Dynamic visibility in polygonal scenes with the visibility complex

RIVIÈRE Stéphane *

1 Introduction

This paper proposes methods for efficient maintaining of the view around a point (i.e., the visibility polygon) in static and dynamic polygonal scenes in the plane. The algorithms presented use the *visibility complex*, that we have adapted for polygonal scenes (composed of disjoint simple polygons) to take advantage of the spatial and temporal coherence.

The first algorithm shows how to maintain the view around a moving point in $O(\log^2 v)$ time at each visibility change (v: current size of the view), improving the $O(\log^2 n)$ time bound (n: number of polygon vertices) of [GS96], and is well-suited for small consecutive moves. The second algorithm maintains the view around a point moving along a known trajectory: After a preprocessing in $O(v \log v)$ time, the view is updated in $O(\log v)$ time at each change of visibility. Finally, we show how to maintain the visibility complex for a dynamic scene (i.e., a scene in which the polygon vertices are moving) and how to maintain the view around a moving point in a dynamic scene.

2 The visibility complex

Visibility computations involve determining the object seen along directions of vision, that is along maximal free line segments (line segments of maximal length in free space), that we also call *rays*. Recomputing each time the object seen along a ray can be too time consuming for solving some visibility problems. One solution to this issue is to classify rays according to their visibility: to find the object seen along a ray, we then just have to identify the set of rays that contains the given ray and to read the visibility properties of this set.

Pocchiola and Vegter [PV96] devised a new data structure, the visibility complex, which represents sets of rays having the same visibility properties. A ray going from an object O_l to another object O_r being characterized by its label (O_l, O_r), the visibility complex is the quotient space of the space of rays under the following relation $\sim$: $r_1 \sim r_2$ iff r_1 can be moved continuously to r_2 while keeping the same label.

We have adapted here this structure, created initially for scenes of convex curved objects, for polygonal scenes: Each side of a polygon is a distinct object. Then the complex is composed of three types of elements: faces (2D components), edges (1D components representing rays passing through a polygon vertex), and vertices (0D components representing rays passing through two polygon vertices).

These elements are best handled by means of a duality relation (figure 1). We consider the duality relation that maps the line $l : y = ax + b$ of the scene into the point $l^* : (a, b)$ in the dual space. This duality maps the set of lines passing through a point $p : (a, b)$ of the scene into the dual curve $p^* : y = -ax+b$. An edge e of the complex is contained in a dual curve p^*: We say that p is the point associated with e. A vertex v of the complex is the intersection of two dual curves p^* and q^*; in the scene, (p, q) is an edge of the visibility graph, and we say that the ray v^* containing (p, q) is the ray associated with v.

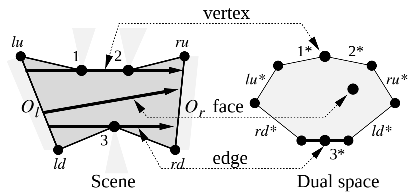

Figure 1: Visibility complex in dual space.

The visibility complex of polygonal scenes has a size $O(k)$ (k size of the visibility graph) and can be computed in optimal $O(n \log n + k)$ time and $O(n)$ working space with the algorithm of Rivière [Riv97].

Pocchiola and Vegter [PV96] have shown how to use the visibility complex to compute a view around a point. We recall here the algorithm for we reuse its results later.

3 Computing a view around a point

Computing the view around a point of view p_v amounts to computing changes of visibility occurring along a ray pivoting around p_v. A change of visibility occurs when the ray is leaving the face f of the visibility complex to which it belongs, and is entering into a new face f'.

*iMAGIS-GRAVIR/IMAG - BP 53
38041 GRENOBLE CEDEX 09 - FRANCE
e-mail: Stephane.Riviere@imag.fr
www: http://www-imagis.imag.fr/Membres/Stephane.Riviere
iMAGIS is a joint project of CNRS/INRIA/INPG/UJF

In the dual space, the set of rays passing through p_v is a dual curve p_v^* which passes through a set of faces of the visibility complex. The dual curve passes from one face f to another face f' by crossing an edge e separating the two faces; the point p associated with e is then a point of the view. The algorithm for computing the view computes the list of edges of the complex crossed by the dual line of the point of view. Depending on the method used to find the crossed edges, the view can be computed in $O(v \log n)$ time, or in $\Omega(v)$ and $O(vn)$ time (v size of the view).

With this list of crossed edges, we can maintain the view when the point of view is moving in the scene.

4 Maintaining a view around a moving point

Let p_v be a point moving in the scene. The view around p_v changes when p_v crosses an extended edge of the visibility graph (i.e., the ray containing the edge): A point of the view becomes invisible, or a new point becomes visible, or p_v crosses the non-extended edge. In the last case, although the view does not change, we still classify this event as a visibility change: It corresponds to a visibility event in the algorithms that we describe.

By extending each edge of the visibility graph in free space, we obtain a partition of free space into regions of constant visibility: The view around a point is the same for all points inside the same *visibility region*. To maintain the view around p_v, we do not compute all the $O(n^4)$ visibility regions, but only the limits of the visibility region that contains p_v.

When p_v is moving, we check whether p_v has left its visibility region; if so, we search the side of the visibility region crossed by p_v and compute the limits of the region incident to that side. We repeat the process until finding the new visibility region containing p_v.

We compute the limits of the visibility region of p_v by using the list of edges of the complex crossed by the dual curve p_v^*. Let us consider a crossed edge $e = (v_l, v_r)$ whose extremities are the vertices v_l and v_r. If p_v^* passes through one of these vertices, then the list of crossed edges — and therefore the view around p_v — changes. Let p be the point associated with e. In the scene, the two rays v_l^* and v_r^* associated with v_l and v_r respectively, form a wedge that contains p_v and no other (extended) edges of the visibility graph incident to p: The wedge contains the visibility region of p_v (figure 2).

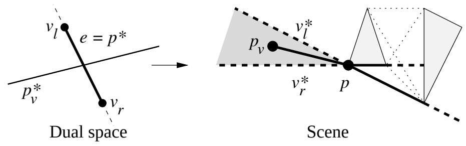

Figure 2: Wedge of visibility given by a point of the view.

More precisely, each wedge is made of an upper line and a lower line. The limits of the visibility region of p_v is then composed of the lower envelope of the set of upper lines and of the upper envelope of the set of lower lines (figure 3).

Once the view around p_v is computed with the visibility complex, the limits of the visibility region of p_v can be computed in $O(v \log v)$ time (v size of the view).

When the point of view is moving from p_v to p'_v, we can check in logarithmic time whether the point has left its region and, if it is the case, find which line of the upper (resp. lower) has been crossed by (p_v, p'_v). In the visibility complex, the vertex v corresponding to the crossed line is swept by the dual curve p_v^*. Changes in the list of crossed edges (and therefore in the view) are local: They concern only the edges around the swept vertex and are done in constant time.

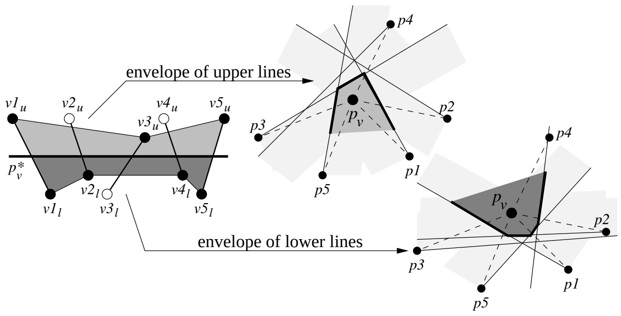

Figure 3: Computing the limits of the visibility region.

We must then update the limits of the visibility region. The changes are local too: We must remove (resp. insert) at most two lines in the set of lines, and the envelopes can be maintained by adapting the algorithms of Overmars and van Leeuwen [OvL81] or of Dobrindt and Yvinec [DY93] for dynamically maintaining a convex envelope.

Proposition 1 *After computing the initial view around a point p_v and a preprocessing step in time $O(v_0 \log v_0)$ (v_0 size of view at $t = 0$), the view around a moving point can be maintained in $O(\log^2 v)$ time (resp. randomized $O(\log v)$ time).*

We suppose now that the point of view is moving along a known trajectory, given for example by its position $p_v(t)$, $t \in [0, 1]$. We do not need to compute all the limits of each visibility region crossed by the trajectory, but only the limits cut by the trajectory.

We suppose that we can compute in constant time the point of intersection of the trajectory with a line, and that we can compare in constant time two points on the trajectory (i.e., establish which one is before the other). It allows us to use a priority queue in which lines are ordered according to the position of their intersection point with the trajectory, so that we can pass the limits of visibility in order.

To initialize the priority queue, we first compute the view around $p_v(0)$, and for each edge of the complex crossed by $p_v^*(0)$, we check whether the upper line (resp. lower line) intersects the trajectory and — if it is the case — put it in the priority queue. Then, at each step we consider the first line of the priority queue, line which indicates the limit of visibility being crossed. We update (locally) the view as in the previous algorithm, consider the (at most two) new lines that limit the region, and put them in the priority queue if they cut the trajectory.

Proposition 2 *After the computation of the initial view around a point p_v 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.*

5 Dynamically maintaining the visibility complex

We now show how the visibility complex can be maintained when the polygon vertices are moving in the scene. More precisely, each of the n polygon vertices is defined by its position $p_i(t)$ for $t \in [0,1]$, and we want to maintain the topological structure of the complex from $t = 0$ to $t = 1$. Modifications in the complex arise at *elementary visibility changes*. They occur when three points p_1, p_2, and p_3 such that (p_1, p_2) and (p_2, p_3) are edges of the visibility graph, change their relative positions: p_2 previously below (resp. above) (p_1, p_3) becomes above (resp. below) (p_1, p_3) (figure 4a).

Corresponding changes in the visibility complex are local: They occur around the vertices v_{12} and v_{23} corresponding to (p_1, p_2) and (p_2, p_3) respectively (figure 4b), and can be made in constant time (there are 50 cases to consider).

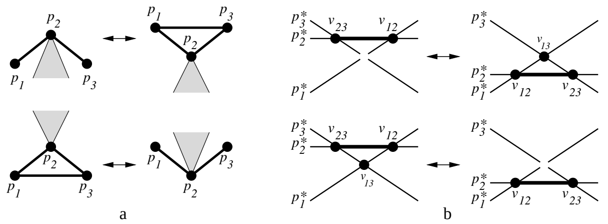

Figure 4: Elementary visibility changes. **a.** In the scene. **b.** In the visibility complex.

To maintain the complex from $t = 0$ to $t = 1$, we must handle all the visibility changes in order. We compute for each edge $e_j = (v_{ij}, v_{jk})$ $(v_{ij} = p_i^* \cap p_j^*)$ of the complex a "date of death" t_d that is the time when the three corresponding points $p_i(t_d)$, $p_j(t_d)$, and $p_k(t_d)$ in the scene will be collinear. Then we put the edges in a priority queue and handle the visibility changes in order. At each visibility change, the set of dates of death can be updated in constant time and the priority queue can be updated in $O(\log k) = O(\log n)$ time (k, size of the visibility graph, verifies $k = O(n^2)$). If dates of death can be computed in constant time, then:

Proposition 3 *After a preprocessing step in $O(k_0 \log n)$ time (k_0 size of the visibility complex at $t = 0$), the visibility complex of a moving polygonal scene can be maintained in $O(\log n)$ time at each visibility change.*

6 Maintaining a view around a moving point among moving polygons

We finally show how to maintain a view around a point when the point of view and the obstacles are moving. In this context, using the data structures shown before for maintaining a view (dynamic convex envelope or priority queue) is not well-suited: At each change in the visibility complex, these structures must be updated and the modifications may not be local. Thus we integrate the visibility changes around the point of view with the visibility changes of the visibility complex.

For each vertex v_{ij} of the complex extremity of an edge crossed by the dual curve p_v^* we also compute a "date of death", that is the date t_d when $p_v^*(t_d)$ passes through $v_{ij}(t_d)$, that is when in the scene $p_v(t_d)$, $p_i(t_d)$, and $p_j(t_d)$ are collinear, and put all these events in the priority queue.

When an elementary visibility change occurs in the complex, some dates of death related to the view around p_v must also be updated. However, these changes are local, and if for each edge put in the priority queue we add a flag indicating whether the edge is crossed by the dual curve of the point of view, all modified dates of death can be updated in constant time.

Proposition 4 *After a preprocessing step in $O(k_0 \log n)$ time (k_0 size of the visibility complex at $t = 0$), the view around a moving point among moving polygons can be maintained in $O(\log n)$ time each time a visibility change occurs in the scene or in the view.*

7 Conclusion

We have shown two output-sensitive algorithms for maintaining the view around a moving point in a fixed scene. These algorithms use the visibility complex so that local changes of visibility in the scene yield also "local" recomputations. They are particularly efficient in large occluded environments, where the size of the visibility complex is rather linear than quadratic (in the number of polygon vertices) and the size of the view is much smaller that the size of the scene. They are relatively simple and have been implemented.

Then we have shown an algorithm for maintaining the visibility complex for dynamic scenes which can be used in global visibility computations. The algorithm handles the changes of visibility according to their date of occurrence. We are currently studying whether visibility changes happening in different hidden parts of the scene could be handled independently in any order. We are also studying algorithms for maintaining the complex upon insertion or deletion of polygons in a scene, algorithms which could reuse (for a first study) our algorithms for dynamic visibility.

Finally, we have shown an algorithm for maintaining the view around a point moving in a dynamic scene, algorithm more suited to the case when most of the objects of the scene are seen at some point in time from the point of view.

References

[DY93] K. Dobrindt and M. Yvinec. Remembering conflicts in history yields dynamic algorithms. In *Proc. 4th Annu. Internat. Sympos. Algorithms Comput. (ISAAC 93)*, volume 762 of *Lecture Notes in Computer Science*, pages 21–30. Springer-Verlag, 1993.

[GS96] S. Ghali and A.J. Stewart. Incremental update of the visibility map as seen by a moving viewpoint in two dimensions. In *Proc. Eurographics Workshop on Comp. Anim. and Simul.*, pages 3–14, 1996.

[OvL81] M. H. Overmars and J. van Leeuwen. Maintenance of configurations in the plane. *J. Comput. Syst. Sci.*, 23:166–204, 1981.

[PV96] M. Pocchiola and G. Vegter. The visibility complex. *Internat. J. Comput. Geom. Appl.*, 6(3):279–308, 1996. Special issue devoted to ACM-SoCG'93.

[Riv97] S. Rivière. *Visibility computations in 2D polygonal scenes*. Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 1997.