Weak Visibility of Two Objects in Planar Polygonal Scenes*

Mostafa Nouri, Alireza Zarei**, and Mohammad Ghodsi

[1] Computer Engineering Department
Sharif University of Thechnology
[2] IPM School of Computer Science

Abstract. Determining whether two segments s and t in a planar polygonal scene *weakly* see each other is a classical problem in computational geometry. In this problem we seek for a segment connecting two points of s and t without intersecting edges of the scene. In planar polygonal scenes, this problem is 3SUM-hard and its time complexity is $\Omega(n^2)$ where n is the complexity of the scene. This problem can be defined in the same manner when s and t are any kind of objects in the plane. In this paper we consider this problem when s and t can be points, segments or convex polygons. We preprocess the scene so that for any given pair of query objects we can solve the problem efficiently. In our presented method, we preprocess the scene in $O(n^{2+\epsilon})$ time to build data structures of $O(n^2)$ total size by which the queries can be answered in $O(n^{1+\epsilon})$ time. Our method is based on the extended visibility graph [1] and a range searching data structure presented by Chazelle *et al.* [2].

Keywords: Computational geometry, weak visibility, 3sum-hard problems, object inter-visibility.

1 Introduction

The problem of detecting visibility between objects has many applications in computer graphics, VLSI, motion planning and computational geometry. In computer graphics and simulations, for example, computing the regions illuminated by a fluorescent lamp in a scene may be needed. As the light source may be in different positions, we seek for a way to quickly find the lightened up regions in each position. This can be achieved by preprocessing the scene to do queries efficiently. However, various versions of visibility problems has been defined.

In this paper, we focus on *weak-visibility* between objects in a planar polygonal scene. Two objects s and t are said to be weakly visible from each other (or simply weakly visible) if a point of s sees a point of t. Two points see each other if the segment connecting them does not intersect edges of the scene. Given

* This work was partially supported by IPM school of computer science (contract: CS1385-2-01).

** This author's work was partially supported by Iran Telecommunication Research Center(ITRC).

O. Gervasi and M. Gavrilova (Eds.): ICCSA 2007, LNCS 4705, Part I, pp. 68–81, 2007.

two objects and a scene, the problem is whether these two objects are weakly-visible. When s and t are line segments, it has been proved by Gajentaan and Overmars [3] that this problem is in the class of 3SUM-hard problems and thus the lower bound of the time complexity of its solutions is $\Omega(n^2)$. Throughout this paper n is the complexity of the scene which is the number of its vertices or edges. Wismath [4] has presented an algorithm for this problem with optimal $O(n^2)$ time complexity. His method is based on the visibility graph which will be introduced in the next section.

The set of points of the scene that are visible from a point p is called its visibility polygon and is denoted by $VP(p)$. We know that $VP(p)$ is a star-shaped simple polygon. Visibility polygon can also be defined for a segment or polygon of a scene. Visibility polygon of a planar object s, or $VP(s)$, is the set of the points of the scene that are visible from at least one point of s. Generally $VP(s)$ is a polygon with holes.

We consider weak-visibility problem for two objects s and t, when these objects are points, segments or convex polygons. Also, we consider this problem in two cases: (1) when one of the objects is known in advance and the other one is given in query time, and (2) when both of the objects are given in query time. For the first case, we can preprocess the scene based on the given object say s, so that the queries for each t can be answered efficiently. This is done by first finding $VP(s)$ in the preprocessing step and then checking the intersection of t with this region in query time.

In the second case, the scene is preprocessed to build data structures by which the queries can be answered efficiently. Initially, we assume that the objects are line segments. In this case, we first preprocess the scene to find its extended visibility graph, to be explained later. Then we build a multi-level range searching structure on the edges of this graph. This range searching structure is based on the scheme proposed by Chazelle *et al.* to be discussed in the next section. Having this structure, we can find the edges of the extended visibility graph that are intersected by both query segments. We will show that if the intersection is not empty, then the query segments are weakly visible, otherwise, it is sufficient to check the weak-visibility of the endpoints of the query segments.

When the query objects are convex polygons, we will prove that the convex polygons are weakly visible if and only if two of their edges are weakly visible. Therefore, to solve the problem for convex objects, we just need to solve the problem for any pair of edges.

In a brief summary, we achieve the following results on weak-visibility problem in planar polygonal scenes when the complexity of the query objects is constant and an object can be a point, a segment, or a convex polygon.

- The weak-visibility between a query object and a given point can be answered in $O(\log n)$ time using $O(n \log n)$ and $O(n)$ preprocessing time and space, respectively.
- The weak-visibility between two query points can be answered in $O(\sqrt{n} \log n)$ time using $O(n \log^2 n)$ and $O(n\sqrt{n} \log^{4.3} n)$ preprocessing time and space.

- The weak-visibility between a query point and a line segment or a convex polygon can be answered in $O(n \log n)$ time.
- The weak-visibility between two query line segments or convex objects can be detected in time $O(n^{1+\epsilon})$ using $O(n^{2+\epsilon})$ and $O(n^2)$ preprocessing time and space, respectively.

Other than the weak-visibility problem, we have proposed a range searching method for determining the segments of a planar arrangement that are intersected by two given line segments. This solution can also be used in other range searching problems.

In the rest of this paper, the basic concepts and data structures are discussed in Section 2. Some properties of weak-visibility are described and proved in Section 3 and our methods and results are presented in Section 4. The materials are summarized and concluded in Section 5.

2 Basic Data Structures and Concepts

In this section we introduce the basic data structures and concepts that are used in our weak-visibility detection methods. We first describe the extended visibility graph of a scene. The edges of this graph define the boundaries of the regions with different views. Then, we describe a point location algorithm in a star-shaped simple polygon. This method help us to solve the weak-visibility problem when one of the objects is a point. Another problem that we have to solve as a subproblem in our method is ray shooting problem in a planar environment. If the environment was a simple polygon, we can do this work more efficiently than doing it in a planar arrangement as will be discussed next. Finally, we describe range searching in a planar scene and present a method for solving an special version of range searching: in a planar scene, find the set of segments intersected by two query segments.

To solve this problem, we extend the range searching scheme presented by Chazelle *et al.* [2]. As will be proved, we can answer this range searching problem in $O(n^{1/2+\epsilon})$ using $O(n^{1+\epsilon})$ preprocessing time and $O(n)$ space.

2.1 Extended Visibility Graph

Consider a set $S = \{s_1, s_2, ..., s_n\}$ of n segments in the plane. The visibility graph $G = (V, E)$ is defined as a graph whose vertices are the set of the end points of the segments in S and there is an edge $v_i v_j \in E$ when v_i sees v_j. It is easy to show that the number of edges of G is $O(n^2)$. Initial algorithms for computing the visibility graph were proposed by Welzl [5] and Asano *et al.* [6] with time complexity of $O(n^2)$. Later, Ghosh and Mount [7] developed an optimal output sensitive algorithm that computes the visibility graph in $O(E + n \log n)$ time. Finally, Overmars and Welzl [8] presented a suboptimal but practical algorithm that computes the visibility graph in $O(|E| \log n)$ time.

The extended visibility graph [1] is defined over the visibility graph by extending each edge $v_i v_j \in E$ at both ends until it intersects a segment in S. Assume

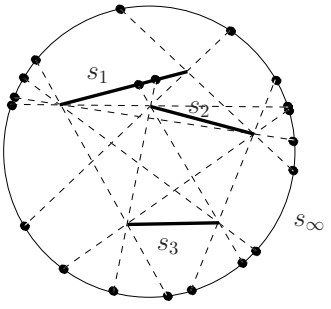

Fig. 1. The dashed segments are the edges of the extended visibility graph of $S = \{s_1, s_2, s_3, s_\infty\}$

that s_l and s_m are the first segments intersected by v_iv_j when it is extended from its endpoints. If p and q are these intersection points then they are two vertices of the extended visibility graph and pq is an edge of this graph. In cases that there is no intersection, a segment s_∞ is assumed at infinity that is intersected by all extended edges. Therefore each extended edge of G intersects two segments in the set $S \cup s_\infty$ and itself is a segment. The set of these extended edges compose an arrangement of $O(n^2)$ possibly intersecting segments in the plane. Fig. 1 shows a sample extended visibility graph.

Suri and O'Rourke [1] used a modified version of Welzl's algorithm [5] and compute the extended visibility graph in $O(n^2)$ time. Keil *et al.* [9] presented a method that for any edge of the visibility graph, its corresponding edge in the extended visibility graph can be computed in constant time. Combining this method and the algorithm of Ghosh and Mount [7], the extended visibility graph can be computed in $O(E + n \log n)$ time.

2.2 Point Location

As a subproblem in our methods, we need to solve a special case of the point location problem. The general point location problem is to preprocess a planar subdivision $\mathcal{S}$ with n edges, so that we can quickly find the face f of $\mathcal{S}$ that contains a query point q. This problem can be solved in $O(\log n)$ query time using $O(n \log n)$ and $O(n)$ preprocessing time and space, respectively [10]. But, the point location problem that we need to answer is to check whether a query point q lies inside a give star-shaped simple polygon P.

We can solve this version of point location problem more efficiently than the general case, when we know a kernel point of P. Recall that a polygon P is star-shaped when there is a point p inside it such that for any other point p' inside P, the segment pp' lies completely inside P. If so, p is said to be a kernel point of P. Assume that p is a kernel point of P and $v_1v_2...v_n$ are the vertices of P in counterclockwise order such that pv_1 has the least angle with the x-axis.

Having this ordered list of vertices $v_1v_2...v_n$, we can locate position of a query point q in this list in $O(\log n)$ time by a classical binary search. Assume that

q lies between v_k and v_{k+1}. Then, we must only check whether the segment pq intersects $v_k v_{k+1}$ or not which can be performed in constant time. Therefore, we can answer the point location query in $O(\log n)$ time only by having a kernel point and the ordered list of the vertices of P. Trivially, p is a kernel point of $VP(p)$ and we can use this method for point location on these polygons.

2.3 Ray Shooting

In a planar scene, the ray shooting problem is to find the first segment intersected by a ray from a given point toward a given direction. We examine this problem when the scene is a simple polygon and when the scene is a planar arrangements of segments.

Ray shooting in a simple polygon. The problem of shooting a ray in a simple polygon was first addressed by Chazelle and Guibas [11]. They showed that it can be answered in $O(\log n)$ time using $O(n)$ preprocessing time and space. Then, simpler methods were presented by Chazelle *et al.* [12] and Hershberger and Suri [13]. The method of Hershberger and Suri is based on finding a Steiner triangulation of the polygon. In this triangulation, any ray intersect at most $O(\log n)$ triangles and by tracing the set of the intersected triangles, we can find the first intersection point of the ray and the polygon boundary.

Ray shooting in a planar subdivision. There are many approaches for solving the ray shooting problem in a planar subdivision. This problem can be solved using half-plane range searching data structures to be discussed later. Using this approach, Agarwal and Erickson [14] have shown that this problem can be solved in $O(n^{1/2+\epsilon})$ query time using $O(n \log^3 n)$ preprocessing time and space, or it can be solved in $O(log^3 n)$query time using $O(n^{2+\epsilon})$ preprocessing time and space.

Another method with near linear space requirement, is the ray shooting algorithm introduced by Cheng and Janardan [15]. They showed that ray shooting in an arrangement of n non-intersecting segments can be answered in $O(\sqrt{n} \log n)$ by spending $O(n \log^2 n)$ space and $O(n\sqrt{n} \log^\omega n)$ preprocessing time, where ω is a constant less than 4.3. In the case of possibly intersecting segments, the space increases to $O(n \log^3 n)$.

2.4 Range Searching

In range searching problems, there is a set of n points in d-dimensional space and we want to report (or count) the points lying in a region R in this space. In this paper, we need to solve this problem when P is a set of points in the plane and R is a half-plane or a triangle.

The first near optimal query time using linear preprocessing time and space was achieved by Welzl [16]. He used the idea of spanning tree with low crossing numbers and answered the queries in time close to $O(\sqrt{n})$. Matoušek and Welzl [17] developed a method that solve half-plane range queries in $O(\sqrt{n} \log n)$

time using $O(n \log n)$ preprocessing time and space. Chazelle *et al.* [2] introduced a simplex range searching method, called CSW, for any dimension d that answer queries in $O(n^{1-1/d+\epsilon})$ by using $O(n^{1+\epsilon})$ preprocessing time and $O(n)$ space, for any arbitrary small positive constant ϵ. They also allow a tradeoff between storage and query time, so if one can spend storage of size $O(m)$, where $n \leq m \leq n^d$, the preprocessing can be done on the set of points in time $O(m^{1+\epsilon})$, so that the query can be answered in $O(\frac{n^{1+\epsilon}}{m^{1/d}})$. This solution comes close to the lower bound, up to a factor of n^{ϵ}. Since we use this approach for our problem(finding the set of segments intersected by two given segments), we give an overview of this method.

We briefly describe the CSW method just for 2 dimension. For a point set S of n points, a family $\mathcal{F} = \{\Xi_1, ..., \Xi_k\}$ of triangulations of the plane is constructed such that the size of any one of these triangulations is $O(r^2)$ for some constant r. This family of triangulations has this property that for any line l, there is at least one triangulation Ξ_i that only $O(n/r)$ of the set of n points lie inside the triangles of Ξ_i that are intersected by l. We denote this triangulation associated for a line l by T_l. This process is continued recursively for each triangle of these family of triangulations that contains more that i pints for some constant value of i. Inside the leaf nodes of this tree the search is done in a standard partitioning scheme.

Assume that we want to search for points lie inside a half-plane H that is above a line l. We first find T_l from the above range searching data structure. The points lie inside the triangles of T_l which are above (below) l are (are not) inside the half-plane and we must recursively continue the search only over the triangles of T_l intersected by l. However these triangles only contains $O(n/r)$ of the points.

The size of this data structure is $O(n)$ and can be constructed in $O(n^{1+\epsilon})$ time. Using this data structure, the half-plane range searching (counting) can be answered in time $O(n^{\frac{1}{2}+\epsilon})$.

The above data structure, or generally any other partition tree gives the result of range searching as the disjoint union of some canonical subsets. As previously has been used, e.g. by Dobkin and Edelsbrunner [18], these canonical subsets can further be preprocessed, so that a conjunction of range searchings can be answered on the point set. It can be shown that using these data structures in a multilevel fashion does not increase the amount of needed space, but increase the amount of query time by a poly-logarithmic factor, however since the query time has a factor n^{ϵ}, this factor can be neglected. Therefore we can use the data structure of Chazelle *et al.* recursively to answer a conjunction of range searchings, without any space/time overhead.

2.5 Common Intersecting Segment Detection

A new problem we encountered while trying to solve the weak-visibility problem is to determine if any line segment e_i from a set $E = \{e_1, e_2, ..., e_n\}$ of n segments in the plane is intersected by two given segments s and t. We refer to this problem as CISD for Common Intersecting Segment Detection.

In order to solve CISD we use the technique described above for half-plane and triangle range searching. If a segment e_i with x_i and x'_i as end points intersects segment s, then x_i and x'_i are in opposite sides of the supporting line of s, denoted by l_s. The same statement is true for the endpoints of s, v_s and v'_s, and the supporting line of e_i, l_{e_i}. When e_i intersects both s and t, x_i and x'_i lies in certain positions. As can be seen in Fig. 2, l_s and l_t divide the plane into four regions. Let call them by the position of them relative to l_s and l_t. If the two segments are intersecting, one of x_i and x'_i lies in UU and the other one in LL, or one in LU and the other one in UL. But if the two segments do not intersect, only one of these situations are possible. Therefore in order to detect whether e_i intersects both s and t, we should check the positions of x_i and x'_i in that four regions, and the positions of the end points of s and t in the two regions separated by l_{e_i}.

Thus, the CISD problem can be solved in this way: Consider s and t intersect each other. We first find all segments in E that have one end point in UU and the other one in LL. We also find all segments that have one end point in LU and the other one in UL. These segments are the set of segments that intersects l_s and l_t. The same thing should be done for end points of s and t. We dualize s and t and also l_{e_i} for each e_i in the resulting set of segments. In the dual plane, each l_{e_i} is mapped to a point and segments s and t are mapped to two double wedges. In this plane, we should search for points lie in the intersection of two double wedges corresponding to s and t.

The procedure for non-intersecting query segments is similar, except that in some cases we should search in only one pair of regions (UU, LL) and (LU, UL). If both s and t lies in one side of the vertical line drawn from the intersection of l_s and l_t, then we must only search intersecting segments that has an end point in UU and the other in LL. If both s and t lies in different sides of this vertical line we should search in regions LU and UL. Otherwise, we should search in both pairs of regions.

Therefore we solve the problem by combining a series of half-plane and triangle range searchings. We define three range searching problems $\mathcal{P}_i$ for $1 \leq i \leq 3$ with relations $\Diamond^i$ as below [14]:

- $e \Diamond^1 \mathcal{H}$: The left endpoint of segment e lies in half-plane $\mathcal{H}$.
- $e \Diamond^2 \mathcal{H}$: The right endpoint of segment e lies in half-plane $\mathcal{H}$.
- $e \Diamond^3 \gamma$: The supporting line of segment e intersects segment γ; or equivalently, in the dual plane, the point dual to l_e lies in the double wedge dual to γ.

By combining this searching problems, we can solve the CISD problem. Let problems $\mathcal{P}_i$, $1 \leq i \leq 3$, use the relation $\Diamond^i$. In order to solve we must solve four subproblems:

- 1(2): Find segments whose left end points lie above(below) both lines l_s and l_t and right end points lie below(above) both lines l_s and l_t and also the supporting line of them intersects both segments s and t.
- 3(4): Find segments whose left end points lie above(below) l_s and below (above) l_t and right end points lie below(above) l_s and above(below) l_t and also the supporting line of them intersects both s and t.

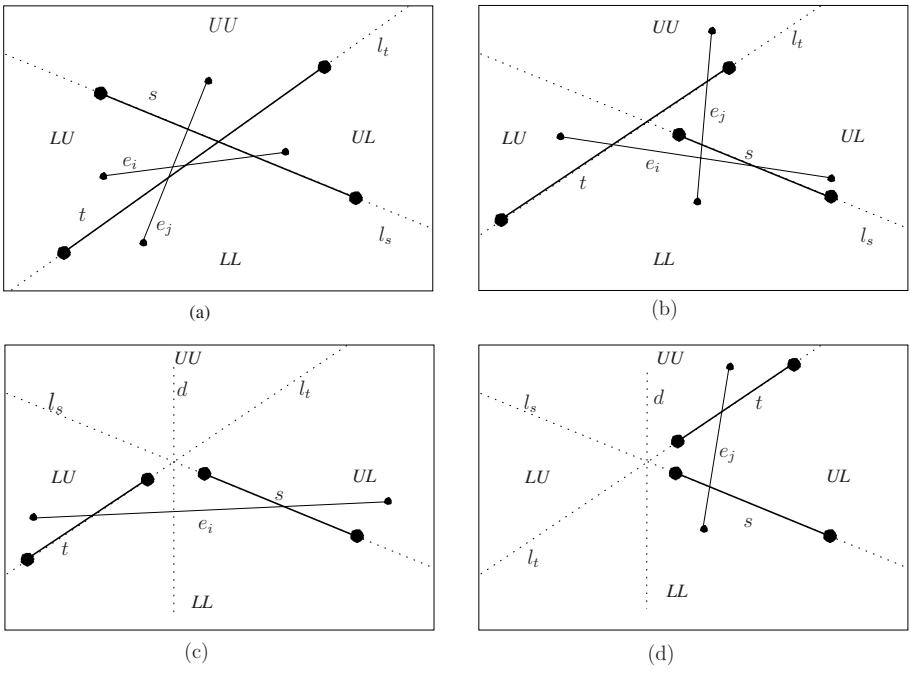

Fig. 2. Different cases arise when a segment intersects s and t

In the first subproblem we should find segments whose left end points lie above l_s and l_t. It can be done by using $\mathcal{P}_1$ two times, the first time with the half-plane above l_s and the second time with the half-plane above l_t. In the result set, we should select those segments, whose left end point lie below l_s and l_t. This can be achieved by using $\mathcal{P}_2$ two times, with the half-planes below l_s and l_t, respectively. Then, we select those segments of the result that intersect both l_s and l_t, and it can be done by using $\mathcal{P}_3$ two times with s and t, respectively. Therefore, by joining problems $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$, we can solve subproblem 1. As discussed in Section 2.4 for multi-level searches, and discussions in [14], we conclude that subproblem 1 can be solved in $O(\frac{n^{1+\epsilon}}{m^{1/d}})$ time using $O(m)$ space and $O(m^{1+\epsilon})$ preprocessing time, where $n^2 \geq m \geq n$. As commented, the n^ϵ factor can also be replaced by a poly-logarithmic factor. Similarly, other subproblems (2-4) can be solved with the same time and space complexities.

This discussion leads to the following lemma about solving the CISD problem:

Lemma 1. *We can preprocess a scene of n segments in time $O(n^{1+\epsilon})$ and build a data structure of size $O(n)$ such that for any given pair of segments s and t, we can determine whether both segments intersect a segment of the scene in $O(n^{1/2+\epsilon})$ query time.*

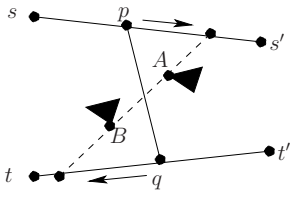

Fig. 3. Weak-visibility between two segments

In our main weak-visibility problem, we need to solve the CISD problem on the edges of the extended visibility graph of a scene of n segments. Since the size of the extended visibility graph is $O(n^2)$ we can conclude the following theorem:

Theorem 1. *The extended visibility graph of a scene of n segments can be preprocessed in $O(n^{2+\epsilon})$ time and $O(n^2)$ space such that for any given pair of segments s and t, we can determine whether there is a segment in this graph that intersects both segments in $O(n^{1+\epsilon})$ query time.*

3 Weak-Visibility Properties

In this section we illustrate two properties that facilitate detection of the weak-visibility between two objects s and t in a planar polygonal scene. The first property is about the visibility of two segments and the second property is about the visibility between two convex polygons.

Lemma 2. *In a planar polygonal scene, two segments ss' and tt' are weakly visible from each other if and only if one endpoint of ss' or tt' sees a point on the other segment or there is at least one edge in the extended visibility graph of the scene which is intersected by both ss' and tt' segments.*

Proof. Proof of the *if* part: Trivially, when an endpoint of one segment sees the other segment, the segments are weakly visible. Moreover, the edges of the extended visibility graph do not intersect the scene objects. So, if both segments intersect an edge of this graph they can weakly see each other along this edge and therefore they are weakly visible.

Proof of the *only if* part: Assume that the segments are weakly visible. Then, there are middle points p and q on ss' and tt', respectively, which are visible from each other. As shown in Fig. 3, we move the endpoints of pq along ss' and tt' in opposite directions. We do this process in a manner such that pq does not intersect edges of the scene. Continuing this movement, either p or q reaches the corresponding endpoint of ss' or tt', or pq touches two vertices A and B of the scene from its opposite sides. The former means that one endpoint of ss' or tt' sees the other segment. In the latter, both ss' and tt' intersect the edge of the extended visibility graph drawn from A and B.

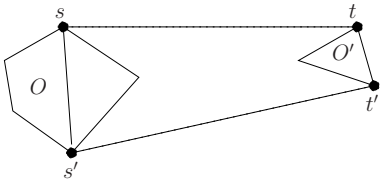

Fig. 4. Weak-visibility between two objects

Using the above lemma, we can determine the weak visibility of two segments by reducing it to range search and point-segment visibility problems to be discussed in the next section.

The next lemma is about the weak visibility of two convex polygons. Assume that O and O' are two disjoint convex objects and $CH_{OO'}$ is the convex hull of these polygons. Some of the segments of O and O' lie on the boundary of $CH_{OO'}$ and other segments lie inside this convex hull. Assume that $O_{ss'}$ and $O'_{tt'}$ are respectively those segments of O and O' that lie inside $CH_{OO'}$ (See Fig. 4).

Lemma 3. *In a planar polygonal scene, the objects O and O' are weakly visible if and only if a segments of $O_{ss'}$ is weakly visible from a segment of $O'_{tt'}$.*

Proof. Proof of the *if* part: Trivially, the segments belong to their corresponding objects and if an edge of O sees an edge of O', the objects are also weak-visible.

Proof of the *only if* part: Trivially, when the objects are weakly visible, a point p from O sees a point q of O'. The segment pq intersects an edge of $O_{ss'}$ and an edge of $O'_{tt'}$ and therefore these intersected edges are also weakly visible.

According to this lemma, to determine the weak visibility of two convex objects, it is enough to check this problem for any pair of their segments(one from each object).

We use these lemmas in the following section to determine whether two segments or two convex polygons are weakly visible.

4 Visibility Detection Methods

Now, we are ready to discuss our method of detecting weak-visibility between two objects. The objects we consider in this paper can be points, line segments or convex polygons. To simplify our analysis, we assume that convex objects have constant complexities, i.e they have at most c vertices for some constant value of c.

We consider two versions of this problem: In the first version, one of the objects is known in advance and we can do some preprocesses on it and the other object is given in query time. In the other version, both of the objects are given as query. We refer to the first problem by SOQ(Single Object Query) and the socond by TOQ(Two Objects Query).

4.1 Visibility Detection in SOQ

Assume that object s is known in advance and t is given in query time. If s is a point, $VP(s)$ can be obtained in $O(n \log n)$ [1] in the preprocessing phase. In query time, the query object, t, is tested against $VP(s)$. If t intersects $VP(s)$ then s and t are weakly visible. Whereas $VP(s)$ is a simple polygon of size $O(n)$, we can build a point location structure on it in linear time (as discussed in Section 2.2) by which any point location query is answered in $O(\log n)$ time. As discussed in Section 2.3, we can also build a ray shooting structure on it in $O(n)$ time and space by which any ray shooting query is answered in $O(\log n)$ time. Therefore, if we preprocess $VP(s)$ for point location and ray shooting queries, we can find the answer in $O(\log n)$ time when t is a point. If t is a segment, we first locate one endpoint of it and do a ray shooting towards the other endpoint and check the result of this ray shooting. If the intersection point of the ray shooting problem(if any) lies on the segment t, it means that t intersects $VP(s)$. Finally, if t is a convex object, it is enough to only check its edges and while it has a constant number of edges, we can check the intersection between t and $VP(s)$ in $O(\log n)$ query time. So, we can say the following result about this case of the problem:

Corollary 1. *In a planar polygonal scene, the visibility between a query object t and a given point s can be answered in $O(\log n)$ time using $O(n \log n)$ and $O(n)$ preprocessing time and space, respectively.*

When s is a line segment, we can use the same method as the one discussed above. However, for a line segment s, $VP(s)$ can be of size $O(n^4)$ and is obtained in $O(n^4)$ time [1]. Moreover, $VP(s)$ is not a simple polygon and it is a polygon with holes. For this kind of $VP(s)$, to answer the point location queries in $O(\log n)$ time, a point location data structure of size $O(n^4)$ is required [10]. Unfortunately, preparing the corresponding ray shooting data structure requires $O(n^6 \log^{4.3} n)$ and $O(n^4 \log^2 n)$ preprocessing time and space by which the ray shooting problems can be answered in $O(n^2 \log n)$ time [15]. Although, we can reduce the preprocessing cost by considering $VP(s)$ as a set of overlapping triangles, but, the query time will be still high. According to Lemma 3, the same result is obtained when s is a convex polygon.

In the next section we present a method with less preprocessing cost and better query time when both objects are given in query time. Also, we can use this method in SOQ problems in which one of the query objects is known in advance.

4.2 Visibility Detection in TOQ

In TOQ version of the problem, both objects s and t are given in query time and it is not possible to preprocess based on them in advance. However we can preprocess the underlying scene to facilitate answering the TOQ problems. We present the visibility detection methods in four subversions of the problem: both object are points, only one of the objects is a point, both objects are line segments, and one or both objects are convex polygons.

If both s and t are points, we can preprocess the scene for ray shooting to answer the problem efficiently. Whereas the scene is a polygonal scene, its ray shooting data structure requires $O(n\sqrt{n}\log^{4.3} n)$ time and $O(n\log^2 n)$ space and the ray shooting queries can be answered in $O(\sqrt{n}\log n)$ time (Section 2.3). Having this data structure, we shoot a ray from s towards t and if the first intersection point lies between s and t they are not visible from each other and otherwise they are visible from each other. So, we can say,

Corollary 2. *In a planar polygonal scene, the visibility between two query points can be answered in $O(\sqrt{n}\log n)$ time using $O(n\sqrt{n}\log^{4.3} n)$ and $O(n\log^2 n)$ preprocessing time and space, respectively.*

If one of the objects is a point we do not preprocess the scene. If s is the point, $VP(s)$ is found in $O(n\log n)$ time. While $VP(s)$ is a star-shaped simple polygon we can test whether it is intersected by t in $O(n)$ time. This can be done when t is a line segment or it is a convex polygon of constant complexity. Therefore,

Corollary 3. *In a planar polygonal scene, the visibility between a query point and a query line segment or convex polygon can be answered in $O(n\log n)$ time.*

Now, we return to our main problem: assume that both s and t are line segments and both are given in query time and we need to decide whether they are weakly visible. To solve this problem we use the result of Lemma 2 and Theorem 1.

Theorem 2. *A planar polygonal scene of total complexity of n can be preprocessed in $O(n^{2+\epsilon})$ time to build data structures of $O(n^2)$ total size, so that the weak-visibility between two query line segments can be determined in $O(n^{1+\epsilon})$ time.*

Proof. According to Lemma 2, we must check two cases to decide about the weak-visibility between two segments: an endpoint of one segment weakly sees the other segment or both segments intersect some edges of the extended visibility graph of the scene. According to Corollary 3, the first case can be checked in $O(n\log n)$ time without any preprocessing cost. To check the other one, the extended visibility graph of the scene can be constructed in $O(n^2)$ in the preprocessing phase as described in Section 2. This extended visibility graph is preprocessed for range searching according to the result of Theorem 1. There are $O(n^2)$ segments in the extended visibility graph. So, the range searching structure of size $O(n^2)$ can be constructed in $O(n^{2+\epsilon})$ preprocessing time. According to Theorem 1, this range searching structure enables us to check if two query segments intersect some edges of the extended visibility graph in $O(n^{1+\epsilon})$ time. Clearly, the preprocessing cost and the query time of this argument follow the theorem.

We can extend our result of two line segments to solve the problem for two convex polygons. According to Lemma 3, to determine the weak visibility between two convex polygons, it is enough to decide about the weak-visibility of any pair of their edges. If we assume that the complexity of the objects is constant, the above

argument along with the result of Theorem 2 leads to the following theorem about determining weak visibility of two convex object in a planar polygonal scene:

Theorem 3. *A planar polygonal scene of total complexity of n can be preprocessed in $O(n^{2+\epsilon})$ time to build data structures of $O(n^2)$ total size so that the weak-visibility problem for two query convex polygons with constant complexity can be determined in time $O(n^{1+\epsilon})$.*

5 Conclusion

Despite the extensive research and results on visibility problems, there are still many open problems in this area. Many practical applications of these problems motivate researchers to optimize solutions of these problems and make them more practical. Here, we focus on determining weak visibility between two objects in a planar environment. We use the extended visibility graph and build a multi-level range searching structure to facilitate answering our problem.

In this problem, the preprocessing data structures are used to efficiently decide whether two query objects are weakly visible. Our method uses $O(n^{2+\epsilon})$ preprocessing time to build a data structure of size $O(n^2)$ which enables us to answer the queries in $O(n^{1+\epsilon})$ query time. It is notable that this problem is 3SUM-hard and the lower bound of its solutions is $\Omega(n^2)$.

Although the off-line version of the problem has been solved optimally, but to our best knowledge, this is the first attempt to solve this problem in the query version. This work can be extended in several directions: we can extend this method to other types of objects, for example, to concave objects. The method can also be extended for upper dimensions as well as to cover dynamic environments.

References

1. Suri, S., O'Rourke, J.: Worst-case optimal algorithms for constructing visibility polygons with holes. In: Symposium on Computational Geometry, pp. 14–23 (1986)
2. Chazelle, B., Sharir, M., Welzl, E.: Quasi-optimal upper bounds for simplex range searching and new zone theorems. Algorithmica 8(5&6), 407–429 (1992)
3. Gajentaan, A., Overmars, M.H.: On a class of o(n2) problems in computational geometry. Comput. Geom. 5, 165–185 (1995)
4. Wismath, S.K.: Computing the full visibility graph of a set of line segments. Inf. Process. Lett. 42(5), 257–261 (1992)
5. Welzl, E.: Constructing the visibility graph for n-line segments in o(n) time. Inf. Process. Lett. 20(4), 167–171 (1985)
6. Asano, T., Asano, T., Guibas, L.J., Hershberger, J., Imai, H.: Visibility of disjoint polygons. Algorithmica 1(1), 49–63 (1986)
7. Ghosh, S.K., Mount, D.M.: An output sensitive algorithm for computing visibility graphs. In: FOCS, pp. 11–19. IEEE Computer Society Press, Los Alamitos (1987)
8. Overmars, M.H., Welzl, E.: New methods for computing visibility graphs. In: Symposium on Computational Geometry, pp. 164–171 (1988)

9. Keil, M., Mount, D.M., Wismath, S.K.: Visibility stabs and depth-first spiralling on line segments in output sensitive time. Int. J. Comp. Geom. Appl. 10(5), 535–552 (2000)
10. Lee, D., Preparata, F.: Location of a point in a planar subdivision and its applications. SIAM Journal of Computing 6(3), 594–606 (1977)
11. Chazelle, B., Guibas, L.J.: Visibility and intersection problems in plane geometry. Discrete & Computational Geometry 4, 551–581 (1989)
12. Chazelle, B., Edelsbrunner, H., Grigni, M., Guibas, L.J., Hershberger, J., Sharir, M., Snoeyink, J.: Ray shooting in polygons using geodesic triangulations. Algorithmica 12(1), 54–68 (1994)
13. Hershberger, J., Suri, S.: A pedestrian approach to ray shooting: Shoot a ray, take a walk. J. Algorithms 18(3), 403–431 (1995)
14. Agarwal, P.K., Erickson, J.: Geometric range searching and its relatives. In: Chazelle, B., Goodman, J.E., Pollack, R. (eds.) Advances in Discrete and Computational Geometry, AMS Press, Providence, RI (1998)
15. Cheng, S.W., Janardan, R.: Space-efficient ray-shooting and intersection searching: algorithms, dynamization, and applications. In: SODA '91. Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms, pp. 7–16. ACM Press, New York (1991)
16. Welzl, E.: Partition trees for triangle counting and other range searching problems. In: Symposium on Computational Geometry, pp. 23–33 (1988)
17. Matousek, J., Welzl, E.: Good splitters for counting points in triangles. J. Algorithms 13(2), 307–319 (1992)
18. Dobkin, D.P., Edelsbrunner, H.: Space searching for intersecting objects. J. Algorithms 8(3), 348–361 (1987)
19. Matousek, J.: Efficient partition trees. Disc. & Comp. Geom. 8, 315–334 (1992)