. H. Overmars and J. van Leeuwen. Dynamically maintaining configurations in the plane. In *Proc. 12th Annu. ACM Sympos. Theory Comput.*, pages 135–145, 1980.

[OW88] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 164–171, 1988.

[Poc90] M. Pocchiola. Graphics in Flatland revisited. In *Proc. 2nd Scand. Workshop Algorithm Theory*, volume 447 of *Lecture Notes in Computer Science*, pages 85–96. Springer-Verlag, 1990.