CISC 868 Fall 2011
Week 9

November 14, 2011

Line Arrangements

After convex hulls, Voronoi diagrams, and Delaunay triangulations the next most important concept in computational geometry are so called, line arrangements. This structure is based on an input that is a set of lines. That said its primary utility will be to solve problems on sets of points, by using an ingenious dual transform. We will postpone examining that transform until after we look at line arrangements, and an efficient algorithm to build one.

Let L be a set of lines in the two dimensional plane. As usual we will start with inputs that are non-problematic, that is, we will assume that the lines are in general position. The general position assumption implies that no two lines from L are parallel, and no three lines from L intersect at the same point. Thus the lines in L span the infinite plane, and outline a mosaic of faces. Observe that each face in the arrangement is convex, because it is in essence the intersection of half-planes. The vertices are intersection points of lines and the edges are line segments found between consecutive intersection points on a line. The size of an arrangement of n lines is $O(n^2)$. We know that the number of vertices is $\binom{n}{2}$, and the number of edges is n^2. Euler's relation can be used to determine the number of faces. That quantity can be found in Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars Computational Geometry:Algorithms and Applications (CGAA), but for now you can be satisfied that the numbers of faces is proportional to the number of vertices and edges, justifying the claim that $O(n^2)$ is the size of the arrangement. There is a technical difficulty dealing with the unbounded faces of the arrangement. That can be resolved by wrapping the plane over a sphere and thinking of a single point at infinity. A more pedestrian trick is to simply but a big rectangle around all the vertices of the arrangement, effectively adding four more vertices and a bunch of new edges. Nevertheless, the complexity of the "boxed in" arrangement is still $O(n^2)$ and now all of the faces are bounded.

A *line arrangement* of L, denoted by $\mathcal{A}(L)$, can be thought of as a data structure that stores the planar subdivision induced by L. One can assume that this is stored in something like a DCEL, so that one can navigate the subdivision, so that in constant time determine adjacent neighbours of edges, faces, or vertices of $\mathcal{A}(L)$. It is important to realize that simply computing the intersection points induced by the lines in L, is not enough to construct $\mathcal{A}(L)$. In essence getting the adjacencies

is akin to sorting the intersection points along each of the lines. If we in fact do this sorting, n sorts of $n-1$ intersection points on each of the lines, we can obtain $\mathcal{A}(L)$ in $O(n^2 \log n)$ time. Knowing that $\Omega(n \log n)$ is a lower bound for sorting one might think that a $O(n^2 \log n)$ algorithm is the best that we can hope for. However, the $\binom{n}{2}$ points are not any arbitrary collection of points. They emanate from n lines. It's possible that there is something special about the structure of these intersection points that would allow us to get the n sorted lists with less effort. (Take a moment to contemplate this paragraph to make sure you understand the subtleties that are involved.)

The remarkable result about arrangements is that we will be able to compute $\mathcal{A}(L)$ in $O(n^2)$ time. The algorithm is simple and incremental, using no data structure more complicated than the DCEL. The result is not an expected time result, rather it is a worst case result, that works for any arbitrary insertion sequence that is used. A full statement of the algorithm in its elegant simplicity is found in Algorithm MakeArrangement.

Algorithm 1: MakeArrangement

Input: A set of n lines L
Output: A DCEL D storing the arrangement $\mathcal{A}(L)$.
1 initialize D empty;
2 **for** *i = 1 to n* **do** insert the next line into D;
3 **return** D;

The important detail missing in the description of Algorithm MakeArrangement is the procedure used to insert a line into the arrangement. This is easiest to explain through an illustrative example. Refer to Figure 1 for a pictorial rendition of the insertion algorithm. Note that this figure is copied from CGAA. Please refer to CGAA for further details.

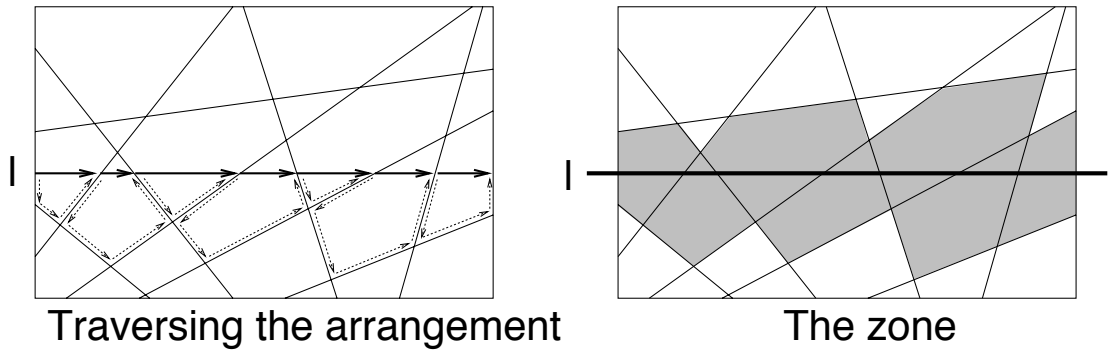

Figure 1:

Let L denote a set of n lines and let $\mathcal{A}(L)$ its arrangement. Now consider inserting one additional line ℓ into $\mathcal{A}(L)$. The *zone* of ℓ is the collection of faces in $\mathcal{A}(L)$ that are incident to ℓ. Algorithm MakeArrangement is implemented so that the complexity of inserting ℓ into $\mathcal{A}(L)$ is directly proportional to the complexity of the zone of ℓ. Observe that there may be faces in the zone of ℓ that themselves have complexity $O(n)$, so some subtle counting argument is needed to show that the complexity of the whole zone of ℓ is $O(n)$. This result is known in the literature as the zone theorem. The proof of this theorem is covered in detail in CGAA. It should now be apparent

that the zone theorem implies that $\mathcal{A}(L)$ can be computed in $O(n^2)$ time using a simple insertion algorithm.

Point-Line Duality

The utility of an arrangement of lines is revealed when one sees that a set of points can be represented in a dual space as a set of lines. The dual transformation is useful because the line arrangement reveals a structure that is not apparent when one considers the points. Consider a point $p : (p_x, p_y)$. The dual transform of p is the line $p^* : y = p_x x - p_y$. The dual of a line $\ell : y = mx + b$ is the point $\ell^* : (m, -b)$. We will refer to the "space" that points like p live in as the *primal space* and the space that the line p^* inhabits as the *dual space*. Suppose the point p lie on line ℓ. Thus we have

$$p_y = mp_x + b. \tag{1}$$

If we look to dual space we can verify that equation (1) implies that the point l^* lies on the line p^*. Furthermore, if ℓ is a line passing through two points p and q, the dual lines p^* and q^* must intersect at the point ℓ^*.

Now consider a set of points in the plane P, and we want to determine whether three or more lie on the same line. One way to do this would be to determine the line equation for all pairs of points and then see if any two are the same. The lower bound for determining whether any two of $O(n^2)$ objects are the same is $\Omega(n^2 \log n)$. However by using the insertion algorithm to construct arrangements and the dual transform this very same problem can be solved in $O(n^2)$ time. This is just one example where duality and line arrangements reveal structure in the input that was not apparent in the primal space.