Efficient Visibility Maintenance of a Moving Segment Observer inside a Simple Polygon

Amir Ali Khosravi*† Alireza Zarei*† Mohammad Ghodsi*†

Abstract

In this paper we consider maintaining the visibility of a segment observer moving inside a simple polygon. A practical instance of this problem is to identify the regions of a planar scene illuminated by a fluorescent lamp while the lamp moves around. We consider both strong and weak visibility in this paper. Our method is based on the shortest path tree which builds a linear-sized data structure in $O(n)$ time, where n is the number of the vertices of the underlying simple polygon $\mathcal{P}$. We first compute $VP(\overline{st})$, the initial view of the segment observer $\overline{st}$. Then, as $\overline{st}$ moves, each change of $VP(\overline{st})$ can be computed in $O(\log^2(|VP(\overline{st})|))$ time when the observer is allowed to change its direction, and in $O(\log(|VP(\overline{st})|))$ time when the observer moves along a given line.

1 Introduction

Two points p and q of a planar scene are visible to or see each other if pq does not intersect edges of the scene. The visible region or the visibility polygon of a point p of a scene $\mathcal{P}$ is the set of points in $\mathcal{P}$ which are visible from p and is denoted by $VP(p)$. Two versions of the visibility polygon are defined for a line segment $\overline{st}$ inside a planar scene $\mathcal{P}$. The set of points in $\mathcal{P}$ which are visible from all points of $\overline{st}$ is called its *strong visibility polygon* and is denoted by $SVP(\overline{st})$. The set of points in $\mathcal{P}$ which are visible from at least one point of $\overline{st}$ is called its *weak visibility polygon* and is denoted by $WVP(\overline{st})$.

There are optimal algorithms that find $VP(p)$, $SVP(\overline{st})$, and $WVP(\overline{st})$ of a point p or a line segment $\overline{st}$ in a static planar scene $\mathcal{P}$. When $\mathcal{P}$ is a simple polygon of n vertices, $VP(p)$, $WVP(\overline{st})$, and $SVP(\overline{st})$ can be found in $O(n)$ time [2, 3, 4, 5]. When $\mathcal{P}$ is a polygonal domain of n total vertices, $VP(p)$ can be found in $O(n \log n)$ time [8] and $WVP(\overline{st})$ can be found in $O(n^4)$ time [6].

Also, there are several efficient algorithms for maintaining the visibility of a moving point p in planar scenes. Almost all of these methods have a preprocessing phase to build data structures about the visibility coherence of the scene. These data structures are then used to apply the changes of VP efficiently as the point observer moves. There is a trade-off between the preprocessing time and space and the update time; the higher preprocessing cost, the lower update time. In a simple polygon $\mathcal{P}$, changes of $VP(p)$ can be handled in $O(\log^2 |VP(p)|)$ time using $O(n \log n)$ time to preprocess $\mathcal{P}$ and build a data structure of size $O(n)$ [1]. However, updates can be handled in $O(\log n)$ optimal time if we spend $O(n^3 \log n)$ and $O(n^3)$ preprocessing time and space, respectively [7].

The above methods which have been presented for a moving point can be extended and adjusted naively to maintain $WVP(\overline{st})$ or $SVP(\overline{st})$ for a moving segment $\overline{st}$. But, to the best of our knowledge, there is no result on applying and analyzing these approaches or on directly solving this problem for a moving line segment. In this paper, we consider the problem of maintaining $WVP(\overline{st})$ and $SVP(\overline{st})$ of a segment $\overline{st}$ inside a simple polygon. To solve this problem, we use the shortest path tree as used by Aronov *et al.* [1]. Our approach leads to a method with optimal $O(n \log n)$ and $O(n)$ preprocessing time and space, respectively. Then, strong or weak visibility change events are handled in $O(\log^2 n)$ time if the observer is allowed to change its movement direction and in $O(\log n)$, if the observer moves along a given line.

For simplicity purposes, we assume that points of the scene are in general position (no three points are collinear), and the observer position can be computed by a fixed degree algebraic function of time.

2 Segment Visibility in Simple Polygons

In a simple polygon $\mathcal{P}$, a point p may see the whole, a portion, or nothing of a segment $\overline{st}$. These situations have been shown in Figure 1. The visible portion can be obtained by drawing the shortest paths from p to s and t. The region contained between the initial segments of these two shortest paths is called the visible cone of p with respect to $\overline{st}$ and is denoted by $VC(p, \overline{st})$. According to this figure, p is strongly visible to $\overline{st}$ if $\overline{st}$ completely lies inside $VC(p, \overline{st})$ (Part A of Figure 1), p is weakly visible to $\overline{st}$ if $\overline{st}$ intersects boundary of

*Computer Engineering Department, Sharif University of Technology, IPM School of Computer Science, a_khosravi@ce.sharif.edu, zarei@mehr.sharif.edu, ghodsi@sharif.edu

†This work was partially supported by IPM school of computer science (contract: CS1385-2-01) and Iran Telecommunication Research Center(ITRC)

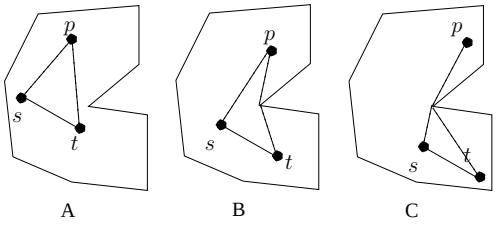

Figure 1: Point-segment visibility.

$VC(p, \overline{st})$ (Part B of Figure 1), and otherwise, p is not visible to $\overline{st}$ (Part C of Figure 1).

Therefore, having the shortest paths from both endpoints of a segment $\overline{st}$ to all vertices of a simple polygon $\mathcal{P}$, we can determine the set of strongly visible and weakly visible vertices from $\overline{st}$ from which $SVP(\overline{st})$ and $WVP(\overline{st})$ can be computed.

It is notable that in a simple polygon, none or a single connected portion of each edge is weakly (strongly) visible from a segment. As described by Guibas *et al.* [3], the vertices of $WVP(\overline{st})$ are of three types:

- Type A: Vertices of $\mathcal{P}$ which are weakly visible from s or t,
- Type B: Vertices produced by drawing a ray from an endpoint x of $\overline{st}$ that passes through a visible vertex v of $\mathcal{P}$ and intersects an edge e (denoted by $V(\overrightarrow{xv}, e)$), and
- Type C: Vertices produced by drawing a ray from some interior point of $\overline{st}$ that passes through two vertices u and v of $\mathcal{P}$ and intersects an edge e (denoted by $V(\overrightarrow{uv}, e)$).

Part (I) of Figure 2, shows these three types of vertices that may appear in $WVP(\overline{st})$. Vertices of $SVP(\overline{st})$ are also of three types as shown in part (II) of Figure 2:

- Type A: Vertices of $\mathcal{P}$ which are strongly visible from s and t,
- Type B: Vertices produced by drawing a ray from an endpoint x of $\overline{st}$ that passes through a strongly visible vertex v of $\mathcal{P}$ and intersects an edge e (denoted by $V(\overrightarrow{xv}, e)$)), and
- Type C: Vertices produced from the intersection of two rays drawn from s and t that pass through vertices v and u of $\mathcal{P}$, respectively (denoted by $V(\overrightarrow{sv}, \overrightarrow{tu})$).

We can build the shortest path trees $SPT(s)$ and $SPT(t)$, from endpoints of $\overline{st}$ in linear time and by a linear trace over these trees, $WVP(\overline{st})$ can be obtained in $O(n)$ time [3]. Also, we can obtain $SVP(\overline{st})$ in linear time in a similar method which we omit the details here.

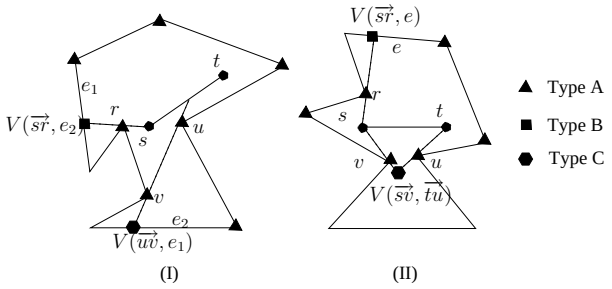

Figure 2: Vertex types of SVP and WVP.

Now, assume that $\overline{st}$ moves inside $\mathcal{P}$ along a given line. In order to maintain and update $WVP(\overline{st})$ and $SVP(\overline{st})$, we can monitor changes of $SPT(s)$ and $SPT(t)$ during this movement. When a point moves along a line segment inside a simple polygon, the shortest path tree from it to vertices of polygon does not change very much. This idea is used in [9] to compute the shortest path trees of all of the vertices in a simple polygon, and visibility graph respectively. Whenever one of the $SPT(s)$ or $SPT(t)$ is changed combinatorially, $WVP(\overline{st})$ and $SVP(\overline{st})$ must be updated accordingly. Aronov *et al.* [1] proposed a method for predicting and handling changes of $SPT(q)$ for a moving point q in a simple polygon. In their method, each change of $SPT(q)$ is handled in $O(\log(|VP(q)|))$ time when q moves along a single line and in $O(\log^2(|V(q)|))$ when q is permitted to change its motion direction. So, we can use this method for handling changes of $SPT(s)$ and $SPT(t)$. Instead of building WVP and SVP from scratch, we like to apply only the necessary changes to $WVP(\overline{st})$ and $SVP(\overline{st})$, preferably, in logarithmic time.

During the motion, $WVP(\overline{st})$ and $SVP(\overline{st})$ are changed continuously while $SPT(s)$ and $SPT(t)$ are changed in discrete time-stamps. We overcome this problem by maintaining only the combinatorial structure of a visibility polygon. We define the combinatorial structure of a visibility polygon as the sequence of the vertices of this polygon. Some of these vertices are the vertices of $\mathcal{P}$ which their exact positions are fixed. Others lie on the edges of or inside $\mathcal{P}$ and belong to some vertices of $\mathcal{P}$. Examples of such vertices are Types B and C vertices of strong and weak visibility polygon of a segment as discussed before. We do not maintain the exact position of these vertices during the motion. However, having this sequence of visible vertices, we can construct the exact visibility polygon by a linear trace over it.

Lemma 1 *Using the sequence of weakly visible vertices of segment $\overline{st}$, we can find $WVP(\overline{st})$ in $O(|WVP(\overline{st})|)$.*

Proof. According to our definition of the sequence of visible vertices, we only need to compute the exact position of type B and C vertices in this sequence. For a

type B vertex $V(\overrightarrow{xv}, e)$ (a type C vertex $V(\overrightarrow{uv}, e)$), we have the two lines defining this vertex which are the supporting lines of xv (uv) and edge e. So, this vertex can be computed in constant time. Therefore, the exact position of all vertices of $WVP(\overline{st})$ can be computed by a linear trace over the sequence of its vertices. □

Lemma 2 *Using the sequence of strongly visible vertices of segment $\overline{st}$, we can find $SVP(\overline{st})$ in $O(|SVP(\overline{st})|)$.*

Proof. We only need to find the exact position of the type B and C vertices. For a type B vertex $V(\overrightarrow{xv}, e)$ (a type C vertex $V(\overrightarrow{sv}, \overrightarrow{tu})$), we have the two lines defining this vertex which are the supporting lines of xv and edge e (the supporting lines of sv and tu). So, these vertices can be computed in constant time. Therefore, exact position of all vertices of $SVP(\overline{st})$ can be computed by a linear trace over the sequence of its vertices. □

Therefore, we use $WVP(SVP)$ for the sequence of weakly (strongly) visible vertices instead of the exact portion of visible edges.

In order to update $WVP(\overline{st})$ and $SVP(\overline{st})$ as $\overline{st}$ moves, we claim that it is enough to only track changes of $SPT(s)$ and $SPT(t)$.

Lemma 3 *For a moving line segment $\overline{st}$ in a simple polygon, $WVP(\overline{st})$ is changed only when $SPT(s)$ or $SPT(t)$ is changed.*

Proof. Trivially, the set of type A vertices of $WVP(\overline{st})$ is changed whenever the set of first level children of s or t in $SPT(s)$ or $SPT(t)$ is changed. Whenever a type B or a type C vertex appears in (disappears from) WVP, a vertex of $\mathcal{P}$ has been disappeared from (appeared in) $VP(s)$ or $VP(t)$. So, changes of these types of vertices happen only when $SPT(s)$ or $SPT(t)$ is changed. Therefore, $WVP(\overline{st})$ is changed only when $SPT(s)$ or $SPT(t)$ is changed. □

Lemma 4 *For a moving line segment $\overline{st}$ in a simple polygon, $SVP(\overline{st})$ is changed only when $SPT(s)$ or $SPT(t)$ is changed.*

Proof. The changes of $SVP(\overline{st})$ corresponds to changes of $SPT(s)$ and $SPT(t)$ by the same argument as the above lemma except for one situation: When two type B vertices are merged and produce one type C vertex as $\overline{st}$ moves, or when this scenario happens in reverse order: a type C vertex is splitted into two type B vertices. In such situations, $SPT(s)$ and $SPT(t)$ are not changed and we can not predict such events on $SVP(\overline{st})$ only by tracking events of $SPT(s)$ and $SPT(t)$. We solve this problem by doing a simple extra check when we want to obtain the exact strong visibility polygon of $\overline{st}$ based on the sequence of $SVP(\overline{st})$. Assume that two type B vertices $V(\overrightarrow{sv}, e)$ and $V(\overrightarrow{tu}, e)$ have been merged and built a single type C vertex $V(\overrightarrow{sv}, \overrightarrow{tu})$ during the motion, but we did not detect this change to update $SVP(\overline{st})$, accordingly. Then, if we are asked to draw the exact view of $\overline{st}$ in its current position, as we trace the sequence of $SVP(\overline{st})$, we detect that the exact position of the two consecutive vertices $V(\overrightarrow{sv}, e)$ and $V(\overrightarrow{tu}, e)$ on edge e do not obey their correct order in the sequence of $SVP(\overline{st})$. This is enough to alert us that we have already missed an event and instead of these two vertices there must be a single vertex $V(\overrightarrow{sv}, \overrightarrow{tu})$. So, whereas we may missed some changes of $SVP(\overline{st})$ by only tracking changes of $SPT(s)$ and $SPT(t)$, but, these missed events are detected whenever we compute the exact view without increasing the complexity of the drawing process. Therefore, we ignore such events and only handle the events happened to $SPT(s)$ and $SPT(t)$. □

So, as $\overline{st}$ moves, we only track the events related to changes of $SPT(s)$ and $SPT(t)$ and apply necessary changes to $WVP(\overline{st})$ and $SVP(\overline{st})$ whenever such an event happens. As discussed by Aronov *et al.*[1], two types of events may happen to $SPT(q)$ as a point q moves inside a simple polygon. The first event type, named first-level event, occurs when two consecutive children of the root of $SPT(q)$ become collinear. In this case, depending on the moving direction of q, one of these children will be a child of another one. The second kind of events happens whenever one of the children of the root of $SPT(q)$ and one of its children become collinear. The latter child is called the principal child of the former one. We refer to this type of events as principal-child event type. In this case, depending on the movement direction, the principal child becomes a child of the root of $SPT(q)$ next to its current parent.

We apply required changes to $SVP(\overline{st})$ for these two types of events as follows: Assume that a first-level event has occurred on $SPT(t)$. This means that two previously visible vertices u and v from t and t itself are collinear and as t continues its motion, v will not remain visible to t anymore. If v is not currently in $SVP(\overline{st})$ this event do not affect this sequence. Otherwise, v must be removed from $SVP(\overline{st})$. Also, the vertex $V(\overrightarrow{tu}, vv')$ (v' is the vertex next to v in $\mathcal{P}$) or $V(\overrightarrow{tu}, \overrightarrow{sv})$ which lies between v and u in $SVP(\overline{st})$ must be removed from this sequence. Moreover, based on the type of the vertex before v in $SVP(\overline{st})$ it must be updated as follows: 1) It is a type A vertex, named v''. In this case, v'' must lie before v on $\mathcal{P}$ and here a new vertex $V(\overrightarrow{tu}, v''v)$ is added to $SVP(\overline{st})$ before v in this sequence. 2) It is a type B vertex, named $V(\overrightarrow{tv}, e)$. In this case, this vertex is removed from $SVP(\overline{st})$ and a new vertex $V(\overrightarrow{tu}, e)$ is inserted in place of it. 3) It is a type C vertex, named $V(\overrightarrow{tv}, \overrightarrow{su'})$. In this case, it will be removed from $SVP(\overline{st})$ and a new vertex $V(\overrightarrow{tu}, \overrightarrow{su'})$ is inserted in place of it.

Now, assume that a principal-child event has occurred on $SPT(t)$, meaning that a previously non-visible vertex v and a visible vertex u from t and t itself are collinear. If v is not visible to s (it does not exist in $WVP(\overline{st})$), $SVP(\overline{st})$ is not changed. Otherwise, v must be inserted into $SVP(\overline{st})$ in its correct place and this sequence must be changed accordingly. In this case, it is simple to verify that u exists in $SVP(\overline{st})$ at the event time and there is either a type B vertex $V(\overrightarrow{tu}, e)$ or a type C vertex $V(\overrightarrow{tu}, \overrightarrow{su'})$ in $SVP(\overline{st})$. In both cases this vertex must be removed from $SVP(\overline{st})$ and, instead, for the former, a new vertex $V(\overrightarrow{tv}, e)$ and for the latter a new vertex $V(\overrightarrow{tv}, \overrightarrow{su'})$ must be inserted into $SVP(\overline{st})$ before v in this sequence. Also, based on the direction of the edge vv' of $\mathcal{P}$ (v' is the vertex next to v in $\mathcal{P}$) a new vertex $V(\overrightarrow{tu}, vv')$ or $V(\overrightarrow{tu}, \overrightarrow{sv})$ must be added to $SVP(\overline{st})$ next to v in this sequence.

It is simple to show that the above updates on $SVP(\overline{st})$ can be done in $O(\max(\log(|VP(s)|, \log(|VP(t)|, \log(|SVP(\overline{st})|))$ which is $O(\log n)$:

Lemma 5 *Any visibility change event on $SVP(\overline{st})$ can be computed in $O(\max(\log(|VP(s)|, \log(|VP(t)|, \log(|SVP(\overline{st})|))$ which is $O(\log n)$.*

Proof. As discussed above, to handle an event we need to search a vertex in $VP(s)$ or $VP(t)$ and find the position of the required changes in $SVP(\overline{st})$. Then, the required changes can be done in constant time. Therefore, any event can be applied to $SVP(\overline{st})$ in $O(\log(|VP(s)|+ \log(|VP(t)|) + \log(|SVP(\overline{st})|))$. Events of $SPT(s)$ and $SPT(t)$ can also be handled in $O(\log(|VP(s)|)$ and $O(\log(|VP(t)|))$, respectively. Hence, every event is handled in $O(\log n)$ time. □

The required changes to $WVP(\overline{st})$ are applied in a similar fashion. Assume that a first-level event has occurred on $SPT(t)$; two previously visible vertices u and v from t and t itself are collinear and as t continues its motion v would not be visible to t anymore. If v will remain visible from $\overline{st}$ it is sufficient to add vertex $V(\overrightarrow{uv}, e)$ to $WVP(\overline{st})$. Otherwise, v, $V(\overrightarrow{uv}, e)$ and $V(\overrightarrow{tv}, e)$ (if any) must be removed from $WVP(\overline{st})$. Moreover, if $WVP(\overline{st})$ currently contains vertex $V(\overrightarrow{tu}, e)$, it must be recomputed based on the new intersection point of the ray from t to u. Otherwise, $V(\overrightarrow{uv}, e)$ must be inserted in $WVP(\overline{st})$ sequence in place v.

To handle a principal-child event on $SPT(t)$, assume that a previously non-visible vertex v and a visible vertex u from t and t itself are collinear. In this case, v (if it does not currently exist in $WVP(\overline{st})$) and $V(\overrightarrow{tu}, e)$ and $V(\overrightarrow{uv}, e)$ vertices must be added to $WVP(\overline{st})$ in their correct positions. Otherwise, we must remove vertex $V(\overrightarrow{uv}, e)$ from $WVP(\overline{st})$ and instead, vertex $V(\overrightarrow{tv}, e)$(if v is a reflex vertex) must be inserted into $WVP(\overline{st})$.

Lemma 6 *Any visibility change event on $WVP(\overline{st})$ can be computed in $O(\log(|WVP(\overline{st})|))$ which is $O(\log n)$.*

Proof. As discussed above, to handle an event we need to search a vertex in $VP(s)$ or $VP(t)$ and find the position of the required changes in $WVP(\overline{st})$. Also, we need to check whether a vertex is visible from the segment $\overline{st}$. All of these searches can be done in logarithmic time. Having the result of these searches, the required changes can be done in constant time. Events of $SPT(s)$ and $SPT(t)$ can also be handled in $O(\log(|VP(s)|)$ and $O(\log(|VP(t)|))$, respectively. Hence, every event is handled in $O(\log n)$ time. □

Summarizing the above discussions and results, we have the following theorem:

Theorem 7 *A simple polygon $\mathcal{P}$ of n vertices and a given segment $\overline{st}$ inside it can be preprocessed in $O(n \log n)$ time such that the strong and weak visibility polygons of $\overline{st}$ can be updated in $O(\log n)$ time for each change when $\overline{st}$ moves in a given direction inside $\mathcal{P}$. Moreover, updates can be computed in $O(\log^2 n)$ time whenever $\overline{st}$ is allowed to change its motion direction.*

References

[1] B. Aronov, L. Guibas, M. Teichmann, L. Zhang Visibility Queries in Simple Polygons and Applications. *J. Descrete and Comp. geometry*, 27(4):461–483, 2002.

[2] H. A. ElGindy, D. Avis A Linear Algorithm for Computing the Visibility Polygon from a Point. *J. Algorithms*, 2:186–197, 1981.

[3] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan Visibility and Intersection Problems in Plane Geometry. *Proceedings of sym. on Comp. geometry*, p1–13, 1986.

[4] D. Lee Visibility of a simple polygon. *Vision, Graphics and Image Processing*, 22:207–221, 1983.

[5] B. Chazelle, L. Guibas Visibility of a simple polygon. *Discrete & Computational Geometry*, 4(6):551–581, 1989.

[6] S. Suri, J. O'Rourke Worst-Case Optimal Algorithms for Constructing Visibility Polygons with Holes. *Proceedings of sym. on Comp. geometry*, p14–23, 1986.

[7] P. Bose, A. Lubiw, J. Munro Efficient visibility queries in simple polygons. *Comp. Geometry: Theory and Applications*, 23(3):313–335, 2002.

[8] T. Asano An Efficient Algorithm for Finding the Visibility polygon for a Polygonal Region with Hole. *Trans. of IECE of Japan*, 9(68):557–559, 1985.

[9] John Hershberger An Optimal Visibility Graph Algorithm for Triangulated Simple Polygons. *Algorithmica*, 4(1): 141-155, 1989.