Visibility graph of a set of line segments: A dynamic sequential algorithm and its parallel version

Yosser ATASSI
CRIN-CNRS BP 239
54506 Vandoeuvre-lès-Nancy
Cedex France (e-mail: atassi@loria.fr)

I Introduction

The visibility graph of a set of nonintersecting line segments G in the plane is a graph whose vertices are the endpoints of the segments and whose edges are the pairs of endpoints (u,v) such that the open line segment between u and v does not intersect any of the line segments of G.

Asano et al. [AGHI86] and Welzl [Wel85] presented optimal algorithms for constructing the visibility graph for n line segments in $O(n^2)$ time. Hershberger [Her89] studied the visibility graph of triangulated simple polygon with n sides. He described an algorithm that finds the visibility graph in $O(m)$ time where m is the number of edges in the visibility graph. Because m can be as small as $O(n)$. Vegter [Veg91] presented an algorithm to maintain dynamically the visibility graph of a set of n line segments in the plane in $O(log^2\ n + Klog\ n)$ time, where K is the total number of arcs of the visibility graph that are destroyed or created upon insertion or deletion of a line segment.

Goodrich et al. [GSG92] presented an algorithm for computing the shortest path between two vertices in a simple polygon by constructing the visibility graph in $O(n)$ time using $O(nlog\ n + \frac{m}{log\ n})$ processors, where m is the number of edges of the resulted visibility graph.

In this paper, an on-line algorithm and its parallel version to construct the visibility graph of a set of line segments using the dynamic data structure I-DAG are developed. This algorithm is simple, easy to code and efficient in practice.

II I-DAG (Influence Directed Acyclic Graph)

Randomised incremental construction is a new paradigm in computational geometry that has been successfully applied to a variety of problems [CS89] [BDS+92] [BY95] [Dev96]. These algorithms are rather simple, easy to code and efficient in practice. Moreover, they do not require that the input data satisfy some probabilistic distribution but that they are inserted in random order.

In this structure, geometrical problems are stated in terms of objects, regions and conflicts between objects and regions.

The *objects* are member of a universe U and are the input data of the problem. The *regions* are defined by subsets of the universe U of cardinality less than a constant b. The definition of conflicts between objects and regions has to be made precise for each specific problem. The subset of objects of U which are in conflict with a region is called the *influence range* of the region.

For a finite set of objects G we denote $F(G)$ the set of regions defined by the objects in G. We call the number of objects of G that belong to the influence range of a region width with respect to G. Let $F_j(G)$ be the subset of $F(G)$ consisting of the regions that have width j with respect to G.

Let G be the set of objects which have already been introduced. At a given stage, the incremental algorithm inserts a new object in G and updates the set $F_0(G)$ of regions of zero width defined by G. This is performed through the maintenance of a dynamic structure called Influence DAG (I-DAG) described below.

The I-DAG is a rooted directed acyclic graph whose nodes are associated with regions that at some stage of the algorithm have appeared as regions of zero width defined by the set of objects that have been introduced at that stage.

The construction of I-DAG can be sketched as follows:

- We initialise G with the first b objects. A node of the I-DAG is created for each region of $F_0(G)$ and made a child of the root of I-DAG.
- At each subsequent step, a new object O is added to G and the I-DAG is updated. The two following substeps are performed:
  - *Location substep.* This substep finds all the nodes of the I-DAG whose regions have zero width and are in conflict with O.
  - *Creation substep.* From the information collected during the location substep, the creation substep creates a new node for each region $F_0(G \cup \{O\}) - F_0(G)$ and links the new nodes to already-existing nodes in the structure.

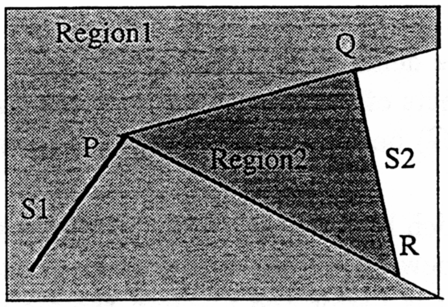
We have the following main theorem [BDS+92]:

Theorem 1 *If the set of already-inserted objects G has cardinality n, the I-DAG of G requires $O(\sum_{j=1}^{n} \frac{f_0(\lfloor \frac{n}{j} \rfloor, G)}{j})$ expected memory space. The insertion of a new object can be done in $O(\frac{\sum_{j=1}^{n} f_0(\lfloor \frac{n}{j} \rfloor, G)}{n})$ expected update time.*

III A sequential algorithm for constructing the visibility graph of a set of line segments

We present in this section a sequential on-line algorithm for constructing the visibility graph of a set G of n line segments in the plane.

The objects are the segments of G. A region is an angular sector QPR defined by three endpoints P, Q, and R such that $PQ \notin G$ and $PR \notin G$ (see Figure 1), the angle QPR can be more than π. The region corresponding to an angular sector QPR is called PQR. When $QR \in G$ and the angle QPR is less than π, the region PQR is bounded by the line segment QR. A line segment S is in conflict with a region PQR if and only if it intersects this region. A region of I-DAG has zero width if and only if PQ and PR are



Region2 $\widehat{PRQ}$ corresponding to the sector QPR

Region1 $\check{PRQ}$ corresponding to the complement sector QPR

Figure 1: The region of the I-DAG

edges of the visibility graph of G. So computing the visibility graph of G is equivalent to computing the zero width regions.

Let us describe the algorithm. Suppose that I-DAG has been constructed for the subset S_I ($S_I \subset G$) and that we want to insert a new segment S ($S = V_1V_2$).

The location substep gives the regions LT of $F_O(S_I)$ intersected by S. The I-DAG is modified in the following manner:

1. The width of selected regions LT is incremented.
2. For every region $T \in LT$ and T in the form of $V_3V_4V_5$ do:
   (a) If S intersects the line segments V_3V_4 and V_3V_5 then (see Figure 2.a):

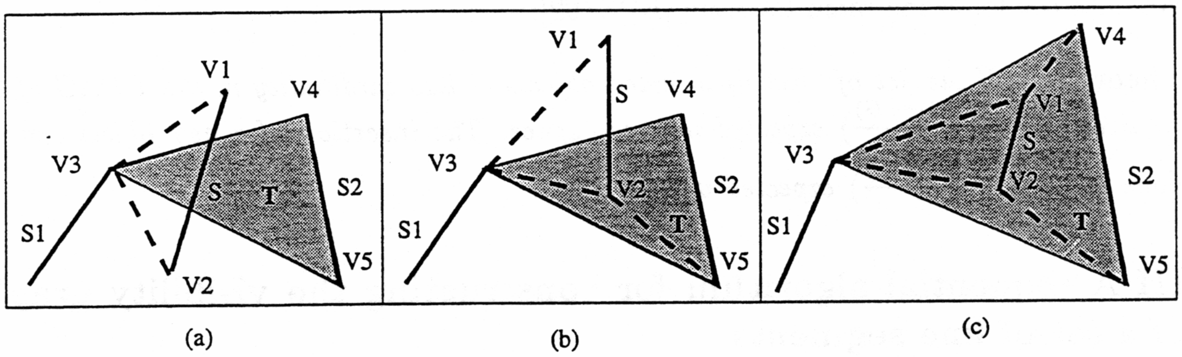


Figure 2: The relations between the line segment S and the region T

Let V_i be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_4 around V_3 in the inverse direction of V_5. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the inverse direction of V_4. We add the region $V_3V_iV_j$ as a child of T if this region has not been added before, else T has no child.

(b) Else, if S intersects only V_3V_4 then (see Figure 2.b):
Let V_i be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the direction of V_4. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_i around V_3 in the direction of the other endpoint of S. We add the regions $V_3V_5V_i$ and $V_3V_iV_j$ as children of T if these regions have not been added before.

(c) Else, if S intersects only V_3V_5 then:
Similar to the preceding case.

(d) Else, if S is in T then (see Figure 2.c):
Let V_i be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_4 around V_3 in the direction of V_5. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the direction of V_4. We add the regions $V_3V_4V_i$, $V_3V_iV_j$ and $V_3V_jV_5$ as children of T if these regions have not been added before.

3. For every endpoint V_3 of $T \in LT$ do:
   Let V_i be the first endpoint of S_I met while rotating V_1V_3 around V_1 counterclockwise. We add the region $V_1V_3V_i$ as a child of the I-DAG.

4. For every endpoint V_3 of $T \in LT$ do:
   Let V_i be the first endpoint of S_I met while rotating V_2V_3 around V_2 counterclockwise. We add the region $V_2V_3V_i$ as a child of the I-DAG.

IV A parallel algorithm for constructing the visibility graph of a set of line segments

We use the same definitions of object, region and conflict relation used in the preceding section. Suppose that I-DAG has been constructed for the subset S_I ($S_I \subset G$) and that we want to insert a new segment S ($S = V_1V_2$). We associate every endpoint V_i with a processor P_i which is responsible of regions whose first endpoint is V_i. At the location substep, every processor P_i tests if one of its regions of zero width has been influenced by the insertion of S, then P_i constructs the list of influenced regions LT_i.

At the creation substep, every processor P_i performs the sequential algorithm in the following manner:

1. The width of selected regions LT_i is incremented.

2. For every region $T_i \in LT_i$ and T_i in the form of $V_3V_4V_5$, the processor P_3 ($P_i = P_3$) associated to V_3 realises the following operations:

(a) If S intersects the line segments V_3V_4 and V_3V_5 then (see Figure 2.a)
Let V_k be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_4 around V_3 in the inverse direction of V_5. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the inverse direction of V_4. The processor P_3 adds the region $V_3V_kV_j$ as a child of T_i if this region has not been added before, else T_i has no child.

(b) Else, if S intersects only V_3V_4 then (see Figure 2.b):
Let V_k be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the direction of V_4. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_k around V_3 in the direction of the other endpoint of S. The processor P_3 adds the regions $V_3V_5V_k$ and $V_3V_kV_j$ as children of T_i if these regions have not been added before.

(c) Else, if S intersects only V_3V_5 then:
Similar to the preceding case.

(d) Else, if S in T_i then (see Figure 2.c):
Let V_k be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_4 around V_3 in the direction of V_5. Let V_j be the first endpoint of $S_I \cup \{S\}$ met while rotating V_3V_5 around V_3 in the direction of V_4. The processor P_3 adds the regions $V_3V_4V_k$, $V_3V_kV_j$ and $V_3V_jV_5$ as children of T_i if these regions have not been added before.

3. Two new processors P_1 and P_2 are used. They are associated respectively to the endpoints V_1 and V_2.

4. For every endpoint V_3 of $T_i \in LT_i$ do:
Let V_k be the first endpoint of S_I met while rotating V_1V_3 around V_1 counterclockwise. The processor P_1 adds the region $V_1V_3V_k$ as a child of the I-DAG.

5. For every endpoint V_3 of $T_i \in LT_i$ do:
Let V_k be the first endpoint of S_I met while rotating V_2V_3 around V_2 counterclockwise. The processor P_2 adds the region $V_2V_3V_k$ as a child of the I-DAG.

At last, every processor P_i goes over the list of its regions in the form of $V_iV_jV_k$ of zero width and gives as edges of the visibility graph, the line segments V_iV_j and V_iV_k.

V Average analysis

Here $f_0(r, G)$ is the expected size of the visibility graph of r segments, which is clearly $O(r)$, so applying Theorem 1 we deduce:

Proposition 1: the visibility graph of a set G of n segments in the plane can be computed sequentially with $O(m * log\ n)$ expected memory space and $O(m)$ expected update time where m is the number of edges of the resulted visibility graph.

Proposition 2: the visibility graph of a set G of n segments in the plane can be computed in parallel with $O(m * log\ n)$ expected memory space and $O(\frac{m}{P})$ expected update time where m is the number of edges of the resulted visibility graph and P the number of processors.

VI Conclusion

We have presented two new on-line algorithms for computing the visibility graph of a set G of line segments using the dynamic data structure I-DAG.

We can easily implement the parallel algorithm on machines with SPMD architecture. The communication among the processors is almost negligible, so this algorithm is efficient and faster than the sequential one.

These algorithms can be generalised easily to treat the case of a set of polygons.

References

[AGHI86] T. Asano, L. Guibas, J. Hershberger, and H. Iami. Visibility disjoint polygons. *Algorithmica*, 1:49–63, 1986.

[BDS+92] J.D. Boissonnat, O. Devillers, R. Schott, M. Teillaud, and M. Yvinec. Application of random sampling to on-line algorithms in computational geometry. *Discrete Comput. Geom.*, 8:51–71, 1992.

[BY95] J.D. Boissonnat and M. Yvinec. *Géométrie Algorithmique*. Ediscience international, Paris, 1995.

[CS89] K.L. Clarkson and P.W. Shor. Applications of random sampling in computational geometry. *Discrete and computational geometry*, 4:49–63, 1989.

[Dev96] O. Devillers. A comprehensive introduction to randomization in computational geometry. *Theoret. Comput. Sci.*, 157, 1996.

[GSG92] M.T. Goodrich, S.B. Shank, and S. Guha. Parallel methode for visibility and shortest-path problems in simple polygons. *algorithmica*, 8:461–486, 1992.

[Her89] J. Hershberger. An optimal visibility graph algorithm for triangulated simple polygons. *Algorithmica*, 4:141–155, 1989.

[Veg91] G. Vegter. Dynamically maintaining the visibility graph. In *Proc. 2nd Workshop Algorithms Data Struct.*, LNCS 519, pages 425–436, 1991.

[Wel85] E. Welzl. Constructing the visibility graph for n line segments in $O(n^2)$ time. *Information Processing Letters*, 20:167–171, 1985.