Walking in the visibility complex with applications to visibility polygons and dynamic visibility

RIVIÈRE Stéphane *

1 Introduction

The visibility complex is a data structure that encodes all visibility relations between objects of a scene in the plane. However, in some applications only a subset of these informations is relevant at a time, in particular visibility informations about rays issued from (resp. going to) a given object O.

We first consider the set of faces of the complex representing rays issued from a given object. We define an order on these faces and show, once the visibility complex is computed, how to visit all of them with no additional data structure in optimal time $O(k_f)$, where k_f is the number of faces visited. Then we show how to use this walk to perform a topological sweep of the vertices incident to these faces in optimal time $O(k_v)$, where k_v is the number of these vertices, still with no additional data structure.

We next consider the set of faces of the complex representing rays issued from the "blue sky" and passing through a line segment s which is outside the convex hull of the scene. We show that the previous algorithms can be adapted simply in this new context, and we use these walks for two applications. First, we show how to compute the visibility polygon of a line segment $s = (p, q)$ outside the convex hull of the scene in $O(vis(s)+t_v(p)+t_v(q))$ time, where $vis(s)$ is the size of the visibility polygon and $t_v(p)$ (resp. $t_v(q)$) is the time to compute the view around p (resp. q). Second, we show how to maintain the view around a point moving from p to q. Once the view around p is computed, the algorithm has a total running time $O(\max(v(p), v(p, q)))$, where $v(p)$ is the size of the view around p and $v(p, q)$ the number of changes of visibility along (p, q). This algorithm is an alternative to those we have described in [Riv97b].

*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

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 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] have 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 this structure, created initially for scenes of convex curved objects, for polygonal scenes: Each side of a polygon is a distinct object (there is an additional object O_∞ which represents the "blue sky" surrounding the scene). The visibility 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, that is edges of the visibility graph). We say that an element of the complex representing rays of label (O_l, O_r) has itself a label (O_l, O_r).

These elements are best handled by means of a duality relation, which maps a line l of the scene into a point l^* in a dual space, and maps all the lines passing through a point p in the scene into a dual curve p^*. The only relevant informations in dual space are topological relations between elements of the complex. We use here any duality relation equivalent to $l : y\cos\theta - x\sin\theta - u = 0 \mapsto l^* : (\theta, u)$, that is, with the following property: If l and l' are two parallel lines such that l' is above l, then in dual space l^* and l'^* are two points with the same x-coordinate and l'^* is above l^*. This property implies that if a point p is below (resp. above) a line l, then the dual point l^* is above (resp. below) the dual curve p^*.

Figure 1 shows the elements of the complex represented in dual space. This figure also shows the general structure of a

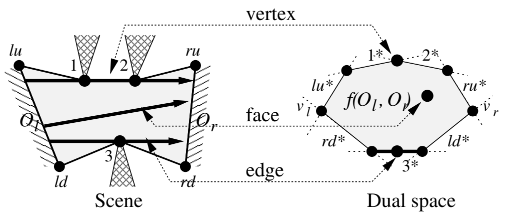

Figure 1: Visibility complex in dual space.

face. A face has two extremal vertices (v_l and v_r) that separate its frontier into two chains of edges: a chain of upper edges (lu^*, 1^*, 2^*, and ru^*) and a chain of lower edges (rd^*, 3^*, and ld^*). The complex has a lower and an upper semi-infinite faces of label (O_∞, O_∞) that represent rays that are outside the convex hull of the scene.

Edges of the complex corresponding to rays passing through p have p^* as supporting curve. As shown in Figure 2, edges whose supporting curve is ru^* (resp. rd^*, lu^*, and ld^*) can be in fact divided into sub-edges: Such edges are then called *fat edges*. They can occur only at the beginning or at the end of chains of edges and will be called *left/right-upper/lower* fat edges.

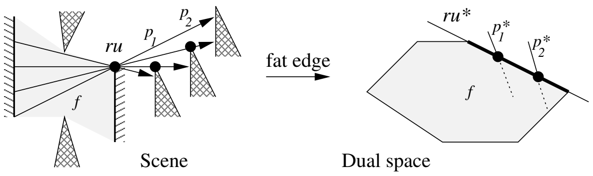

Figure 2: Right-upper fat edges.

The visibility complex of a polygonal scene of n total polygon vertices 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 we have described in [Riv97a].

3 Walking in $\mathcal{C}_l(O)$, elements of the complex of label $(O, .)$

The visibility complex encodes all visibility relations between objects of the scene. However, in some applications only a subset of these relations must be used at a time, and visiting all the elements of the complex is then a waste of time. For example, in lighting simulations (see [ORDP96] for more informations on how to use the visibility complex for radiosity computations), only faces of label $(O, .)$ (i.e., O is the left object of their label) must be processed to update the illumination of an object O.

Let $\mathcal{C}_l(O)$ denote the set of elements of the complex of label $(O, .)$ (we do not include in $\mathcal{C}_l(O_\infty)$ the two semi-infinite faces of label (O_∞, O_∞)). To visit all faces of $\mathcal{C}_l(O)$, we define an order on theses faces, and visit them in order.

Let us consider two faces f and f' of $\mathcal{C}_l(O)$ incident to a same edge e. Notice that, since both faces have label $(O, .)$, one face is above e and the other one is below e, that is e is an upper edge of one face and a lower edge of the other. If f is below e, then f' is above e and we say that $f < f'$ (and vice versa). We can extend this order: We say that $f \prec f'$ if there is a sequence (f_i) of faces of $\mathcal{C}_l(O)$ such that $f_0 = f$, $f_k = f'$ and $\forall\, 0 \leq i < k$, $f_i < f_{i+1}$. It can be proved that any two faces f and f' of $\mathcal{C}_l(O)$ are comparable with respect to $\prec$, therefore:

Proposition 1 *The relation $\prec$ on faces of $\mathcal{C}_l(O)$ defined by $f \prec f'$ iff there exists a sequence (f_i) of faces of $\mathcal{C}_l(O)$ such that $f_0 = f$, $f_k = f'$ and $\forall\, 0 \leq i < k$ there exists an edge e_i that is an upper edge of f_i and a lower edge of f_{i+1}, is a total order on faces of $\mathcal{C}_l(O)$.*

Given a face f, we can now define its previous face $prev(f) = max_\prec\{f' \in \mathcal{C}_l(O) \mid f' \prec f\}$ and its next face $next(f) = min_\prec\{f' \in \mathcal{C}_l(O) \mid f \prec f'\}$.

All faces incident to a lower (resp. upper) edge of f are inferior (resp. superior) to f . These inferior faces are of two

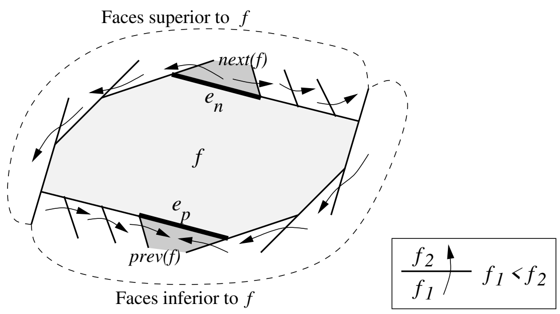

Figure 3: Ordering of faces of $\mathcal{C}_l(O)$.

sorts: faces incident to sub-edges of the left-lower fat edge, that are ordered from left to right, and faces incident to normal lower edges, that are ordered from right to left (figure 3). Therefore, the previous face of f is the face incident to the last sub-edge of the left-lower fat edge of f if this edge exists, incident to the first lower edge else. Similarly, the next face of f is the face incident to the first sub-edge of the right-upper fat edge if this edge exists, incident to the last upper edge else.

A face always has a previous and a next face in $\mathcal{C}_l(O_\infty)$. If the complex is "warped" modulo 2π (i.e., rays of slopes differing by 2π are identified), then by starting from f and always walking into the next (resp. previous) face we visit all faces of $\mathcal{C}_l(O_\infty)$ and "come back" to f. Figure 4 shows an example of walk in $\mathcal{C}_l(O_\infty)$ when faces are visited in decreasing order.

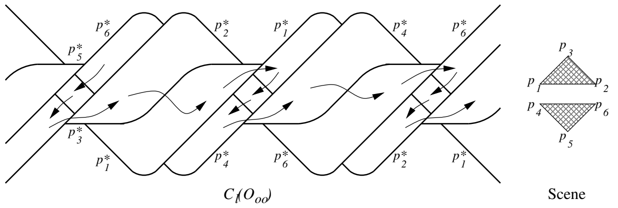

Figure 4: Visiting faces of $\mathcal{C}_l(O_\infty)$ in decreasing order.

On the other hand, if O is a line segment (p,q) of the scene, then the previous (resp. next) face of f may not belongs to $\mathcal{C}_l(O)$. If f has only one lower edge whose supporting curve is p^*, then $prev(f)$ does not belongs to $\mathcal{C}_l(O)$. In this case, the left extremal vertex of f is $v_l = p^* \cap q^*$ and corresponds to the ray (p,q): f is the first face of $\mathcal{C}_l(O)$. Likewise, if f has only one upper edge whose supporting curve is q^*, then $next(f)$ does not belongs to $\mathcal{C}_l(O)$. The right extremal vertex of f is $v_r = q^* \cap p^*$ and corresponds to the ray (p,q): f is the last face of $\mathcal{C}_l(O)$.

So $\mathcal{C}_l(O)$ has a finite number of faces that are located between the curves q^* and p^*. To visit all faces of $\mathcal{C}_l(O)$, we just start from its first face (which can be found in $O(\log n)$ time if needed) and walk into the next face until encountering the last face.

If each face has a pointor to the first (resp. last) sub-edges of its right (resp. left) fat edges (pointors that can be computed during the construction of the complex), then the previous and the next faces of f can be found in constant time:

Proposition 2 *Without any supplementary data structure, faces of $\mathcal{C}_l(O)$ can be visited in increasing (resp. decreasing) order in optimal time proportional to the number of visited faces.*

4 Topologically sweeping vertices of $\mathcal{C}_l(O)$

We have seen that we can visit the faces of $\mathcal{C}_l(O)$ in optimal time. If for each face f we visit each of its incident vertices, then we can visit all vertices of $\mathcal{C}_l(O)$ in optimal time: Such vertices are incident to at most three faces of $\mathcal{C}_l(O)$. However, visiting a vertex several time is not practical: If a vertex must be processed only once, then we must keep an historic to check whether a vertex has been visited before. Moreover, there is a natural partial order on the vertices of the complex — $v < v'$ if there is a monotonous path of consecutive edges from v to v' — and it is more interesting to perform a topological sweep of the vertices of $\mathcal{C}_l(O)$, that is, to visit then in a way compatible with their order: We can then use the coherence of the sweep to update informations in constant time instead of recomputing them each time.

We use the walk devised in the previous section to do a topological sweep of these vertices in increasing order: We walk on the faces of $\mathcal{C}_l(O)$, and for each visited face f we sweep some of its incident vertices.

Those swept vertices must be chosen so that when the walk is completed, (1) each vertex of $\mathcal{C}_l(O)$ has been swept exactly once (therefore only a subset of vertices of f must be swept when f is visited), (2) the sweep is coherent locally, that is, for each face f its vertices have been swept from left to right, and (3) the sweep is coherent globally. Property (2) is a consequence of property (3), but we mention it because it helps to devise the sweep.

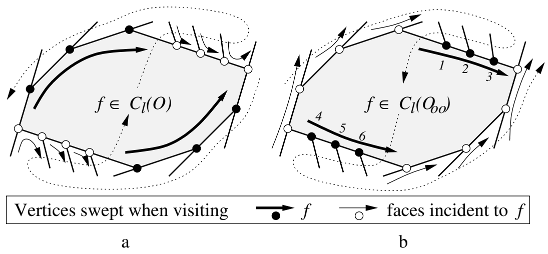

Figure 5: Sweeping vertices in increasing order **a.** in $\mathcal{C}_l(O)$, **b.** in $\mathcal{C}_l(O_\infty)$.

We first consider $\mathcal{C}_l(O)$, with $O \neq O_\infty$. Figure 5(a) shows a face f of $\mathcal{C}_l(O)$ and the order in which incident faces of f are visited (dotted arrows) in increasing order. We see that when we visit f, we can sweep neither its extremal vertices, nor vertices that separate sub-edges of its right-upper (resp. left-lower) fat edge: If we did, the local coherence of property (2) would not be satisfied. So we visit only the other incident vertices (black circles, bold arrows), those of upper edges

e = first upper edge of f
  while $e \notin$ right-upper fat edge
    sweep left vertex of e
    e = next upper edge

and those of lower edges

e = last lower edge of f
  while $e \notin$ left-lower fat edge
    sweep right vertex of e
    e = previous upper edge

as shown in figure 5(a).

We see that incident vertices that are not swept when visiting f are swept when visiting incident faces of f during the walk (white circles, normal arrows), and finally each vertex is swept exactly once (with the exception of the left (resp. right) vertex of the first (resp. last) face of the walk, but we just have to sweep them before (resp. after) the walk). We also see that vertices incident to f are swept correctly from left to right. By considering a dual curve and all successive faces incident to this curve, we can show that the sweep is also coherent globally.

Although this sweep is correct for $\mathcal{C}_l(O)$, it cannot be used for $\mathcal{C}_l(O_\infty)$: The face incident to the first (resp. last) subedge of the left-lower (resp. right-upper) fat edge of f may be a semi-infinite face of label (O_∞, O_∞), and since these faces are not visited during the walk, some vertices may not be swept.

So we perform the walk by visiting faces in decreasing order (dotted arrows in figure 5b). We see that this time only vertices subdividing the right-upper and the left-lower fat edges of f can be swept. We sweep them from left to right, by visiting first vertices incident to the right-upper fat edge, then those incident to the left-lower fat edge (black circles, bold arrows). This order is important: A dual curve may cut first the right-upper fat edge of f and cut next the left-lower fat edge of f. Other incident vertices of f are swept when visiting other faces (white circles, normal arrows). As in the previous algorithm, it can be shown that this sweep visits each vertex exactly once and is coherent globally.

This sweep is only valid for $\mathcal{C}_l(O_\infty)$ and cannot be used for $\mathcal{C}_l(O)$ when O is a line segment (p, q) of the scene: The first upper (resp last lower) edge of f may be supported by q^* (resp. p^*), and in this case faces incident to these edges would not belong to $\mathcal{C}_l(O)$ and vertices incident to these edges would not be visited.

Both algorithms have their counterpart for sweeping vertices in decreasing order: The walk is done in reverse order and vertices are swept in decreasing order. Since the walk is done in optimal time,

Proposition 3 *The vertices of $\mathcal{C}_l(O)$ can be swept topologically in increasing (resp. decreasing) order in optimal time proportional to the number of vertices swept.*

Figure 6 shows an exemple of sweep: It takes the walk of figure 4 and shows how vertices are swept during the walk. The order in which vertices are swept is similar to the order

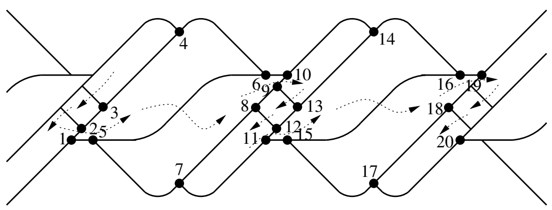

Figure 6: Sweeping vertices of $\mathcal{C}_l(O_\infty)$.

in which vertices of an arrangement of lines are swept with the algorithm of Overmars and Welzl [OW88]. This algorithm, a simplified version of the original algorithm of Edelsbrunner and Guibas [EG86], uses only one horizon tree and sweeps each time the leftmost upper vertex of the upper horizon tree.

Finally, we can notice that other combinations of walk/sweep produce partial sweeps that visit only interior vertices (i.e., vertices incident to three faces of $\mathcal{C}_l(O)$).

5 Walking in the zone of a line segment

We want now to compute the visibility polygon of a line segment $s = (p, q)$, oriented from p to q, which is outside the convex hull of the scene. We must handle the objects of the scene weakly visible from the right side of s.

We must therefore consider faces of the complex that contain rays issued from O_∞ and passing through s (from left to right). The set of rays passing through s is represented in the visibility complex by the zone comprised between the dual curves p^* and q^* of the extremities of s, that is, the zone located below q^* and above p^*. So we consider the subset of elements of the complex of $\mathcal{C}_l(O_\infty)$ that intersect this zone. We note this subset $\mathcal{C}_l(s)$ and call it the *zone of s*.

We can define a total order on faces of $\mathcal{C}_l(s)$ in the same way we did for faces of $\mathcal{C}_l(O)$. However, when searching the next and the previous face of a face f, we must now be carefull to stay in $\mathcal{C}_l(s)$: The next (resp. previous) face of f in $\mathcal{C}_l(O)$ may not belongs to $\mathcal{C}_l(s)$. To compute the upper edge e_{ns} incident to the next face $next_s(f)$ of f in $\mathcal{C}_l(s)$, we must do some checking (figure 7a):

e_n = edge incident to $next(f)$ in $\mathcal{C}_l(O)$
if e_n is above q^*
  then e_{ns} = upper edge of f cut by q^*
else if e_n is below p^*
  then e_{ns} = upper edge of f cut by p^*
else $e_{ns} = e_n$

We compute the lower edge e_{ps} incident to the previous face $prev_s(f)$ of f in $\mathcal{C}_l(s)$ similarly (figure 7b):

e_p = edge incident to $prev(f)$ in $\mathcal{C}_l(O)$
if e_p is above q^*
  then e_{ps} = lower edge of f cut by q^*
else if e_p is below p^*
  then e_{ps} = lower edge of f cut by p^*
else $e_{ps} = e_p$

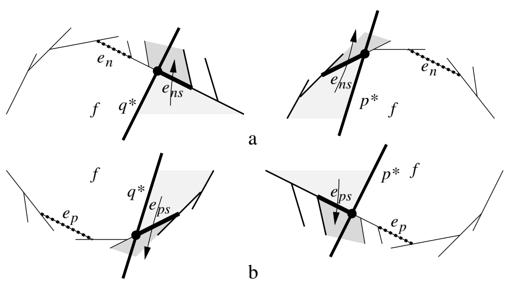

Figure 7: **a.** Computing $next_s(f)$. **b.** Computing $prev_s(f)$.

$\mathcal{C}_l(s)$ has a finite number of faces, and therefore has a first and a last face. More precisely, a face f does not have a next (resp. previous) face in $\mathcal{C}_l(s)$ either if it contains the vertex $v = p^* \cap q^*$ corresponding to the ray (p, q) (resp.

(q, p)), or if its next (resp. previous) face is a semi-infinite face (O_∞, O_∞).

If the supporting line of s does not cut the convex hull of the scene, then $v = p^* \cap q^*$ is in a semi-infinite face (O_∞, O_∞). In this case, the first face of $\mathcal{C}_l(s)$ is the face incident to the first edge cut by p^*, and the last face is the face incident to the last edge cut by q^* (figure 8 left). Both faces are incident to a semi-infinite face (O_∞, O_∞).

If the oriented supporting line (p, q) of s cuts the convex hull of the scene such that s is behind the scene, then the first face of $\mathcal{C}_l(s)$ is the face incident to the edge cut by p^* and to the lower semi-infinite face (O_∞, O_∞), and the last face is the face containing $v = p^* \cap q^*$ (figure 8 middle).

If the line (p, q) cuts the convex hull of the scene such that s is before the scene, then the first face of $\mathcal{C}_l(s)$ is the face containing $v = p^* \cap q^*$, and the last face is the face incident to the edge cut by q^* and incident to the upper semi-infinite face (O_∞, O_∞) (figure 8 right).

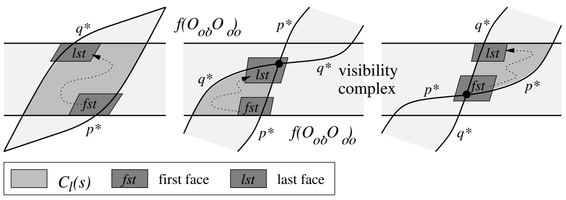

Figure 8: First and last faces of the zone of s.

Checking if e_n (resp. e_p) is below p^* or above q^* is done in constant time. The computation of all the edges cut by p^* (resp. q^*) is in fact the computation of the view around p (resp. q). These views can be computed with a sweep algorithm and need not be computed in advance: The edges cut by p^* (resp. q^*) can be computed one at a time when needed so that the computation of the walk and of the views are synchronized. So the sweep can be performed with no additional data structure, and

Proposition 4 *Given a line segment $s = (p, q)$ outside the convex hull of the scene, the n_f faces of $\mathcal{C}_l(s)$ can be visited in increasing (resp. decreasing) order in $O(\log n + n_f + t_v(p) + t_v(q))$ time, where $t_v(p)$ (resp. $t_v(q)$) is the time to compute the view around p (resp. q).*

The view around a point can be computed by two sweep algorithms respectively in $O(v \log n)$ time, v size of the view, and in $\Omega(v)$ and $O(nv)$ time where these bounds are tight (see [PV96] and [Riv97a]).

We show in the remaining sections how the walk in the zone of s can be used to compute the visibility polygon of s, and to maintain the view around a point moving from p to q.

6 Computing the visibility polygon of a line segment

The visibility polygon of a line segment $s = (p, q)$ is the set of points of objects of the scene that are visible from (at least) one point in s. Every two side of the polygon is a transversal side supported by an object of the scene. The other sides are radial sides that link transversal sides (figure 9). We consider here that s is outside the convex hull of the scene.

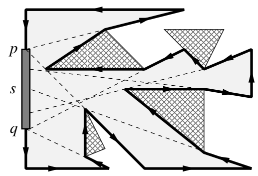

Figure 9: Visibility polygon of a line segment.

We compute the visibility polygon by sweeping its successive transversal sides. A transversal side s_1 of the visibility polygon supported by an object O_1 is the set of right extremities of rays of a face f_1, face of $\mathcal{C}_l(s)$ of label (O_∞, O_1). s_1 can be considered as the portion of O_1 lightened by the neon s, the rays lightening s_1 being those of f_1.

Let $f2$ be the next face of f_1 in $\mathcal{C}_l(s)$ ($f_2 = next_s(f_1)$), and let (O_∞, O_2) be its label. Then it can be shown that the next transversal side of the visibility polygon is the part of O_2 "lightened" by rays of f_2. Moreover, if e denotes the edge incident to f_1 and f_2, then the radial side linking s_1 to s_2 (when it is not reduced to a point) is supported by the ray whose dual point in the complex is either a vertex extremity of e, or the intersection point of e and p^* (resp. q^*).

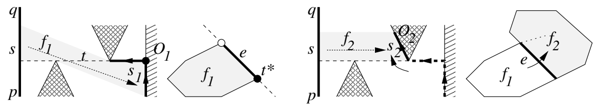

Figure 10: Sweep of the visibility polygon.

The walk in $\mathcal{C}_l(s)$ allows us to compute the visibility polygon of s easily:

Proposition 5 *The visibility polygon of $s = (p, q)$ can be computed in $O(vis(s) + t_v(p) + t_v(q))$ time, where $vis(s)$ is the size of the visibility polygon and $t_v(p)$ (resp. $t_v(q)$) the time to compute the view around p (resp. q).*

Although its complexity is low, this algorithm is only close to optimal: Some points seen by p (resp. q) may not correspond to a radial side of the visibility polygon. Notice also that the visibility polygon is not necessarily a simple polygon.

7 Maintaining the view around a moving point

We show in this section how to maintain the view around a point moving from p to q along a line segment $s = (p, q)$ which is outside the convex hull of the scene.

In [Riv97b] we have proposed two algorithms that maintain the view around p_v by processing visibility changes in their order of occurence. Here we use the walk in $\mathcal{C}_l(s)$ to process visibility events in topological order.

The view around a moving point p_v changes when p_v crosses an (extended) edge of the visibility graph. In the visibiliy complex, the view around p_v is the set of consecutive edges cut by the dual curve p_v^*. The view changes when the dual curve p_v^* sweeps a vertex of the complex. The vertices swept by p_v^* during a displacement of p_v must be processed in topological order so that the view around p_v can be updated in constant time at each visibility change.

We show here how to maintain the view when the supporting line of s does no intersect the convex hull of the scene (others cases are similar). After computing the view around p, we must sweep vertices of $\mathcal{C}(s)$ from p^* to q^*, that is in decreasing order. When we are in a face f, instead of computing directly the edge incident to the next face, we use the sweep of the vertices to find the next face. We first sweep sub-edges of the left-lower fat edge in decreasing order. If p^* cuts the fat edge, we start from the sub-edge cut by p^*, else we start from the last sub-edge of the fat edge. We sweep the previous sub-edges until encountering a sub-edge cut by q^* or the first sub-edge of the fat edge. Then we sweep the right-upper fat edge the same way, and the last sub-edge swept is the edge incident to the next face.

With this method, the view around q is not computed directly, but computed implicitly after all the updates of the view around p.

Proposition 6 *Let p_v be a point moving from p to q along the line segment $s = (p, q)$. After computing the view around p, the view around p_v can be maintained in total $O(\max(v(p), v(p, q)))$ time, where $v(p)$ is the size of the view around p and $v(p, q)$ the number of changes of visibility along s.*

We have previously presented in [Riv97b] two algorithms for maintaining the view around a point. The algorithm of this paper has the advantage over these two algorithms of updating the view in constant time (instead of $O(\log^2 v(p_v))$ (resp. $O(\log n)$) time) at each change of visibility. Moreover, it does not need data structures such as a dynamic convex hull or a priority queue and is in fact independant of the real trajectory of p_v between p and q.

If a program does not need to process visibility changes in temporal order, but only in topological order, then this algorithm improves the running time of the second algorithm in [Riv97b] (which needs an initialization in $O(v(p) \log v(p))$ time).

When used for consecutive moves, this algorithm has however the disadvantage over the first algorithm in [Riv97b] of visiting all faces of $\mathcal{C}(s)$, even if some of these faces do not yield a visibility change, and therefore runs in at least $O(v)$ time for each move.

8 Conclusion

We have shown how to sweep only some parts of the visibility complex in optimal time and without any additional data structure. Although we have studied in this paper walks in subsets of elements of the complex having the same left object in their label, all the algorithms have their counterpart for subsets of elements having the same right object in their label. All these algorithms appear to be relatively simple and have been or are being implemented.

We have also shown two applications of these walks: Computing the visibility polygon of a line segment and maintaining the view around a moving point.

Asano, Guibas, and Tokuyama [AGT94] have shown how to sweep some parts of an arrangement of lines in the plane without the arrangement being built. It would be interesting to see if our walks in the complex can also be performed without the complex being built.

We are also studying if these walk could be used in algorithms for maintaining the visibility complex upon insertion or deletion of polygons in a scene, to identify the elements of the complex that are modified by this insertion/deletion.

Finally, we have extended the algorithm for computing the visibility polygon of a line segment to line segments inside the scene. However work remains to be done to ameliorate its complexity.

References

[AGT94] Te. Asano, L. J. Guibas, and T. Tokuyama. Walking on an arrangement topologically. *Internat. J. Comput. Geom. Appl.*, 4:123–151, 1994.

[EG86] H. Edelsbrunner and L. J. Guibas. Topologically sweeping an arrangement. In *Proc. 18th Annu. ACM Sympos. Theory Comput.*, pages 389–403, 1986.

[ORDP96] R. Orti, S. Rivière, F. Durand, and C. Puech. Radiosity for dynamic scenes in flatland with the visibility complex. *Comput. Graph. Forum*, 15(3):237–248, 1996. Proc. Eurographics '96.

[OW88] M. H. Overmars and E. Welzl. New methods for computing visibility graphs. In *Proc. 4th Annu. ACM Sympos. Comput. Geom.*, pages 164–171, 1988.

[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.

[Riv97a] S. Rivière. *Visibility computations in 2D polygonal scenes*. Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 1997.

[Riv97b] Stéphane Rivière. Dynamic visibility in polygonal scenes with the visibility complex. In *Proc. 13th Annu. ACM Sympos. Comput. Geom. (Communication)*, 1997.