2009 International Conference on Computational Science and Its Applications

Visibility of a Moving Segment

Mojtaba Nouri Bygi
Computer Engineering Department
Sharif University of Technology
Tehran, Iran
Email: nouribaygi@ce.sharif.edu

Mohammad Ghodsi
Computer Engineering Department
Sharif University of Technology
Tehran, Iran
Email: ghodsi@sharif.edu

Abstract

In this paper we define topological segment visibility, and show how to compute and maintain it as the observer moves in the plane. There are n non-intersecting line segment objects in the plane, and we have a segment observer among them. As the topological visibility of a line segment has not been studied before, we first consider static case of the problem, in which the observer and objects are static, and then we study dynamic case of the problem, in which the observer can move among obstacles.

1. Introduction

Visibility is an important topic in computer graphics, motion planning, and computational geometry. To deal with the increasing complexity of the scenes considered, some research has been performed in visibility processing in order to accelerate the visibility determination, e.g., visibility graph and visibility complex.

Visibility of a point observer has been studied widely in computational geometry literature. In visibility computations, the object which is seen along rays (maximal free line segments) is determined. As recomputing visibility of moving object that can be seen along a ray is a time consuming task, some solutions are proposed for this problem. One proposed solution is to classify rays according to their visibility, that is, after finding the object along a given ray, we only need to identify the set of rays that contains the given ray and read the visibility properties of the specified rays. Such approaches could be called *topological visibility* in the sense of only considering the objects seen along a ray, and not the exact parts of the objects that can be seen.

In this paper we define combinatorial (or topological) segment visibility, and show how to compute and maintain it as the segment moves in the plane. Topology deals with the classification of spaces that are the same up to some equivalence relation [16].

Assume that there are n line segment objects in the plane, and we have a line segment observer among them. We say that the observer *sees* an object, if and only if there is a point in the observer that can see the object. This definition differs from other possible segment-visibility definitions in that here we just want to know whether an object is visible from the observer, but the exact portion of the object seen from the observer is not important. This kind of visibility definition is mainly practical in global visibility studies, e.g. [2], [12].

As the topological visibility from a line segment has not been studied before, we first consider static case of the problem, in which the observer and objects are static. Then we study dynamic case of the problem, in which the observer can move among obstacles.

The rest of the paper is organized as follows. In Section 2 we review important results on viewpoint visibility and some concepts related to our work. In Section 3 we define combinatorial visibility for a segment observer and study both static case and dynamic case, in which the observer can move among obstacles.

2. Preliminaries

2.1. Point Visibility

Visibility of a point observer has been studied widely in computer geometry literature. One proposed solution is to compute regions that have constant visibility (see Fig. 1) and switch among the regions as the point observer moves. By moving the observer, the visibility changes only along some special places, which are called *discontinuity lines* or *visibility events*. So, we can compute an arrangement of these lines that a scene contains. Usually a visibility graph is used to store the set of all discontinuity lines [8]. For n input segments, there would be $O(n^2)$ discontinuity lines, and the number of regions in the arrangement is $O(n^4)$.

Two algorithms whose internal structure is derived from the visibility graph are proposed to solve the problem, namely, Ghali and Stewart's algorithm [7], and Rivière's algorithm [13] for polygonal scenes. After deriving the internal structure, Rivière's algorithm creates another structure named visibility complex [12], which contains more information than the visibility graph while still has the same complexity as it. Ghali and Stewart use a duality concept in their algorithm to gain more efficiency. Geometric duality

978-0-7695-3701-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCSA.2009.31

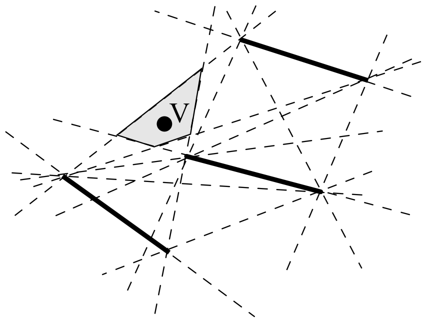


Figure 1. A region with the constant visibility, [10].

is a known tool which has been used in classical projective geometry for many years. This concept has had a useful role in solving a number of geometric problems in recent years (see e.g. [4]). We will describe these two approaches in more detail.

2.2. Discontinuity Lines

As stated before, visibility changes only along the discontinuity lines. So the algorithm has to specify discontinuity lines which are considered in it. Some algorithms are based on all discontinuity lines which are stored in a visibility graph [7], [13], while there are others algorithms that only consider a subset of all discontinuity lines [15], where lines are stored in a map which is updated during the path.

The visibility graph for a scene containing n points and m discontinuities can be constructed in $O(m+n\log n)$ time and requires $O(m)$ space [5], [9], in which m, the number of discontinuity lines, is $O(n^2)$. The mentioned map is also constructed in $O(n \log n)$ time and requires $O(n)$ space in the same scene.

2.3. Geometry Duality

Geometric duality is a known tool which has been used in classical projective geometry for many years. This concept has had a useful role in solving a number of geometric problems (see e.g. [4]). In duality transformation we map a set of more unfamiliar objects like lines and hyperplanes which are hard to think about, to another set of objects with which we have more intuition, like points.

For example a duality function D that maps a point (a, b) into the line $ax + by = 1$ and vice-versa is such a duality transformation in which the line and the point are called dual of each other. The plane containing the original data is the *primal* plane while the other plane to which original data are mapped is called the *dual* plane.

2.4. Computing a view around a point

In this section we review two algorithms presented for computing the view around a moving point.

2.4.1. Ghali and Stewart's Algorithm. As our approach is related to the Ghali and Stewart algorithm [6], [7], we will shortly review it in this section. The algorithm consists of the following steps:

Having n segments in the scene, they first build the visibility graph of them. For each segment endpoint they put a node and connect two nodes with an edge if and only if the line which connects the corresponding endpoints doesn't cross any of the segments. A discontinuity line in this graph is the line that supports two endpoints. Visibility graph is constructed in $O(m + n \log n)$ time and requires $O(m)$ space, in which m is the number of discontinuities and is bounded by $O(n^2)$ [8].

Next, the given set of discontinuity lines is preprocessed into a data structure using a duality concept that one can query it to know whether any of discontinuities has been crossed. By this duality transformation, each discontinuity line is mapped to a point and the viewpoint V is also mapped to a line V'. The line V' generates two separated sets of points (see Fig. 2).

By moving the viewpoint in the primal plane, its equivalent line in the dual plane is also moving. To know if the viewpoint in the primal plane crosses a discontinuity line, we can see if a point in the dual plane changes its position with respect to the line. To answer this question quickly, a convex hull for both sets of points in dual plane is constructed and only its boundary points are checked. Doing the duality transformation takes $O(m \log m)$ time and $O(m)$ space and deciding whether a discontinuity line is crossed with the viewpoint using the constructed convex hull takes $O(\log m)$.

The set of visible segments will be unchanged until a discontinuity line is crossed. After crossing one discontinuity line, the constructed data structure in previous step must be updated. It means that one point in the dual plane is moved from one side of the line to the other and so it must be removed from one convex hull and added to the other one. Ghali and Stewart's algorithm uses the Overmars and van Leeuwen's algorithm [11] to maintain a dynamic convex hull in which an insert/permit operation is done in $O(\log^2 m)$, which was the best known algorithm for maintaining the convex hull at the time, but later improved to $O(\log m)$ [3]. When the viewpoint crosses a discontinuity line, the set of visible segments is not necessarily updated but the convex hull is updated in order to maintain the description of the current cell.

2.4.2. Rivière's method . The algorithm introduced by Rivière [14] constructs another structure named *visibility*

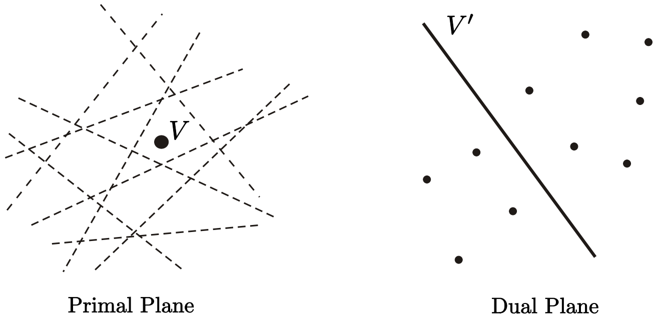


Figure 2. Schematic view of the situation in the primal and dual plane, [15].

complex [12] that has the same complexity as the visibility graph while containing more information than it. It encodes all the visibility relations between objects of a scene in the plane.

In the dual plane, the line V' passes through a set of faces of the visibility complex and the set of crossed edges represent the current visibility region. The exact boundary of the visibility region is found by applying a dynamic envelope to the vertices considered. The visibility complex of a polygonal scene has a size $O(m)$ and can be computed in optimal $O(m + n \log n)$ time [14].

In case of dynamic maintenance of visibility, we can decide in $O(\log v)$ time if a visibility event occurs and if so, update the structures in $O(\log^2 v)$, where v is the size of the view, i.e. the size of the set of crossed edges in the visibility complex [13].

3. Segment Visibility

In Section 2 we saw how to maintain the visibility along a viewpoint trajectory. In this section we will define combinatorial (or topological) segment visibility, and show how to compute and maintain it as the segment moves in the plane. Assume that there are n line segment objects in the plane, and a line segment observer is among them. We say that the observer *sees* an object, if and only if there is a point in the observer that can see the object (in the sense of Section 2). This definition differs from other possible segment-visibility definitions in that here we just want to know whether an object is visible from the segment observer, but the exact portion of the object seen from the observer is not important. This kind of visibility definition is mainly practical in global visibility studies, e.g. [2], [12]. As the topological visibility of a line segment has not been studied before, we first consider static case of the problem, in which the observer and objects are static. Then we study dynamic case of the problem, in which the observer can move among obstacles.

3.1. Static Case

First we consider the static case, in which the observer AB doesn't move in the scene along the objects (see Fig. 3). First we build the visibility graph for the given n line segments. The visibility edges, or discontinuity lines, divide the plane to some regions with constant visibility. Consider a view point p with initial location at the segment endpoint A. We can compute the visibility of A with one of the algorithms mentioned in Section 2. Here we use the method of [13] that uses visibility complex of the scene to compute and maintain the visibility from a viewpoint. Depending on the method used to find the crossed edges, the view can be computed in $O(v \log n)$ time or $O(n \log n)$ time (v is the size of the view) [13]. We move along the segment toward the other endpoint B. As long as we are on the same region, the visibility will not change. When we cross a discontinuity line, the visibility will change and we need to update the view. So, after the computation of v_0, the initial view around a point p, and a preprocessing step done in $O(v_0 \log v_0)$ time, the view along a trajectory can be maintained in $O(\log v)$ time at each change of visibility [13]. So, if the number of discontinuity lines that intersect the visibility segment is k, the total time needed to compute the view of visibility segment will be $O((n + k) \log n)$. Considering the time needed to construct the visibility complex of the scene, we have the following proposition.

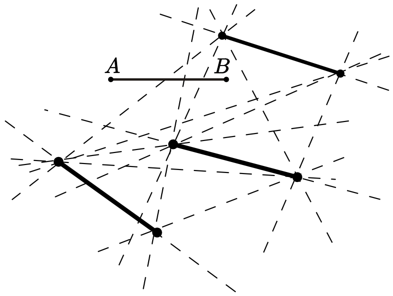


Figure 3. A line segment observer among line segment objects.

Proposition 3.1: After the computation of the visibility complex of a scene that composed of n line segments, and a preprocessing step done in $O(m + n \log n)$ time, where m is the number of discontinuity lines, the view from a line segment can be computed in $O((n+k)\log n)$, where k is the number of discontinuity lines that intersect the line segment.

3.2. Dynamic Case

Now we consider the dynamic case of the problem: A line segment observer s is moving among a set S of n line segment objects. We want to maintain the visibility of the observer as it moves. Our approach is to compute the initial visibility of the observer, as done in Section 3.1, and update it whenever it changes.

In order to compute the visibility, we need the descriptions of the set of cells that the observer intersects, and for each cell, the set of lines bounding the cell in the arrangement of lines in the primal plane. Since computing the arrangement is too expensive, we take the following approach, which is similar to the method of Ghali and Stewart [7]. We compute the dual of S, the set of discontinuity lines, and the dual of s, the line segment observer. Let the duals be $S' = D(S)$ and $w_s = D(s)$. In the dual plane, each discontinuity line l is mapped to a point p_l and the segment s is mapped to a double wedges w_s (see Fig. 4). Notice that segment endpoints A and B are mapped to two lines l_A and l_B, which are boundaries of w_s. The double wedge w_s divides the dual plane and the set of points S' into four sets S'_1, S'_2, S'_3 and S'_4, the first two sets are lying *outside*, and the other two sets are lying *inside* the wedge. When the dual of a discontinuity line is outside the wedge, it means that the line segment observer does not intersects the discontinuity line, and if the dual is inside the wedge, the observer intersects the discontinuity line. This observation helps us to easily see whether a discontinuity line crosses the observer or not.

Now let us see what happens when the visibility of the observer changes. As the observer moves in the plane, it crosses some visibility regions constructed by discontinuity lines. As long as the number and the order of these crossed regions do not change, the visibility of the observer will not change, and vice versa. So we need to show the conditions in which the crossed regions change.

A change in the set of crossed visibility regions is happening in two conditions: (i) the observer intersects a new discontinuity line, and so it enters a new region, or an already crossed discontinuity line no longer intersects the observer, and a visibility region will disappear (Fig. 5), (ii) the order of intersection of two discontinuity lines will change, and an old region will disappear and a new one will appear (Fig. 6). It can be easily seen that these two cases are the only situations that change the crossed regions. So, we need to consider these cases and update the visibility in each one.

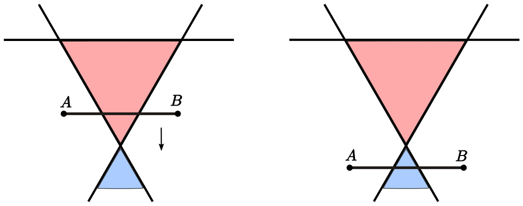


Figure 6. Event of type 2. The order of intersecting two discontinuity lines changes. A visibility region disappears and a new one appears.

3.2.1. Event Type 1.

As mentioned before, first type of events occurs in two cases: when the observer intersects a new discontinuity line, and it enters a new region, or when an already crossed discontinuity line no longer intersects the observer, and a visibility region will disappear (Fig. 5). When the observer intersects a new discontinuity line, it means that its dual point enters the double wedge. Similarly, when the observer stops to intersect a discontinuity line, it means that its dual point leaves the double wedge.

To detect these events, we only need to observe the dual lines of endpoints A and B. As the endpoints A (respectively B) gets closer to a line l on the boundary of a cell and becomes incident to it, the dual line $l_A = D(A)$ (respectively $l_B = D(B)$) becomes incident on the dual of that line $p_l = D(l)$. When A crosses l in the primal plane, the segment either enters another cell or leaves an already crossing cell. This is reflected in the dual plane by the point p_l crossing line l_A (see Fig. 7). In what follows, we discuss how to update the structure upon insertion and deletion of points.

As long as the set of crossed discontinuity lines do not change, no event of type 1 will occur. If a new one is crossed or stops crossing, the data structure has to be updated. In the dual plane, these events reflected by a point entering or removing the wedge. In these cases, the point must be removed from one convex hull and inserted into anther one (one of the convex hulls is outside the wedge, and the other one is inside it). So, we need to maintain the convex hulls as points are added to or removed from them. To maintain a dynamic convex hull, a $O(\log^2 n)$ update were achieved early on [11], and it was improved to $O(\log n)$ in amortized [3].

The set of visible segments is not always updated when the segment observer crosses a discontinuity line, although the convex hulls must be updated to maintain the description of the current cell.

Consider the two convex hulls outside the wedge. When inserting a point into a convex hull in the dual plane, the point will be connected to the two common tangents between it and the hull. The points from the previous convex hull boundary that disappear from the new one correspond to the discontinuities in the primal plane that no longer bound the current cell in which an endpoint of observer exists.

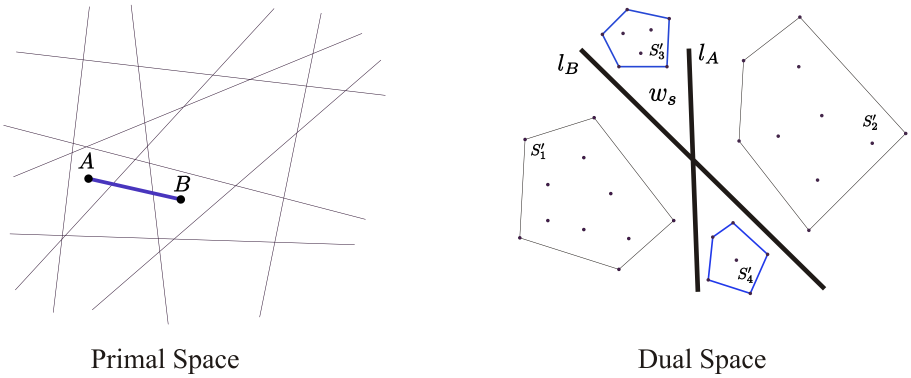


Figure 4. The figure on the left shows the arrangement of the discontinuity lines. The set of points shown on the right is the set of dual points of these lines. When the observer AB moves in the primal plane, its dual wedge w_s, also moves (and possibly rotates) in the dual plane. Intersecting a new discontinuity line (respectively, stopping to intersect one) in the primal plane is signaled in the dual plane by boundaries of the wedge, l_A or l_B, intersecting one of the two convex hulls outside (respectively, inside) the wedge in dual plane.

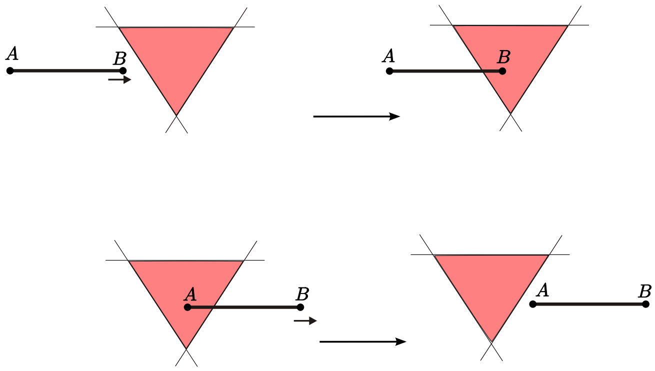


Figure 5. Events of type 1. In the figure above, the observer will intersect a new discontinuity line. The figure on the bottom shows that the observer stops intersecting a discontinuity line.

When deleting a point from a hull in the dual plane, an arbitrary number of points can appear on the hull boundary. These points correspond to the lines in the primal plane that bound the new cell which the viewpoint enters [7].

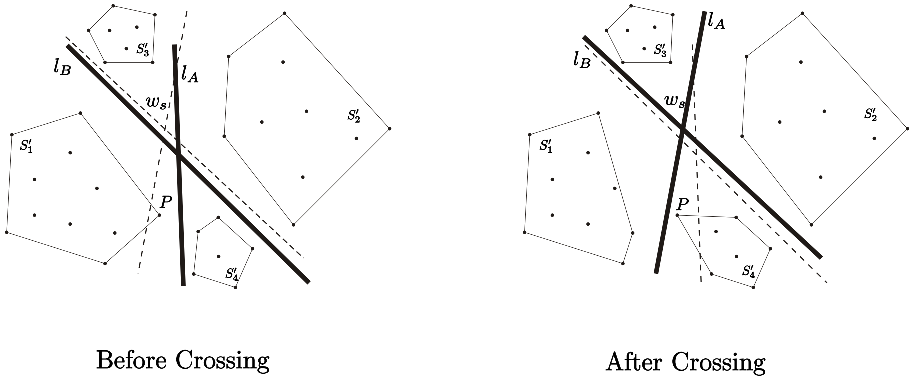
The two convex hulls that we maintain in the dual plane give us a concise description of the discontinuities in the primal plane which, when crossed, may require an update of the visibility cycle. The hulls can be stored in linear space with respect to the number of discontinuities. For every new positions of the endpoint A and B, the duals l_A and l_B are computed and a test performed to see if any point on either of the four convex hulls in the dual plane has switched sides.

In general, we have the following proposition:

Proposition 3.2: After a preprocessing step in $O(m + n \log n)$ and computing the initial view of a line segment observer in $O((n + k) \log n)$ time, where m is the number of discontinuity lines and k is the discontinuities that initially intersect the line segment, the view from the observer among n line segment objects can be maintained in $O(\log n)$ each time a visibility change of type 1 occurs.

3.2.2. Event Type 2. In the second type of visibility events, the order of intersection of two discontinuity lines changes, and an old region will disappear and a new one will appear (Fig. 6). Notice that in this case, the set of crossed discontinuities will not change, and in dual plane, this means that the partition of points that the wedge has been created will not change.

Now let us see what happens in the dual plane upon occurring an event of type 2 (See Fig. 8). As the observer AB come near to the intersection point of two discontinuity lines l_1 and l_2, the lines wp_{l_1} and wp_{l_2} in the dual plane come closer to each other, where w is the center of the wedge. At the event time, when AB passes through the intersection point of l_1 and l_2, wp_{l_1} and wp_{l_2} will lie on each other. After the event, the order of crossing these two discontinuity lines will change, and wp_{l_1} and wp_{l_2} will switch their polar order according to wedge center. It can be easily shown that each time two dual points p_{l_1} and p_{l_2}



Before Crossing

After Crossing

Figure 7. The figure on the left (respectively, right) shows the configuration in the dual plane before (respectively, after) a discontinuity line is crossed. In both figures, the solid lines are the duals of the current position of the observer endpoint A and B with the dashed lines showing the position of the duals after (respectively, before) crossing. As the discontinuity point P is crossed, P is deleted from one set of points and inserted into another one, while maintaining the convex hulls of both sets of points.

and wedge center w become collinear, an event of type 2 is occurred, and vice versa.

Now we show how to handle this kind of event. We claim that this event can be handled with the algorithms presented in Section 2 for computing visibility of a moving viewpoint. Assume that there are $m' = O(m) = O(n^2)$ points *inside* the wedge. Consider these m' points as endpoint objects and think of wedge center, w, as a moving viewpoint. When two of these endpoints become collinear with w, we can say that w has crossed the discontinuity line that passes through these two points.

With m' points, there will be $m'' = \Theta(m'^2) = O(n^4)$ discontinuity lines (notice that in average, m', the number of crossed discontinuities, is very smaller than $O(n^2)$). Using the algorithms of [13] mentioned in Section 2.4.2, we can handle events of type 2. As there are m'' discontinuity lines, we can compute the visibility complex of the scene in $O(m'' + m' \log m') = O(n^4 + n^2 \log n)$ time and $O(m') = O(n^2)$ working space. Having the visibility complex of the scene, each visibility event can be found and handled in $O(\log n)$ time. Each visibility event corresponds to a time that the wedge center w becomes collinear with two other points in the wedge, and an event of type 2 is occurred.

Two situations which are not considered yet are when an event of type 1 occurs, in which a new dual point will be added to the wedge, or when a point will remove the wedge. In these cases we must update the visibility complex of the points inside the wedge. When a new point enters the wedge, we have to insert this object in our visibility complex. This is done by computing a view around the point using the current visibility complex which can be done in $O(m' \log n)$ where m' is the number of points inside the wedge, using the techniques described by Rivière [13]. Similarly, when a point leaves the wedge, the edges of the visibility complex corresponding to segments passing through this point must be removed. This work can be done in $O(m')$.

In general, this approach leads to the following proposition:

Proposition 3.3: After a preprocessing step in $O(m'^2 + (m'+n) \log n)$ and computing the initial view of a line segment observer in $O((m'+n) \log n)$, where m' is the number of discontinuity lines that intersect the segment observer, we can handle an event of type 1 in $O(\log n + m' \log n)$ time and an event of type 2 in $O(\log n)$ time. In other words, if there are k_1 events of type 1 and k_2 events of type 2, we can update the view in $O(k_1(\log n + m' \log n) + k_2 \log n)$.

Another approach for handling visibility events of type 2 is to use Kinetic Data Structure [1]. Kinetic Data Structure (KDS) is a framework for maintaining certain attributes of a set of objects moving in a continuous manner. For example, KDS has been used for maintaining the convex hull of moving objects, or maintaining the closest distance among moving objects. A KDS mainly consists of two parts: a description of the needed attributes and some certificates such that a certificate remains unchanged as long as its attribute does not change. It is assumed that we can efficiently compute the failure time of each of these certificates. Only when a certificate fails the KDS must be updated, otherwise it remains valid.

We claim that to find an event of type 2, we only need to have an ordered list of points in the wedge according to their polar angles, with w as their center. It can be easily seen that when an event of type 2 occurs, for a small time before the event, two points participating in the events becomes adjacent to each other in the list. To maintain this ordered list of points, we need m' certificates. If the sorted list of sites is $s_{i_0}, s_{i_1}, \ldots, s_{i_{m'-1}}$, we need the certificates $s_{i_0} < s_{i_1}, s_{i_1} < s_{i_2}, \ldots, s_{i_{m'-2}} < s_{i_{m'-1}}$.

An event happens if a certificate fails. All the events are placed in a priority queue, sorted by the time they occur.

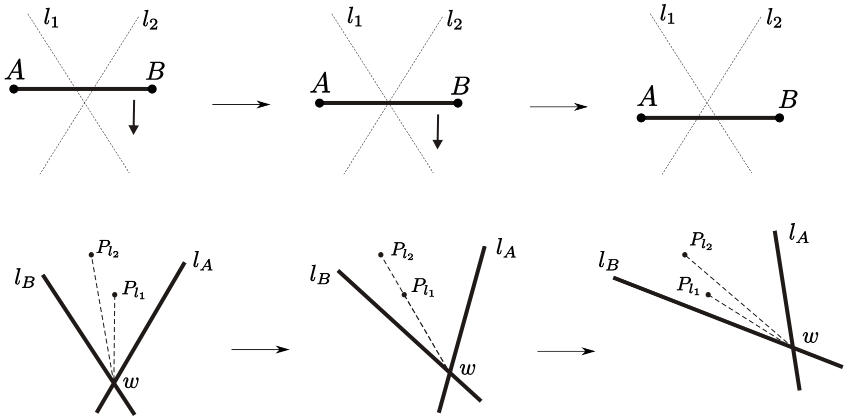


Figure 8. An event of type 2. As the observer AB approaches the intersection of discontinuity lines l_1 and l_2 (top), in dual plane, the lines $\overline{wp_{l_1}}$ and $\overline{wp_{l_2}}$ come closer to each other (bottom). At the event point (center) where AB passes through the intersection of l_1 and l_2, $\overline{wp_{l_1}}$ and $\overline{wp_{l_2}}$ will lie on each other. After the event, the order of crossing these two discontinuity lines will change, and in dual plane, $\overline{wp_{l_1}}$ and $\overline{wp_{l_2}}$ will switch their polar order according to wedge center (right).

When an event happens, we examine the proof (the set of certificates) and update it. The initial event list can be built in $O(m \log m)$ time, using a suitable event queue. For each event, we need to update at most two certificates. We just need to find these certificates in the proof scheme and replace them with the new ones, which takes $O(\log n)$ time.

Upon occurring an event of type 2, we must update the visibility of the segment observer. This can be done by finding the corresponding *face* in the visibility complex of the segment objects which takes $O(\log n)$ time [12].

When an event of type 1 occurs, we need to update our certificate list. This can be done in $O(1)$, because when a new point enters the wedge or leave it, it must always be the first or last point in the ordered list of points (remember that this point must be adjacent to dual of A or B just before changing its position with respect to wedge). In general we have the following proposition.

Proposition 3.4: After a preprocessing step in $O(m + (m' + n) \log n)$ and computing the initial view of a line segment observer in $O((m' + n) \log n)$, where m is the number of discontinuity lines and m' is the number of those discontinuities that intersect the line segment, we can find the next event that changes the visibility of the observer in $O(\log n)$ and update the visibility in $O(\log n)$.

According to the criteria of a good KDS [1] we must evaluate our kinetic model. Similar to other algorithms, a good KDS should take small space, small initialization cost, and efficient update time. In KDS, an update may happen in two cases. One is when a certificate fails and an event happens. The other is when the motion of an object changes. In the first case, we need to update the certificate set, and in the second case we must recompute the failure times for all the certificates of that object. These requirements induce the following quality measurements for KDSs [1]. *Compactness*, the size of the proof, which is $O(m')$ in our case. *Responsiveness*, the time to process an event, which is $O(\log n)$ for processing an event. There are $O(1)$ certificates need to reschedule and each reschedule takes $O(\log n)$ time. *Locality*, the number of certificates that a single object involves in, which is two in our algorithm, as each object is involved in at most two certificates. The last quality measurement is *efficiency* which is the number of events processed. In our algorithm, all events are *exterior* [1] (change both KDS and data structures of algorithm), and in any case, we have to handle them to maintain the visibility.

4. Conclusion

In this paper we defined topological visibility for a line segment observer among line segment objects in the plane. We proposed algorithms for computing the view in both static and dynamic cases.

In this work, we created the complete arrangement of all discontinuity lines. It would be better if one can use a local approach for the problem and avoid unnecessary computations. Another interesting problem is to scale upwards and introduce the problem in three dimensions. In the 3D case, there are some complications that make this a non-trivial task.

References

[1] J. Basch, L. J. Guibas, and J. Hershberger. Data structures for mobile data. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 747–756, 1997.

[2] J. Bittner and P. Wonka. Visibility in computer graphics. *Journal of Environmental Planning*, page 5, 2002.

[3] G. S. Brodal and R. Jacob. Dynamic planar convex hull. In *In Proc. 43rd IEEE Sympos. Found. Comput. Sci*, pages 617–626, 2002.

[4] B. Chazelle, L. J. Guibas, and D. T. Lee. The power of geometric duality. In *Proc. 24th Annu. IEEE Sympos. Found. Comput. Sci.*, pages 217–225, 1983.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin, 1997.

[6] S. Ghali and A. J. Stewart. Incremental update of the visibility map as seen by a moving viewpoint in two dimensions. In *Seventh International Eurographics Workshop on Computer Animation and Simulation*, pages 1–11, Aug. 1996.

[7] S. Ghali and A. J. Stewart. Maintenance of the set of segments visible from a moving viewpoint in two dimensions. In *Proc. 12th Annu. ACM Sympos. Comput. Geom.*, pages V3–V4, 1996.

[8] S. K. Ghosh and D. M. Mount. An output sensitive algorithm for computing visibility graphs. Technical Report CS-TR-1874, Department of Computer Science, University of Maryland, July 1987.

[9] J. E. Goodman. Pseudoline arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 5, pages 83–110. CRC Press LLC, Boca Raton, FL, 1997.

[10] K. Nechvle and P. Tobola. Dynamic visibility in the plane. In *Proc. Seventh Int. Conf. in Central Europe on Computer Graphics and Visualization, WSCG '99*, pages 187–194, 1999.

[11] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[12] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6(3):279–308, 1996.

[13] S. Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Proc. 13th Annu. ACM Sympos. Comput. Geom.*, pages 421–423, 1997.

[14] S. Riviére. *Visibility computations in* 2*D polygonal scenes*. PhD thesis, Université Joseph Fourier, Grenoble, France, 1997.

[15] P. Tobola. Local approach to dynamic visibility in the plane. In *Proceedings of the 7-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media'99*, pages 202–208, 1999.

[16] G. Vegter. Computational topology. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, chapter 32. CRC Press LLC, Boca Raton, FL, 2th edition, 2004.