Finding the closest and visible sites for a line in the presence of barriers [1]

Cao An Wang [2]

Y.H.Tsin.[3]

Abstract: Let S be a set of n sites and L be a set of m disjoint line segments regarded as barriers. Let also l be a straight line outside of the convex hull of $L \cup S$. We present an algorithm to find the site for each point of l (if any) which is visible and closest to this point among the sites of S. The algorithm takes $O((mn + n^2)\ log\ (mn + n^2))$ time and $O(mn + n^2)$ space.

1 Introduction

Visibility problems from a single source (a point or an edge) have been extensively studied in the past [1,2,3,4,5]. A modified version of the above visibility problem is to consider multiple sources. In this paper, we consider the visibility of a straight line from a set of n sites in the presence of m obstacles (line segments). It is easy to see that there may exist mn distinct intervals in l such that each interval is visible from a subset of S and the adjacent intervals are visible from different subsets. Moreover, one may wish to identify the site closest to the corresponding interval among its visible sites. A brute-force method to find the closest and visible site for each interval of the line is as follows. Construct all $2mn + \frac{(n-1)}{n}$ lines determined by the $2m$ endpoints and n sites. The intersections of these line and the given line l divide l into line segments. For each line segment, test the visibility and compute the distance against each site. There may be $2mn + \frac{(n-1)}{n}$ line segments on l and the visibility test for each line segment may take mn time. Thus, this method will take $O((mn)^2 + mn^3)$ time in the worst case. In this paper, we shall present a faster algorithm for this problem. To do so, we solve two subproblems before solve the desired problem: (1) *effective* ray identification problem: Finding these rays, each of them emits at a site and passes through an endpoint of an obstacle, which determine the intervals of l. (2) off-line minimal problem: Finding the minimals of a sorted integer list with insertion and deletion operations.

2 Determining the effective rays w.r.t. the visibility of l

Let L be a set of m disjoint line segments, and let V be the endpoint set of L. Let S be a set of n sites. Let l be a straight line which does not intersect the convex hull $CH(S \cup L)$. Then, S and V will determine $2mn$ rays, each of them emits at a site of S and passes through an element of V.

[1]This work is supported by NSERC grant OPG0041629 and A7811

[2]Department of Computer Science, Memorial University of Newfoundland, St.John's, Newfoundland, Canada A1C 5S7 .3 School of Computer Science, University of Windsor, Windsor, Ontario, Canada N9B 3P4

Definition: The *polar upper envelope* of L w.r.t. $s \in S$ and l consists of the subsegments of elements of L which are polar visible and the segments of rays emitting at s each of which connects two such subsegments or connecting a subsegment to s.

Lemma 2.1: Ray $\vec{sv}$ is effective if entire $\overline{sv}$ lies on the polar upper envelope of L w.r.t. s and l.

By Lemma 2.1, we shall present an algorithm to find the polar upper envelope and then to identify the effective rays. To do so, we convert the polar upper envelope problem to an orthogonal envelope problem, then find the orthogonal upper envelope using a SET-UNION algorithm [6]. Clearly, the effective rays are determined by s and the endpoints of L corresponding to the endpoints of the disjoint segments of the orthogonal upper envelope.

The conversion is briefly described as follows. Let straight line l' pass through s and be parallel to l with L above l. Let ray $\vec{r}$ clockwisely sweep the lower halfplane of l' at center s and starting at the right half of l'. The intersections of $\vec{r}$ and the elements of L determine a sequence L' of line segments parallel to l'. Where the relative y and x coordinates of line segments l_i and l_j of L' are determined as follows: $Y(l_i)<Y(l_j)$ if r crosses l_j before l_i, $X(l_i)<X(l_j)$ if r touches the endpoint of l_i before that of l_j. (Refer to Fig.2.1, and [1] for details.) Clearly, the conversion takes $O(m \log m)$ time by a sweep-line method.

To find the orthogonal upper envelope of L', we represent a line segment l_i of L' by a triple (i,j,k), where i is the y-coordinate of l_i, j and k is the x-coordinates of its endpoints. The elements of L' are in ascending order so that l_i (denoted by (i,j,k)) is ahead of l_j (denoted by (x,y,z)) if $i < x$ or $i = x$ and $j < y$, Let S_t denote the sequence of triples. Initially, $2m$ integers on the x-axis are represented as a forest, where each integer is a tree with one node, the father of each node is empty. Let $r(p)$ denote the root of a tree with leaf p. For each root r, two pieces of information are attached: $R(r)$ denote the rightmost leaf of the tree, and $count(r)$ denotes the number of nodes in the tree. Initially, $R(r)$ and $count(r)$ are set to 1. S_t is put on a stack such that the first triple of S_t is on the top of the stack. **MergeTrees**$(r(x), r(y))$ is the UNION operation and **FindRoot(x)** is the path-compression in [6].

```
Algorithm FindSmallest(S_t, F)
WHILE S_t ≠ ∅ DO
    (i,j,k) ← POP(S_t);
    FOR p = j TO k DO
        IF father(p) = ∅ THEN
            IF p = j THEN
                r(p) ← j; father(p) ← j;
                count(r(p)) ← 1; F ← PUSH (i,p);
            ELSE (* p ≠ j *)
                father(p) ← r(j); count(r(p)) ← count(r(p)) + 1;
                F ← PUSH (i,p); R(r(j)) ← p;
        ELSE (* father(p) ≠ ∅ *)
            r(p) ← FindRoot(p); MergeTrees(r(j),r(p)); p ← R(r(p));
    ENDFOR
ENDWHILE
```

Lemma 2.2: Algorithm **FindSmallest** finds the orthogonal upper envelope of L' in $O(m)$ time.

By scanning the orthogonal upper envelope, we can find the effective rays emitting at s in $O(m)$ time. By applying the above method to each site in S, we obtain all the effective rays in $O(mn \log m)$ time. Let T denote the intersection points of these rays and l.

3 Finding minimals in a sorted list with updatings

Let N be a sublist of n integers sorted in ascending order, where the integers range from 1 to order n. Let S be a sequence of n operations. That is, $S = U_1 F_1 U_2 F_2 ... U_n F_n$, where update operation U_i for $1 < i \leq n$ is either a deletion of an element of N_{i-1} or an insertion of an integer j for $1 \leq j \leq$ order n into the proper place of N_{i-1}, which is the list of the sorted integers after operation U_{i-1}. Find operation F_i is to print the smallest element in N_i. We shall solve this problem by **FindSmallest**.

Let S_t denote a sequence of triples (i, j, k), where j and k are a pair of timestamps such that in time j, integer i is inserted into N_{j-1} and in time k, i is deleted from N_{k-1}. We sort these triples in lexicographic ordering. That is, for any two given pairs (i, j, k) and (x, y, z), (i, j, k) is *ahead* (x, y, z) in the sequence S_t iff $i < x$ or $i = x$ and $y < j$. By bucket sort, the following lemma is true.

Lemma 3.1: S_t can be found from S in $O(n)$ time and space.

Now, we can apply **FindSmallest(S_t, F)** to solve the problem in $O(n)$ time..

4 Finding the closest and visible sites of l

Definition: Let E' be a sequence of edges along l which are determined by the intersections of l and the perpendicular bisectors of n sites. Let CL_i be a list of n sites associated with the i-th edge e_i of l. CL_i is said to be the *closest site list*, if the first site in the list is the one closest to e_i (disregard the obstacles), and the second site in the list will be the one closest to e_i if the first site is not taken into account (for example, it is deleted due to the invisibility from e_i). In general, the k-th site for $1 \leq K \leq n$ will be the one closest to e_i if the first $k - 1$ sites is not taken into account.

Let B denote the intersections of l and the perpendicular bisectors. We sort the intersections of B and T along l and denote the sequence by Q. Clearly, Q divides l into a sequence E of edges ($\|E\| = O(mn + n^2)$). Let e_i and e_{i+1} be such two adjacent edges. Let VB_i and VB_{i+1} be their visible site lists, and CL_i and CL_{i+1} be their closest site lists. Then, the closest and visible site with respect to e_i (respectively, to e_{i+1}) is the first site in CL_i which also appears in VB_i.

Lemma 4.1: Let p be the shared point of e_i and e_{i+1}. If p belongs to an effective ray $\vec{su}$, then the difference of VB_i and VB_{i+1} is s. If p belongs to a bisector $b_{s,t}$, then then the difference of CL_i and CL_{i+1} is to swap the positions of s and t.

Now, we shall solve our problem using **FindSmallest(S_t, F)**. Let e_0 be one of the extreme edges of E. We shall find the visible site list and the closest site list of e_0 by a brute-force method. That is, to sort the distances between an interior point x of l and all $s \epsilon S$ to find the closest site list, and to test the intersections of line segment $\overline{xs}$ and all elements of L for all $s \epsilon S$ to determine the visible site list. (This can be done in $O(mn + nlogn)$ time.)

Starting at e_0 and based on the closest site list and the visible site list of e_0, we traverse l by scanning Q to construct a sequence of triples, where triple $((i,t),j,k)$ represents a site s_t appears in the i-th rank of CL_t's as well as VB_t's between j-th to k-th edges of l. Initially, we associate with each visible site of e_0 a triple with the rank of the site in CL_o and the index of the site as the first item, and 1 as the second item, and empty as the third item (e.g., $((i,t),1,\emptyset)$). If an encountered point of Q belongs to T and the site emitting the effective ray is s_p, then we create a new triple $((d,p),2,\emptyset)$ if s_p is a visible site of rank d w.r.t. the new edge, or we fill the third term of the old triple $((d,p),1,2)$ if s_p is invisible to the new edge. If an encountered point of Q belongs to B and the two associated sites are s_p and s_q, then we fill the third terms of the old triples $((d,p),1,2)$ and $((c,q),1,2)$ and create two new triples $((c,p),2,\emptyset)$ and $((d,q),2,\emptyset)$. Continue this scan until all elements of Q are exhausted. Sort the resulting triples lexicographicly w.r.t. the first integer of the first term and the second term and the resulting sequence of the triples is denoted by S_t, then apply **FindSmallest**(S_t, F) to S_t. The results in F is the closest and visible site (if any) for the corresponding edge of l.

Theorem 4.1: The closest and visible sites of l can be found in $O(mn + n^2)\ log\ (mn + n^2)$ time and $O(mn + n^2)$ space.

References

[1] Asano T., Asano T., Guibas L., Hershberger J., and Imai H., (1986), "Visibility of Disjoint Polygons", *Algorithmica* 1, pp.49-63.

[2] Avis D. and Toussaint G., (1981), "An Optimal algorithm for determining the visibility of a polygon from an edge", *IEEE Trans. on Computers*, Vol. C-30, No. 12, pp.910-914.

[3] El Gindy H. and Avis D., (1981), "A linear algorithm for computing the visibility polygon from a point", *J. Algorithms* 2(2), pp. 186-197.

[4] Bhattacharya B., Kirkpatrick D., and Toussaint G., (1989), "Determining sector visibility of a polygon", *Proceedings of 5th ACM Symposium on Computational Geometry*, Saarbruchen, West Germany, pp.247-254.

[5] Preparata F., Shamos M.,(1985), " *Computational Geometry* ", springer Verlag.

[6] Gabow H. and Tarjan R., (1983), "A Linear-time algorithms for a special case of disjoint set union", *Proceedings of the 15th Annual ACM symposium on Theory of Computing*, pp.246-251.

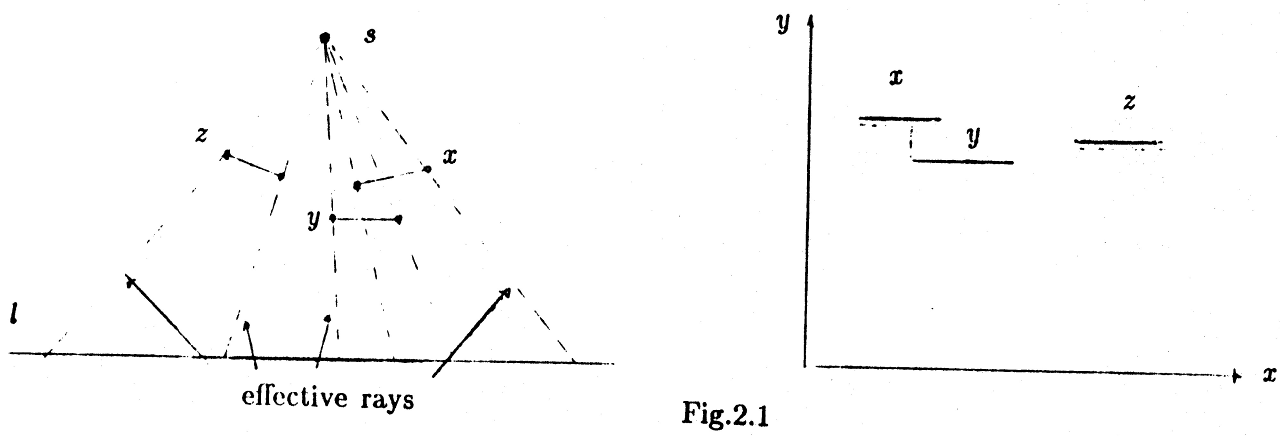

Fig.2.1