An Algorithm for Finding the Weakly Visible Faces from a Polygon in 3D*

Extended Abstract

Harry Plantinga
Department of Computer Science
University of Pittsburgh
Pittsburgh, PA 15260
planting@cs.pitt.edu

1. Introduction

A polygon is said to be *weakly visible* from a viewing polygon **P** in the presence of other polygons when some part of the front side of that polygon is visible from some point in **P**. For applications in computer graphics and virtual reality, it is desirable to determine the set of faces of a model or scene that are weakly visible from a polygon in a preprocessing phase. For example, a CAD model of a building may be divided into regions such as rooms, and weak visibility from each region may be computed in advance. The view from any viewpoint in a region may then be rendered correctly by rendering all of the polygons that are weakly visible from that region with graphics hardware that performs hidden surface removal. The set of weakly visible faces from a region is the smallest set of scene faces that results in the correct image under those conditions. This kind of visibility computation can bring about significant performance increases over rendering all of the faces of the model with hidden-surface-removal hardware for interactive graphics applications such as architectural walkthroughs, in which views must be rendered as quickly as possible and preprocessing time is available.

In this paper I present an algorithm for finding the set of weakly visible faces from a polygon **P** in 3-space under perspective projection. This is equivalent to determining the set of faces at least partially illuminated by a lighted polygon **P**. The polygon is assumed to be a one-sided light, so that only faces on one side of **P** are visible. For a scene with a total of n vertices among all faces including **P**, the algorithm has worst-case runtime $O(n^4 \log n)$. The algorithm works by finding the faces that are weakly visible from some viewpoint in **P** and those that become weakly visible somewhere in **P**. Extensions are also discussed to computing weak visibility from non-polygonal regions and from volumes.

Much work has been done on computing visibility from a point in 2-D [for example, O'Rourke, 1987; Sack and Suri, 1990] or in 3-D, which is closely related to hidden surface removal [see, for example, Foley *et al.*, 1990; Sutherland *et al.*, 1974]. Airey *et al.* [1990] compute visibility from a region for the graphics application of architectural walkthroughs, but the methods given are approximation algorithms: partial shadow computations and sampling by stabbing. They express the desire that better algorithms be devised. Funkhouser, *et al.* [1992] precompute visibility information for interactive walkthroughs, using a stabbing technique to determine which rooms are visible from a given room. Mulmuley [1991] and Bern *et al.* [1990] independently compute visibility along a line segment. Plantinga and Dyer [1990] present an algorithm for computing the *view-*

---

point space partition for a polyhedral scene, which is a partition of viewpoint space into maximal regions of viewpoints from which the topology of the image is constant.

2. Weak Visibility Events

Key to this algorithm is the computation of *weak visibility events*. I define a weak visibility event viewpoint to be a viewpoint v such that there is an object face f_v that is invisible (weakly visible) from v, but an arbitrarily small change of viewpoint in some direction causes f_v to become weakly visible (invisible). The idea is that a weak visibility event viewpoint is a viewpoint at which some part of a face of the scene appears or disappears. The definition follows in spirit the definition of visual event found elsewhere [e.g. Koenderink and van Doorn, 1979; Plantinga and Dyer, 1990; Plantinga, *et al.*, 1990], but it differs in that it concerns the weak visibility status of faces rather than changes in the topology of the image; the events defined in this paper are a proper subset of those representing a topological change in the image. The "extra" events that occur under those other definitions correspond to topological changes in the appearance of the image that are not changes the set of weakly visible faces. In this paper, I will speak of a face's being occluded at v and becoming weakly visible upon a small change in viewpoint, but the symmetric case is also intended.

A face in a scene only becomes invisible upon a small change in viewpoint when it "turns away from" the viewer or it becomes occluded by other faces in the scene. Given an event viewpoint v and a face that becomes occluded at that viewpoint, f_v, let S_f denote the set of points of f_v that appear upon an arbitrarily small change in viewpoint. Viewpoint v can be an event only if it lies in the plane containing f_v or if a line of sight from v to a point of S_f passes through edges of faces in the scene sufficient to constrain it in some direction with respect to maintaining the visibility status of f_v. This requires that it pass through two parallel edges or three edges of which no pair is parallel, possibly including one or two edges of f_v. If only two parallel edges participate in the event, then a line of sight from v to a different point of S_f can be found that passes through three object edges [Plantinga and Dyer, 1990]. These edges are said to participate in the event, and f_v is said to be the face occluded at the event viewpoint v. Examples of images of scenes from viewpoints that are near event viewpoints are shown in Figure 1.
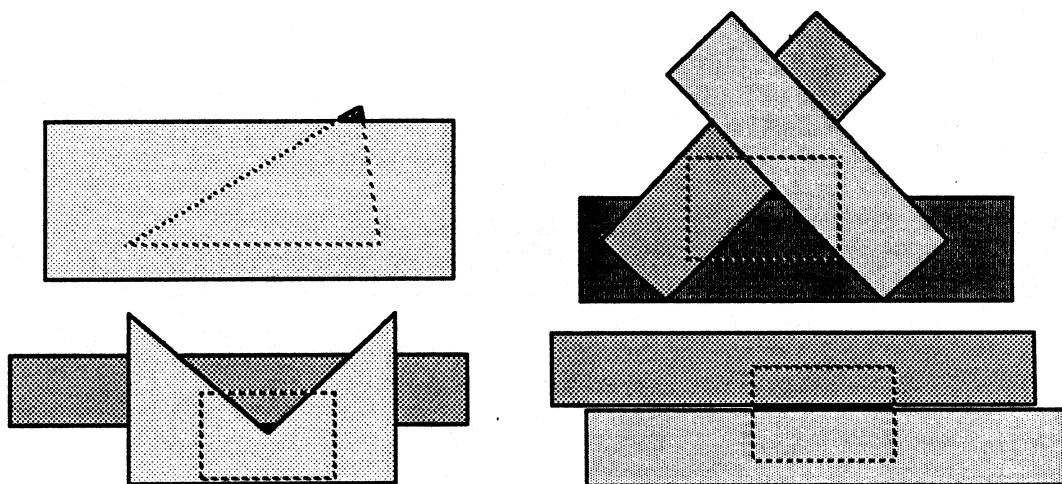

Figure 1. The image from viewpoints near events. Four different examples are shown.

Events can be classified as one of three types: horizon events, in which the viewpoint is in the plane containing the face that becomes invisible, edge-vertex (EV) events, in which the viewpoint is not in the plane containing the face that becomes invisible and two of the edges participating in the event meet at a vertex, and edge-edge-edge (EEE) events otherwise. Since EV events involve the apparent intersection of a vertex and an edge in the image, the lines of sight in question lie in the plane containing the vertex and the edge. In the case of EEE events, the lines of sight pass through three object edges (see Figure 2).
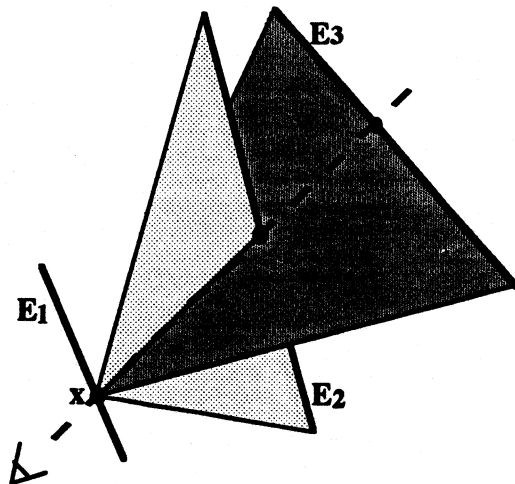

Figure 2. The line of sight through point P on E_1 that passes through E_2 and E_3.

Not all sets of three image edges result in an event, but all events require the apparent intersection of three image edges, that is, the intersection of the projection of three scene edges in the viewing plane. Arrangements of three scene edges that may result in an event can be found by examining the cases that result from different arrangements of faces containing the edges, whether two of the edges meet at a vertex, and whether the occluded face contains an edge or a vertex participating in the event.

In Figure 2, the dotted line is the line of sight that passes through the three edges shown. Note that this line of sight is the intersection of the two shaded planes: one containing a point x of E_1 and all of E_2, and the other containing x and E_3. If edge E_1 is given by $p_1 + s\, a_1$, E_2 by $p_2 + s_2\, a_2$, and E_3 by $p_3 + s_3\, a_3$, then for $x = p_1 + s\, a_1$ on E_1, direction of the line of sight is given by

$$d = [(p_1 + s\, a_1 - p_2) \times a_2] \times [(p_1 + s\, a_1 - p_3) \times a_3] \qquad (1)$$

Notice that d is a quadratic function of s. Thus, the locus of event viewpoints for an EEE event is not a plane but a warped surface. I will define *potential event points* as points that are on lines of sight that start at a point v of x and intersect the edges participating in the event and the occluded face f_v. Potential event points are event points when the line of sight between v and f_v is not obstructed by any other face.

3. Finding Weakly Visible Faces

The algorithm for finding the weakly visible faces from **P** first finds all of the faces visible from some point in **P** using a standard object-space hidden-surface removal algorithm such as that of Mulmuley [1989]. It then finds all of the events in **P** and adds any faces

that become weakly visible at some event in **P** to the list of weakly visible faces. Any face that is weakly visible from **P** must be visible from every point in **P** or must become weakly visible at an event in **P**, so all weakly visible faces are thus found.

The algorithm requires that the locus of potential event points in space for a potential event be represented. Since these points are viewpoints at which a face becomes weakly visible in the absence of other polygons, for horizon events the locus of points in space corresponding to a potential event is a polygonal part of the plane containing the face that becomes occluded. For EV events, it is a polygonal part of the plane containing the edge and vertex participating in the event. For EEE events, it is a part of a warped surface consisting of the points on the lines of sight that start at an event point in **P**, pass through the three edges participating in the event, and reach the face that becomes occluded. In this case, the locus of potential event viewpoints is a part of the warped surface bounded by line segments: the intersection of the warped surface with **P**, the intersection with the face that becomes occluded, and a segment from each of two bounding lines of sight.

Since the surfaces are planar or warped, they can be represented by parametric functions of the endpoints of line segments constituting the surfaces. Thus, each surface is represented as the union of the line segments from $p_1(s)$ to $p_2(s)$ for all s, $0 \le s \le 1$. Here, $p_1(s)$ represents the locus of viewpoints in **P** intersecting the potential event surface and $p_2(s)$ represents corresponding points of $f_{p_1(s)}$ that become occluded. For EV events, p_1 and p_2 will be linear (or piece-wise linear) functions of s, and for EEE events they will be quadratic functions of s. The algorithm represents a subset of these lines of sight by representing an *active subrange* of the parameter s for an event. (These active subranges will represent the lines of sight that are not occluded by other faces of the scene.) A subrange consists of many non-intersecting intervals of [0,1]. These subranges can be maintained in a data structure such as a segment tree [Preparata and Shamos, 1985]. A new interval may thus be subtracted from the subrange in $O(\log n)$ time.

For horizon events, $p_1(s)$ is part of the intersection of **P** with the plane containing the face that becomes occluded, and $p_2(s)$ is the set of edges of that face visible from **P**. For EV events, the lines of sight in question must intersect **P**, the vertex and edge participating in the event, and the face that becomes occluded. $p_1(s)$ is the intersection of the lines of sight that meet these conditions with **P**, and $p_2(s)$ is the intersection of the lines of sight that meet these conditions with the face that becomes occluded (see Figure 3).
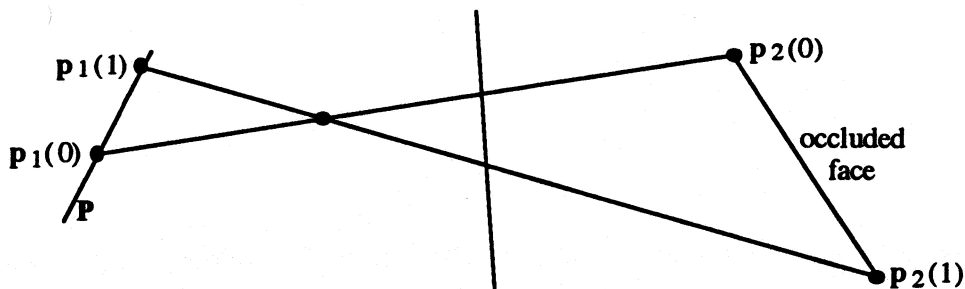

Figure 3. Finding $p_1(s)$ and $p_2(s)$ for EV events.

For EEE events, the lines of sight in question start at a point of **P** and intersect the three edges participating in the event. The lines of sight intersecting the three edges are found using Eq. 1. Initially, all of the lines of sight that pass through edge E_1 and the lines containing E_2 and E_3 (see Figure 2) are represented by Eq. 1 and the parametric equation for E_1, with $0 \le s \le 1$. The values of s for which the lines of sight intersect E_2 and E_3 are then computed using Eq. 1. The "active" values of s are stored in a segment tree. The

surface defined by the active subrange of s, Eq. 1, and the parametric equation for E_1 define a warped surface; the intersection of this surface with P is then found. The result is the potential event surface. Finally, $p_1(s), 0 \le s \le 1$, is the intersection of this surface with P, and $p_2(s)$ represents the corresponding points of the faces that become occluded.

The algorithm for finding weakly visible faces is as follows. In the algorithm, faces marked "invisible" are not known to be weakly visible, but faces marked "visible" are known to be weakly visible. At the end of the algorithm, all faces still marked "invisible" are known not to be weakly visible from P.

1. Since faces are only visible from one side of P, intersect all faces with the halfspace on the correct side of P. This does not increase the number of vertices of the scene.

2. Mark all remaining faces "invisible."

3. Select a point of P (e.g. a vertex). Perform hidden-surface removal from that point, assuming a hemispherical field of view. Mark all of the polygons that appear in the image "visible."

4. Find all horizon and EV potential event surfaces. For each one,

> If the face occluded at that potential event does not contain an edge participating in the event, find the faces occluded at that event by exhaustive search. If all of the faces occluded at that event are marked "visible," skip to the next potential event surface.
>
> Find the endpoint functions $p_1(s)$ and $p_2(s)$ for the surface. Represent the active subrange of s in a segment tree (initially [0,1]). Find the subrange of s corresponding to the lines of sight that intersect P.
>
> Subtract from the segment tree the subrange of s corresponding to the intersection of the surface with every other scene face. For average-case efficiency, the subtractions should be done in order of decreasing size of scene face.
>
> If any lines of sight remain, mark the polygons occluded at this event "visible."

5. Repeat step 4 for all EEE potential events.

A potential event surface is intersected with a polygon by finding the intersection of the surface with the plane containing the polygon. This will be a series of segments on a line or a quadratic curve. The line or quadratic curve is then intersected with the polygon; once the intersection points are found, it is possible to find the intersections of the segments with the polygon in a manner similar to a merge: traverse the line or curve, segment by segment, keeping track of whether you are outside or inside the polygon. The procedure may be accomplished in $O(r + e)$ time where the potential event surface is divided into r subranges and the polygon has e edges.

4. Time Complexity

Steps 1 and 2 can be completed in $O(n)$ time where n is the number of vertices or edges in the scene. Step 3 can be completed in $O(n^2)$ time using a hidden-surface removal algorithm such as that of McKenna [1987]. In step 4, an operation is done for each horizon and EV potential event. There are $O(n)$ horizon events since there is at most one per scene face, and there are $O(n^2)$ EV potential events since there is at most one per scene edge/vertex pair. The number of endpoints for a subrange cannot be larger than the num-

ber of edges in the scene, so the subrange intersection problem for a face with m vertices can be completed in time $O(m \log n)$. Intersecting the potential event surface with sub-ranges for all faces thus takes time $O(n \log n)$, and completing step 4 takes time $O(n^3 \log n)$. The step 5 time requirements are identical except that there are $O(n^3)$ EEE potential events (at most one for every set of three edges of the scene), so the time required to complete step 5 is $O(n^4 \log n)$. Thus, the whole algorithm requires time $O(n^4 \log n)$.

For realistic scenes, the runtime is likely to be much less that the worst case, though still possibly very high. Some reasons are that for realistic scenes, most EV and EEE poten-tial events can be eliminated from consideration by intersecting the potential event sur-face with a single large polygon from the scene, and many tests of potential events will be obviated because the occluded face was already known to be visible in step 3 or step 4 or the potential event surface does not intersect **P**. The worst case occurs only for scenes that are visually very complex (such as a grid in front of another grid), and it typically oc-curs for scenes in which only a few faces are visible from any viewpoint in **P** but most faces are visible from some viewpoint in **P**.

The average-case performance of this algorithm may be improved by many techniques, which generally vary in efficacy according to the specific characteristics of the scene in question. For example, one technique that would significantly improve average-case per-formance for scenes with large occluding faces such as walls and floors in a building is to find the faces that can easily be determined to be invisible from **P** in a preprocessing stage. This stage would proceed as follows: for each large face, the shadow region with respect to **P** considered as a light source is found. Any other face lying completely in that shadow is eliminated from consideration.

5. Extensions and Conclusion

An algorithm is presented here for computing the set of weakly visible faces from a poly-gon in 3D. It is interesting that the worst-case runtime for this algorithm is so much worse than that for hidden-surface removal from a point, which can be accomplished in $\Theta(n^2)$ time. An obvious open question is whether the worst-case runtime can be im-proved. It is possible to prove that the weak visibility problem from a point in 3-D re-quires $\Omega(n \log n)$ time by reduction from element uniqueness. In fact, it seems unlikely that it is possible to improve the worst-case runtime for the weak visibility from a poly-gon problem to $o(n^3)$, because of the existence of $\Theta(n^3)$ event boundaries in any region in the worst case. However, it may be possible to improve on the $O(n^4 \log n)$ algorithm given here. Also, an algorithm with good expected-time performance for the kinds of scenes generally occurring in computer graphics and virtual reality would be highly desirable.

Finding the set of weakly visible faces from a polygon is an expensive computation. The problem statement may be relaxed slightly to the following: given a polygon, find a small superset of the weakly visible faces from that polygon. An efficient algorithm for that problem would also be useful for computer graphics, especially if it could be done much more quickly, because the correct scene could be rendered with hidden surface re-moval hardware. An approach to that problem is presented elsewhere [Plantinga, 1992].

This algorithm may also be used for the weak visibility problem from a planar region R that is not a polygon. To do so, it must be possible to find and represent the intersection of R with the surfaces arising in this algorithm. Also, this algorithm can be used to com-pute weak visibility from a polyhedral volume V. In this case, the faces that are weakly

visible from V are those that intersect V or that are weakly visible from the one of the faces of V.

References

Airey, J. M., J. H. Rohlf, and F. P. Brooks, Jr., "Towards image realism with interactive update rates in complex virtual building environments," *Computer Graphics* **24**(2), 1990, pp. 41-50.

Bern, M., D. Dobkin, D. Eppstein, and R. Grossman, "Visibility with a moving point of view," SODA 1990, pp. 107-117.

Foley, J. D., A. van Dam, S. K. Feiner, and J. F. Hughs, *Computer Graphics: Principles and Practice*, 2nd ed., Addison-Wesley, 1990.

Funkhouser, T. A., C. H. Séquin, and S. J. Teller, "Management of large amounts of data in interactive building walkthroughs," *ACM Symp. on Interactive 3D Graphics*, 1992, pp. 11-20.

Koenderink, J. and A. van Doorn, "The internal representation of solid shape with respect to vision," *Biol. Cybernetics* **32**, 1979, pp. 211-216.

McKenna, M., "Worst-case optimal hidden surface removal," *ACM Transactions on Graphics* **6**, 1987, pp. 19-28.

Mulmuley, K., "An efficient algorithm for hidden surface removal," *Computer Graphics* **23** (3), 1989, pp. 379-388.

Mulmuley, K., "Hidden surface removal with respect to a moving view point (Extended abstract)," *Proceedings of the 23rd. Annual Symp. on Theory of Computing*, 1991, pp. 512-522.

O'Rourke, J, *Art Gallery Theorems and Algorithms*, International Series of Monographs on Computer Science, Oxford University Press, 1987.

Plantinga, H., and C. R. Dyer, "Visibility, occlusion, and the aspect graph," *International Journal of Computer Vision* **5**(2), November 1990, pp. 137-160

Plantinga, H, C. R. Dyer, and W. B. Seales, "Real-time hidden-line elimination for a rotating polyhedral scene using the aspect representation," *Proc. Graphics Interface '90*, pp. 9-16.

Plantinga, H., "Conservative Visibility Preprocessing for Efficient Walkthroughs of 3D Scenes," TR 92-4, Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260.

Preparata, F. P, and M. I Shamos, *Computational Geometry*, Springer-Verlag, 1985.

Sack and Suri, "An optimal algorithm for detecting weak visibility of a polygon," *IEEE Transactions on Computers*, 1990.

Sutherland, I. E., R. F. Sproull, R. A. Schumacker, "A characterization of ten hidden-surface algorithms," *ACM Computing Surveys* **6**(1), 1974, pp. 1-55.