CMSC 754: Spring 1996 Samir Khuller

Visibility Graphs

Visibility graphs are defined as follows. Given a set S of n non-intersecting line segments in the plane, we wish to define a graph, where the vertices are the $2n$ endpoints of the line segments, and there is an edge from x to y, if and only if the open line segment $\overline{xy}$ does not intersect any segments in S. Visibility graphs can be used to find shortest paths between two given points in the plane (in the presence of obstacles). The notion of visibility graphs is well defined for polygons as well as intersecting line segments. For simplicity, we will not worry about those cases (the algorithm can be extended to those cases). We will study the $O(n^2)$ algorithm due to Welzl (*Information Processing Letters* Vol 20 (1985) pp. 167–171). The best algorithm for simple polygons is due to Ghosh and Mount that runs in time $O(E + n \log n)$ where E is the output size.

The main idea is the following: from each vertex (an endpoint), we shoot a ray in direction d and compute the segment that this ray hits. If the ray does not hit any segment, we will assume that hits the segment s_∞ (an artificial segment placed far away). It is easy to compute this information for a particular direction (say $-\infty$). Let $s(v)$ denote the segment that vertex v belogs to. Let $H(v)$ be the segment that is hit by a ray emanating from vertex v in direction d. We will slowly change direction d, increasing it, and maintain the segments that are currently hit by each ray emanating from a vertex. As we increase d, each ray continuously rotates around each vertex. Whenever a "critical" direction is crossed, we have to process an event. A critical direction is defined as the slope formed by a pair of vertices (assume that we measure the slope from x to y where x is to the left of y).

There are a few cases that we need to address when we process a critical direction. Let us deal with the easy cases first. Assume d_i the critical direction is defined by v and v', where v is to the left of v' (see Fig. 1). If $s(v) = s(v')$ then there is nothing to be done as these two points belong to the same segment. We now address the second case. Let w be the intersection point of the ray from v in direction d_i with segment $H(v)$. If $\overline{vw} < \overline{vv'}$ then there is no change either.

The interesting cases are when $H(v) = s(v')$. We *add* the edge vv' to the visibility graph, and set $H(v) = H(v')$, since after the critical direction is swept over, we have to update the segment hit by the ray.

The last case is when $H(v) = H(v')$, we *add* the edge vv' to the visibility graph, and set $H(v) = s(v')$, since after the critical direction is swept over, we are hitting the segment that v' is on.

The only remaining problem is to sort all the directions in increasing slope, so that all these events can be ordered and dealt with. This will unfortunately take $O(n^2 \log n)$ time. One way to deal with this problem is to examine more carefully why we need the slopes in strictly sorted order.

All we need is that when we process the slope vv', at this point of time v' has the correct value of $H(v')$. This is a weaker requirement than sorting all the slopes in order. It turns out that by computing the arrangements of the dual lines corresponding to the points, we can get this information in $O(n^2)$ time. Intuitively, all the rays will not be "rotating" in a synchronized manner as would happen if the slopes were all sorted, and the critical events were being processed in increasing slope order. We will permit the slopes to be processed in any order, so long as when we process vv' we have the property that v' has the correct value of $H(v')$. So, in some sense, even though the rotations are not synchronized, there are some properties that need to be maintained. Let us study this in more detail. Before we process the slope vv' (see Fig. 2(a)), we should have processed $v'w$ and not processed $v'x$. (In summary, a slope formed by two points u and y that are distinct from v and v', does not interfere with the processing of vv'.) Notice that the slopes are in the order $v'w < vv' < v'x$. Any processing order that respects this constraint would work correctly!

To compute this ordering, we dualize the $2n$ endpoints (mapping them to lines), and compute the arrangement of the corresponding lines. Recall that *slopes of pairs of points translate to x co-ordinates of the intersection points of the corresponding lines.* To obtain a legal ordering, we simply create the following directed acyclic graph (DAG)(see Fig. 2(b)). All intersection points in the arrangement are the vertices of the graph, put a directed edge between two intersection points on the same line that are "adjacent" (do not have any intersection points between them). Each edge is oriented from the vertex with the smaller x co-ordinate to the vertex with the higher x co-ordinate. Any topological order of this DAG would maintain the ordering property. This can be obtained in linear time by use of standard algorithms.

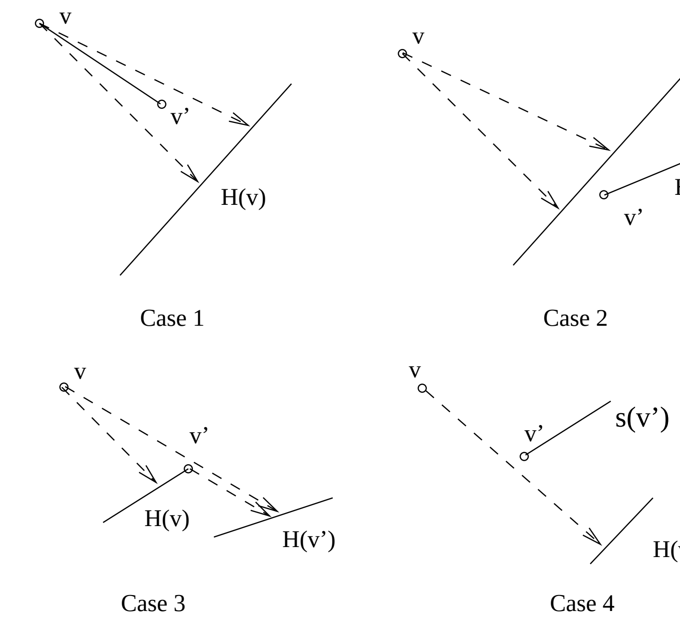

Figure 1: The four cases when we sweep over a critical direction.

Figure 2: Computing the DAG.