Computer Graphics (Proceedings SIGGRAPH '94) pp. 443--450

Global Visibility Algorithms for Illumination Computations

Seth Teller
Institute of Computer Science
Hebrew University of Jerusalem

Pat Hanrahan
Department of Computer Science
Princeton University

Abstract

The most expensive geometric operation in image synthesis is visibility determination. Classically this is solved with hidden surface removal algorithms that render only the parts of the scene visible from a point. Global illumination calculations, however, may require information between any two points in the scene. This paper describes global visibility algorithms that preprocess polygon databases in order to accelerate visibility determination during illumination calculations. These algorithms are sensitive to the output complexity in visibility space; that is, how many pairs of objects are mutually visible. Furthermore, the algorithms are incremental so that they work well with progressive refinement and hierarchical methods of image synthesis. The algorithms are conservative, but exact; that is, when they return visibility predicates they can be proved true. However sometimes they do not return either totally visible or totally invisible, but partially visible, even though in the same situation a better algorithm might return the exact answer. In this paper we describe the algorithms and their implementation, and show that, in a scene with low average visual complexity, they can dramatically accelerate conventional radiosity programs.

CR Categories and Subject Descriptors: I.3.5 [Computational Geometry and Object Modeling]: *Geometric Algorithms, Languages, and Systems*; I.3.7 [Computer Graphics]: *Three-Dimensional Graphics and Realism – Radiosity*; J.2 [Physical Sciences and Engineering]: *Engineering*.

Additional Key Words: Hidden surface removal, visibility space, radiosity, global illumination, algorithmic triage.

1 Introduction

In the early days of image synthesis a central geometric problem was hidden surface removal. With the advent of z-buffering, modern workstations can display pictures of 3D scenes containing millions of polygons in real-time. However, such workstations have limited shading capabilities because they make the assumption that all light sources illuminate every object. One major thrust of current research in image synthesis is to remove this restriction so that the shading correctly accounts for the illumination incident on every object. To do this every surface element must assess what light sources, or more generally, what surfaces reflecting light towards it, are visible to it. This type of illumination calculation is termed *global*, in contrast to local, because the entire scene must be analyzed to determine if there are any occluders interfering with the transfer of light between objects. Collating such visibility information is more difficult than determining merely what is visible from a single vantage point, as is done in hidden surface removal. For example, the fastest algorithm currently known for computing a complete description of the interocclusion due to a polyhedral object of n vertices can take $O(n^6 \lg n)$ time [9].

This paper describes global visibility algorithms that analyze the entire visibility space, and are applicable to a range of illumination problems. Here, we apply them to a hierarchical radiosity algorithm. We have implemented several practical algorithms, and show that they allow efficient global visibility calculations for scenes of low visual complexity. The algorithms are based on three simple ideas:

Visibility preprocessing. To compute what is visible from all points on the surfaces of the objects being shaded, we preprocess the scene to speed future visibility tests. For the purposes of global illumination we need only consider all pairwise interactions between objects. Preprocessing removes totally invisible pairs from consideration, and accelerates later queries regarding visibility between points on partially visible pairs.

Incremental visibility maintenance. The most efficient global illumination algorithms operate iteratively based on error criteria. Examples are hierarchical radiosity, where surfaces are subdivided with respect to each other according to potential light transfers between them [11], and progressive refinement methods where light is transferred among surfaces in order of brightness [5]. Thus, the visibility algorithms should be lazy and sensitive to required precision. They should also allow refinement so that more precise determinations can be made as needed.

Conservative triage. Both the preprocessing and maintenance methods use conservative triage to avoid the combinatorial complexity of exact visibility determination. We classify visibility into three categories: totally INVISIBLE, totally VISIBLE, and PARTIAL (partially visible). The classification is conservative in that all interactions classified as INVISIBLE or VISIBLE are correct; however, it is acceptable for the classification to return PARTIAL when the correct result is either VISIBLE or INVISIBLE. This allows us to forego complex analysis or "punt" if such analysis will take too long to determine the exact answer. Of course, for this to work we need either another visibility algorithm to complete the analysis, or we must expect the situation to simplify eventually (e.g., through subdivision).

The visibility algorithms presented here generalize previous work on preprocessing environments for interactive walkthroughs. In [24], an algorithm was given to preprocess a 2D environment of axial line segments, such as floorplans. This was extended to 3D axial rectangles in [22]. This paper treats the case of convex polygons in general position.

The global visibility algorithms described here have been implemented with a global illumination system that computes radiosity values for polygonal scenes [11]. The algorithm maintains a hierarchy of interactions between subdivided polygons at different levels of detail. A key feature of the algorithm is that only $O(k^2 + n)$ interactions are ever examined (with k the number of input polygons, and n the number of elements created by subdividing those polygons). The hierarchical radiosity algorithm, as originally designed, used pairwise visibility information between polygons. In the original implementation, however, this visibility information was *inexact*. Visibility status was determined by shooting a constant number of rays between two polygons. If all of the rays reached from one polygon to the other, the polygons were considered totally visible, whereas if none of the rays reached, the polygons were considered totally invisible. The conservative algorithms described in this paper, in contrast, are provably more precise.

2 Overview

We present novel algorithms that subdivide space, construct a *conservative visibility graph* over the polygons in a geometric model, then maintain the correctness of the graph under recursive subdivision of the polygons. In the context of the hierarchical radiosity computation, this conservative visibility graph guarantees that throughout the computation, all polygons that potentially interact (e.g., exchange energy) will be known. The construction and maintenance of the graph occurs in four stages.

1. Spatial Subdivision. The geometric model is first *spatially subdivided* into convex polyhedral cells, linked across shared boundaries only when some *portal*, or transparent region, exists on the boundary. For a large class of models, and particularly for architectural models, the subdivision proves a natural way of hierarchically capturing the geometric and occlusive characteristics of the model.

2. Visibility Propagation. Each cell of the spatial subdivision encloses some portion of the geometric model. Clearly, only when two *cells* are mutually visible can their contents (i.e., model polygons) interact. Consequently, we hierarchically enumerate all visibility between portions of the model by first establishing *inter-cell* visibility, then establishing *inter-polygon* visibility only where cells are mutually visible. This is accomplished by *propagating* incremental visibility information through the cells of the spatial subdivision; as each cell ''sees'' into increasingly distant cells, the visibility graph is augmented to record any previously unknown interactions. (Portal enumeration is simply the first and crudest record of visibility propagation.)

Visibility propagation provably discovers all partially or totally visible cell (polygon) pairs, at the cost of occasionally misclassifying an invisible cell (polygon) pair as visible. The alternative, misclassification of some mutually visible interaction as invisible, is plainly unacceptable, since it may omit from consideration an interaction later to prove important.

3. Blocker Detection. In an exacting illumination computation such as global illumination, it is not sufficient to determine simply that two polygons are partially visible; some estimation must be made of the extent to which they are visible, as well as how much error might be incurred by the estimation. Therefore, once potential visibility between a pair of polygons is established, a set of *interfering polygons* or *blockers* is determined that may occlude part of one polygon as seen from some point on the other. This interference computation is again *conservative*; a non-interfering polygon may occasionally be classified as a blocker, but a blocker will *never* be classified as non-interfering. In the visibility graph, *blocker lists* augment existing links between mutually visible polygons; total visibility is established whenever the blocker list is empty. Perhaps surprisingly, we show that these conservative overestimated blocker lists are generally *smaller* than those maintained by existing algorithms.

4. Blocker Maintenance. In a hierarchical radiosity algorithm, polygons (*patches*, in radiosity parlance) are allowed to exchange radiant energy only when the interaction satisfies some specified global error bound [11]. Otherwise, the patches are subdivided, and interaction is recommenced among the child patches. It is natural to consider how the conservative visibility graph among the patches can be *incrementally maintained* under subdivision. Each child patch may be partially or totally visible, or completely invisible, to its child counterparts on the other polygon. We show how, given the parent interaction, conservative blocker lists for the children can be determined incrementally. We present a novel blocker maintenance technique involving *linespace*, a five-dimensional representation of 3D lines (i.e., light rays).

The algorithms we present are of interest in several ways. First, they comprise a practical treatment of visibility issues for unrestricted (i.e., non-axial) three-dimensional environments, in contrast to previous work [1, 8, 24]. Second, the conservative visibility description we compute − identification of all mutually visible pairs, and the blocker set for each pair − is a natural, output-sensitive way of characterizing visibility among polygons or more general objects, for any algorithms that require information about occlusion and/or illumination. Finally, we show that the use of these algorithms dramatically improves the time and space efficiency of an existing radiosity computation [11].

3 Spatial Subdivision

The geometric model is specified as a set P of convex polygons (Figure 1). The space embedding the geometric model is subdivided into convex polyhedral *cells*, typically separated by polygons (Figure 2). The construction is based on BSP trees [7], but the visibility algorithms we subsequently present are provably correct for any spatial subdivision satisfying a few geometric criteria [22].

Figure 1: A geometric model comprised of polygons.

First, a polyhedral *root volume* is constructed as the convex hull of P. While polygons of sufficient size are present, a polygon is chosen whose support plane is to partition the remainder of the set. The choice is made using a simple heuristic that determines the polygon whose cross-section in the current cell is largest, when expressed as a fraction of the cell's areal intersection with the polygon's support plane; if any polygons separate the cell into mutually invisible parts, one such polygon is chosen. This heuristic tends to yield effective splitting trees in practice.

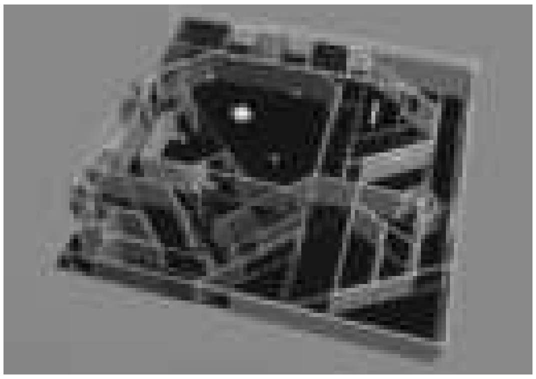


Figure 2: Subdivision of the model into cells and portals.

Next, the *portals*, or transparent portions, of each cell boundary are explicitly constructed. Since cell boundaries are induced on the support planes of polygons, these boundaries are typically partially or completely obscured. Each boundary stores a list of coaffine and incident polygons. The portals on each boundary comprise a convex decomposition of the set difference of the cell boundary with the union of these polygons. Each portal stores identifiers for the cells which it connects. The spatial subdivision therefore comprises an *adjacency graph* over the cells, since two cells are adjacent in this graph iff they share a boundary that is not completely opaque.

4 Visibility Propagation

Once a conforming spatial subdivision is built, visibility propagation commences. The propagation algorithm operates in object space, and performs a *constrained traversal* of the adjacency graph outward from each *source* cell. Whenever a cell is reached by this traversal, its associated polygons are examined pairwise with those in the source for mutual visibility; unreached entities are definitely invisible from the source.

A given cell can see into its neighbors only through portals, and into more distant cells only through *portal sequences*; i.e., ordered lists of portals such that each consecutive pair of portals lead into and out of the same cell. The cell adjacency graph is searched by determining cells between which an unobstructed *sightline* exists. A sightline must be disjoint from any occluders and thus must intersect, or *stab*, a portal in order to pass from one cell to the next. To establish inter-cell visibility, it is sufficient to find a stabbing line through a particular portal sequence; since if some point in the *interior* of one cell can see a point in the interior of another, a sightline must exist between the boundaries of the source cell and the reached cell.

Thus, the problem of finding sightlines between cell interiors reduces to finding sightlines through portal sequences of increasing length. Consequently, the primitive visibility operation in a conforming spatial subdivision is the determination of a stabbing line, given a portal sequence, or the determination that no such stabbing line exists. The portal sequences are generated incrementally by a depth-first search (DFS) emanating from a particular cell boundary; when a sequence no longer admits a sightline, the active branch of the DFS terminates.

4.1 Inter-Cell Visibility

Sightline determination is an *existence* predicate, in that it merely establishes visibility between two points on different cells. Suppose two cells are mutually visible through a portal sequence. In general, only a portion of each cell is visible to the other, due to occlusion by opaque material abutting the edges of intervening portals (Figures 3, 4). Whenever inter-cell visibility is established, mutually visible *volumes* are constructed for the cell pair; these volumes and the reaching portal sequence are then used to determine inter-polygon visibility among the cells' associated polygons.

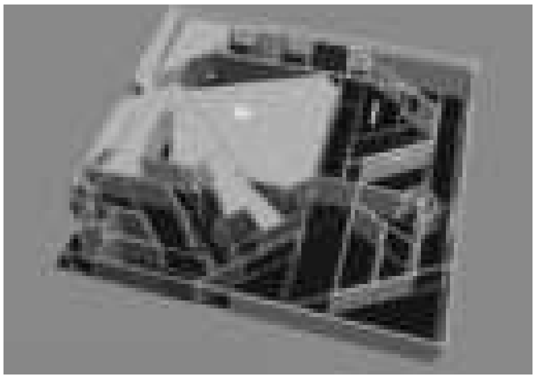


Figure 3: Visibility propagation from a source cell S_1.

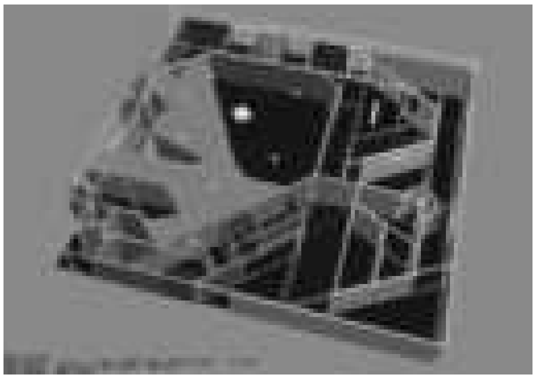


Figure 4: Visibility propagation from a source cell S_2.

The volume visible to a polygon in the presence of polygonal occluders is, in general, bounded by quadratic surfaces [18]. An algorithm for computing this volume was implemented and described in [21], but is not yet sufficiently robust for use on complex models. Consequently, we have developed a simpler algorithm that computes a polyhedral volume guaranteed to enclose the exact visible region. The algorithm is a straightforward construction that, using separating tangent planes, performs a kind of internal pivoting over the edges and vertices occurring along the portal sequence. We treat the algorithm briefly here; details can be found in [22].

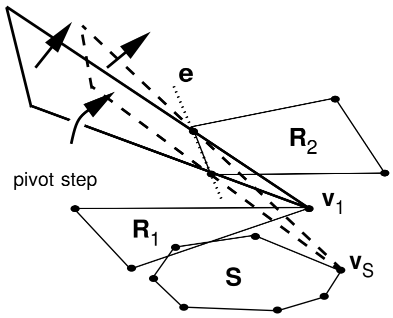


Figure 5: A pivoting step on edge $\mathbf{e}$, from $\mathbf{v}_1$ to $\mathbf{v}_S$.

The algorithm exploits the fact that for each portal edge, at most two separating planes can contribute a face to polyhedral bounds on the illuminated volume (Figure 5), since at most one vertex from each halfspace of the associated portal can span a relevant plane with the edge. Consider some edge $\mathbf{e}$ on a portal $\mathbf{R}_2$, and the portals occurring before $\mathbf{R}_2$. Each of these portals has at most one *extremal* vertex that spans a separating plane with $\mathbf{e}$ (in the figure, $\mathbf{R}_1$ has extremal vertex $\mathbf{v}_1$ and $\mathbf{S}$ has extremal vertex $\mathbf{v}_S$). Together with $\mathbf{e}$, only one of these (at most $\frac{n}{3}$) extremal vertices can span a plane that contains all the other extremal vertices in the same halfspace as the portal $\mathbf{R}_2$. This single plane is the only one of the n candidate planes that can contribute faces to the boundary of the illuminated volume. Therefore, for any n portal edges there are at most $2n$ boundary planes, each of which can be identified in $O(n)$ time by pivoting over the vertices of the other portals. The total time to identify the $2n$ relevant planes is therefore $O(n^2)$. Moreover, the set of $O(n)$ planes can be updated incrementally whenever a new portal is encountered, simply by updating the existing halfspaces with respect to the new portal vertices, and introducing planes tight on the new portal edges. The $O(n)$ positive halfspaces of the planes are inspected for an intersection with the c BSP halfspaces bounding the reached cell in time $O(n + c)$ with a linear programming algorithm [15, 19]. If no such intersection exists then the reached cell can not be visible to the source through the active portal sequence.

4.2 Inter-Polygon Visibility

Whenever a cell is reached by the graph propagation, an *active set* of halfspaces bounds the volume in the reached cell visible to the source. The orientations of each of these halfspaces are reversed to bound the volume illuminated by the *reached* cell in the *source*. Only polygons in these respective volumes can be mutually visible. Each incident source polygon is prefixed to the front of the active portal sequence (Figure 6). The visible volume in the reached cell due to the augmented sequence is then tested for incidence with the appropriate subset of polygons stored in the reached cell. (The notion of conservative inter-polygon visibility can be simply extended to treat visibility between general objects [22].)

Figure 6 depicts this mechanism for an analogous 2D situation, in which ''polygons'' and ''portals'' are line segments. A source cell S (Figure 6-i) establishes inter-cell visibility to a cell R via some portal sequence. The polygon B in S can have no interaction with R's interior, and it is not considered further. Polygon C is incident on the inter-cell visibility volume, and therefore potentially visible from some point in S. However, when the portal sequence is augmented with the constraints due to A (Figure 6-ii), polygon C is found to be invisible from A. Finally, D is found to intersect A's visible region in R, and A and D are established to be mutually

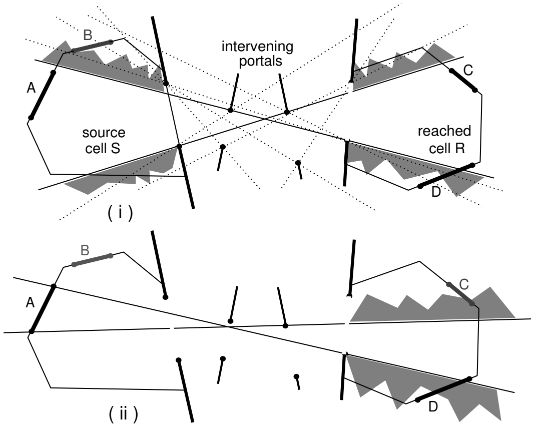


Figure 6: Establishing visibility of 2D ''polygons'' A and D.

visible.

The convexity of spatial subdivision cells allows an important optimization. Any two polygons entirely incident on the boundaries of the same cell can have blockers only in the relative interior of that cell. When the cell interior is empty (as it typically will be), the polygons can be immediately classified as entirely mutually visible. Thus, the spatial subdivision quickly identifies many instances of complete mutual visibility between nearby polygons.

Figures 7 and 8 depict the output of the inter-polygon visibility computation in three dimensions, for two polygons incident on different source cells. Display of the spatial subdivision has been suppressed for clarity.

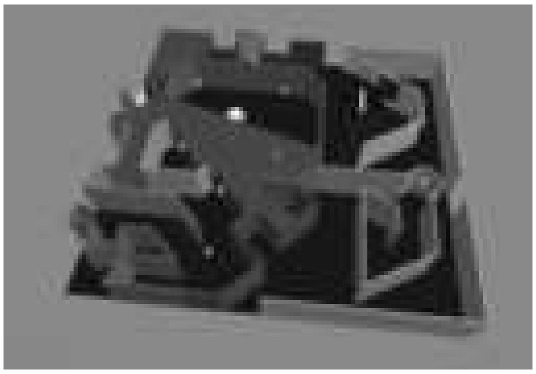


Figure 7: Visibility propagation from a polygon in cell S_1.

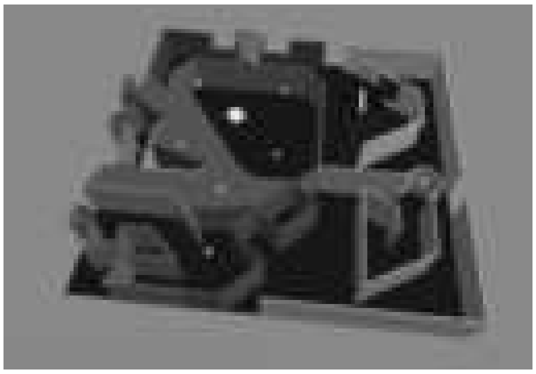


Figure 8: Visibility propagation from a polygon in cell S_2.

5 Blocker Detection

When a pair (S, R) of polygons is found to be mutually visible, we record a visibility *interaction* $I(S, R)$, and proceed to identify the *blocker list* $B(S, R)$ of the pair. One could simply compute the set of blockers as those polygons incident on a convex volume containing S and R (as in [10]). However, the visibility graph and reaching portal sequence generally yield a better (i.e., smaller) blocker list. Denote the convex hull of all vertices of S and R as $conv(S, R)$. Clearly any blocker B must be incident on $conv(S, R)$ to contribute to $B(S, R)$. Moreover, observe that *only polygons visible to S along a sequence reaching R, or to R along a sequence reaching S* need be considered as blockers of S and R. For, if some polygon C is *not* visible to S, then *every* ray leaving S (including those rays to any point on R) must stab some polygon other than C before stabbing C.

The polygons S and R do not generally *see* the same set of blockers (Figures 7 and 8). Therefore, $B(S, R)$ is augmented whenever a search from S (R) to R (S) discovers a previously unknown blocker. Figure 9 depicts the result of the blocker computation, where all polygons except S, R, and $B(S, R)$ have been removed. Note that, of the polygons from the large central room, neither the large blue interior wall panels nor the thin blue doorjambs (cf. Figure 1) are classified as blockers. Thus the purely spatial (shaft) cull produces a blocker list of size 12 or more, whereas the blocker detection algorithm presented here computes a list of 6 blockers.

Figure 9: The final blocker list of S and R.

Finally, the blocker criterion presented above is conservative, since $B(S, R)$ may include polygons that are visible to S or to R but do not affect occlusion between them. The *exact* determination of the blocker list is computationally involved; a polygon B is a blocker of S and R only if some ray from S to R exists whose only front-facing polygon intersection, aside from that with R, is with B. (The asymmetry of the definition arises from the fact that, in a manifold polyhedral environment of oriented polygons, only front faces can be visible to front faces.)

6 Blocker Maintenance

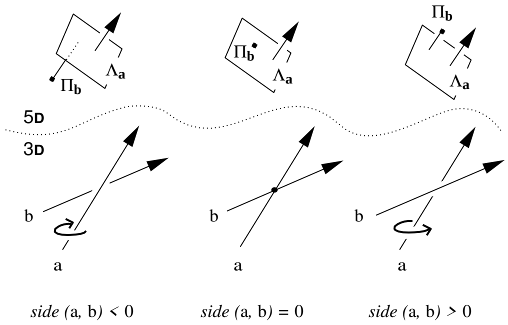
Given a set of polygons P, the visibility preprocessing scheme produces, for every polygon $S \in P$, a set of visible polygons $V(S)$. For each polygon $R \in V(S)$, the *blocker list* $B(S, R)$ enumerates all polygons $B \in P \setminus \{S, R\}$ that potentially impede visibility between S and R. $B(S, R)$ points only to top-level patches; this makes sense, since blockers should be as large as possible to cause maximal occlusion. For each interaction $I(S, R)$, we store a *tube* data structure, which associates an interacting patch pair, a blocker list, the *visibility status* $V(S, R)$ of the interaction (i.e., VISIBLE or PARTIAL), and some additional geometric information used for incremental visibility tests.

In the hierarchical radiosity algorithm, when the energetic interaction between two patches can not be characterized to within the global error bound, one of the patches of the interaction is symmetrically *subdivided*, and its children are allowed to interact with the other patch [11]. Clearly, interactions between either patch and the children of its counterpart are highly coherent. The tube data structure exploits this coherence to perform efficient and accurate visibility reclassification after subdivision.

Each child interaction's blocker list is necessarily a subset of the parent's blocker list; we wish to efficiently, and incrementally, determine the child tube's blockers. We say that a blocker B *impinges* on $I(S, R)$ if it occludes S from R, and that B is *disjoint* from $I(S, R)$ if B can not cause occlusion. Whenever a blocker list is discovered to be *empty* (i.e., to contain no impinging blockers), complete visibility between the interacting patches will be established, and no further visibility computations need be done for any children of this interaction. Conversely, whenever the blocker list is discovered to be *completely occluding*, there can be no energy transport between S and R, and the interaction is discarded (alternatively, the culprit blocker(s) can be retained as ''proof'' that the patches cannot interact). Finally, when neither complete visibility nor complete occlusion can be quickly determined, the status of the child interaction remains partially visible.

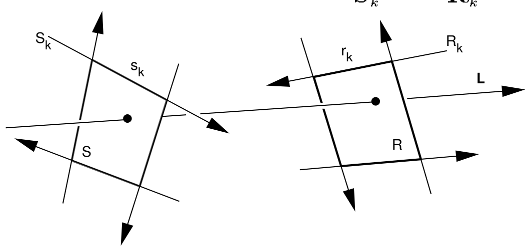
6.1 Linespace

The tube structure efficiently encodes the set of all lines between S and R, using a five-dimensional line representation known as Plucker coordinates [20], or simply *linespace*. Lines in three dimensions correspond to hyperplanes and points in linespace. Any two 3D rays a and b can be *oriented* by considering their linespace counterparts Λ_a, a 5D hyperplane, and Π_b, a 5D point (details of the mapping can be found in [21]).



The signed distance of Π_b from Λ_a determines the sense in which the lines ''go around'' each other in 3D; if Π_b lies on Λ_a the lines a and b are coplanar. This ''sidedness'' property can be used to represent the set of lines through a collection of convex polygons. In practice, there is one caveat to using the linespace representation [23]. The only portion of linespace corresponding to 3D lines with real coefficients are those linespace points lying on a 4D manifold known as the Plucker quadric [20]; all other linespace points correspond to 3D lines with complex coefficients. Fortunately, the algorithms used in this paper need never consider the Plucker quadric, since they manipulate only lines known *a priori* to have real coefficients.

Consider two convex polygons S and R, comprised of sets of oriented edges S_k and R_k, respectively. For there to exist some line $\mathbf{L}$ that stabs the interiors of S and R, $\Pi_{\mathbf{L}}$ must lie in the appropriate signed halfspaces h_k of the hyperplanes $\Lambda_{\mathbf{S}_k}$ and $\Lambda_{\mathbf{R}_k}$.



Thus, the set of all lines through S and R corresponds to the interior of a five-dimensional convex polytope $\cap_k h_k$ [21]. Rather than attempt to compute this polytope directly, we can manipulate the vertices of its intersection with the Plucker quadric, which are comparatively easy to generate. Each such vertex corresponds to a collection of four support lines from S and R, since four 5D hyperplanes must intersect with the Plucker quadric to generate each such vertex. These vertices must correspond to stabbing lines tight on four edges of S and R in 3D; i.e., lines through a vertex of S and a vertex of R (note that these lines necessarily have real 3D coefficients). In our implementation, there are at most sixteen such lines, since all patches are quadrilaterals.

There are several advantages to performing blocker analysis in linespace. The data structure for a single blocker is constant size, and for a single patch interaction is linear in the number of blockers. The linespace analysis obviates complicated 3D topological and numerical computations. The only operations required by the linespace representation are mapping from 3D lines to 5D points and hyperplanes, and computing inner products between points and hyperplanes.

6.2 Incremental Blocker Maintenance

The tube data structure, and incremental visibility maintenance, can now be fully described. Suppose patch R is subdivided against patch S into child elements $C(R) \subset R$. The tube for S and each $C_R \in C(R)$ stores S, C_R, and a constant number of linespace points $\Pi(S, R)$ whose convex hull $conv(\Pi(S, R))$ includes the set of all lines through S and C_R. Finally, each blocker in $B(S, R)$ is reclassified with respect to the child tube to produce $B(S, C_R)$, and the visibility status $V(S, C_R)$ of each interaction $I(S, C_R)$ is determined. As before, many instances of total invisibility, partial visibility, and total visibility are discovered quickly. Other situations are considered too complex to analyze completely, and we ''punt'' and classify the interaction as partially visible (perhaps causing further subdivision [11]).

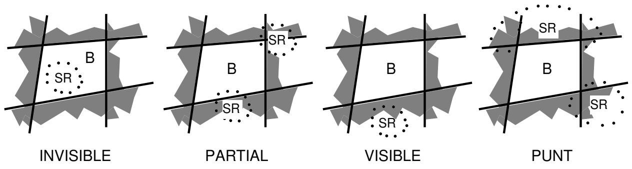


Figure 10: Performing 3D triage in 5D linespace.

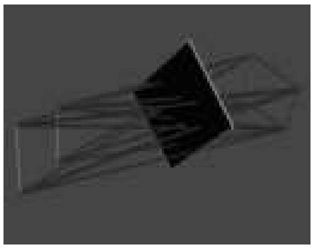
Consider an interaction (S, R) and a single potential blocker B (Figure 10). We wish to determine, without extensive analysis, whether all, none, or some of the lines through S and R stab the blocker B. Respectively, this is equivalent to determining whether *conv*$(\Pi(S, R))$ lies entirely inside, is disjoint from, or has some intersection with $\cap_k \Lambda_k(B)$, the set of lines through the blocker (Figure 11). We exploit the fact that, in linespace, both sets of lines are convex.

The points in $\Pi(S, R)$ are first classified with respect to the blocker hyperplanes $\Lambda_k(B)$. If all of the points lie inside the $\Lambda_k(B)$, then $conv(\Pi(S, R)) \subset \cap_k \Lambda_k(B)$, by convexity. B is therefore completely occluding and $V(S, R)$ is INVISIBLE. If some of the points lie inside the $\Lambda_k(B)$, and some lie outside, some lines through S and R stab B, and $V(S, R)$ is PARTIAL. If *all* of the points lie outside some *single* $\Lambda_k(B)$, $V(S, R)$ is VISIBLE. Finally, the complex case occurs when *all* of the points lie outside *all* of the $\Lambda_k(B)$. This does not guarantee total visibility, since $conv(\Pi(S, R))$ may still have some intersection with $\cap_k \Lambda_k(B)$ (this case is labeled PUNT in Figures 10 and 11); accordingly, $V(S, R)$ is classified as PARTIAL.

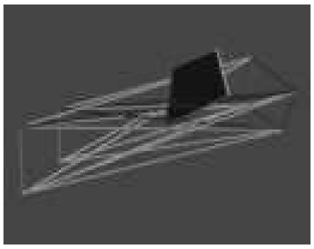
The logic for multiple blockers is straightforward; any single blocker can cause $V(S, R)$ to be INVISIBLE, but all blockers must be disjoint in order for $V(S, R)$ to be VISIBLE. Otherwise, any impinging blocker causes $V(S, R)$ to become PARTIAL.

6.3 Evolution

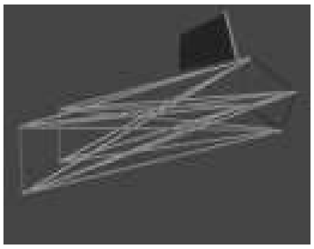
Figure 12 depicts an example of blocker list evolution and incremental reclassification of child interactions. White lines connecting quadrilateral centroids represent VISIBLE interactions; green lines represent PARTIAL interactions, and red lines represent the tube



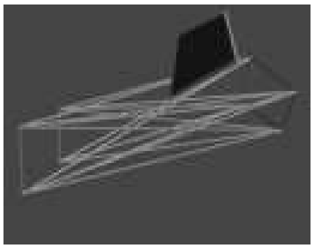
INVISIBLE



PARTIAL



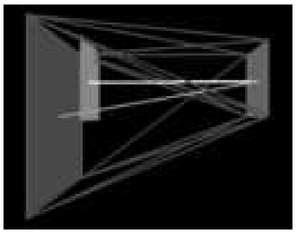
VISIBLE



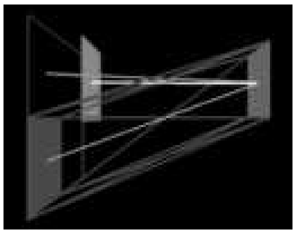
PUNT

Figure 11: The four possible outcomes of a blocker classification.

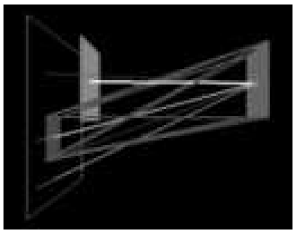
between two interacting polygons. (INVISIBLE interactions are not shown.) In Figure 12-i, two red polygons interact via a blocker list containing the orange polygon. In Figure 12-ii, the large red polygon is subdivided into quadrants, and its child interactions with the small red polygon are shown. One of these interactions is INVISIBLE. The other three are PARTIAL; the tube for one of them is shown. Finally, in Figure 12-iii, the child polygon is subdivided; three of *its* children become VISIBLE to the polygon at right, but one (shown) remains PARTIAL.



(i) Parent interaction.



(ii) PARTIAL interaction.



(iii) VISIBLE interaction.

Figure 12: Reclassification of child interactions after subdivision.

The linespace algorithms guarantee conservative visibility, in that blockers are only discarded from interactions if they are definitely known to be disjoint. Existing algorithms use point-sampling [2, 6, 11, 16] or point-to-area visibility [3, 4] techniques and therefore do not guarantee correct inter-area visibility determination. In contrast, we establish exact visibility information where possible, and adaptively subdivide until the uncertainty of visibility estimation in the remaining cases is so small as to be unimportant.

The linespace blocker maintenance algorithms are simple and fast, although they sometimes overestimate occlusion by classifying disjoint blockers as impinging, and may not identify INVISIBLE interactions as early as might a more sophisticated algorithm. Establishment of improved algorithms for the determination of inter-polygon visibility in the presence of multiple blockers is an active area of research [17, 21, 25]. An exact algorithm was presented in [21], but is not yet sufficiently robust for application here.

The work and storage expended for the incremental visibility maintenance also serves to accelerate the sampling done to establish inter-patch energy transfer (i.e., to estimate form factors). The ray/blocker machinery is simply applied to random sample rays (as used in [11]). The cost of each ray/blocker test is four 5D inner products.

7 Results

7.1 Spatial Subdivision

We implemented the spatial subdivision, propagation, interference, and maintenance algorithms described, and instrumented their execution for the data set shown in Figure 1. All execution was on a lightly loaded 50-MIP Silicon Graphics VGX. The input model comprised 403 patches. Constructing the BSP tree required about thirty CPU seconds; the resulting tree contained 220 leaf cells. Each cell had 5.99 boundary faces, and 5.87 patches coaffine with some boundary face, on average (thus the spatial subdivision heuristics produced fairly local partitioning behavior). Another seven CPU seconds were absorbed by cell neighbor-finding and enumeration of the 525 portals between leaf cells of the subdivision.

7.2 Visibility Propagation

Computing inter-cell and inter-patch visibility for the model absorbed five CPU minutes. Of the $81003 = (403 \times 402)/2$ patch pairs in the model, 4,391, or 5.4%, were classified as mutually visible. (Thus preprocessing obviated nearly 95% of the potential patch-patch interactions). Of these 4,391 patch pairs, 2,814 (64.1%) were partially visible, and 1,577 (35.9%) were totally visible. Each patch saw, on average, 22 other patches.

The inter-cell traversals performed 10,128 stabbing tests of portal sequences, or about 46 tests per cell. The inter-polygon traversals performed 16,055 further incremental stabbing tests, one for each successfully reached cell and potentially visible patch pair. Thus, about 40 tests per patch were required to establish inter-patch visibility for all patches. The average length of a tested inter-cell portal sequence was just over 5 portals. This is consistent with our experience using a ten-thousand polygon, five-thousand cell axial model, in which the average portal sequence length was less than ten [8, 22]. A histogram of observed portal sequence lengths and stabbing percentages is shown in Figure 13.

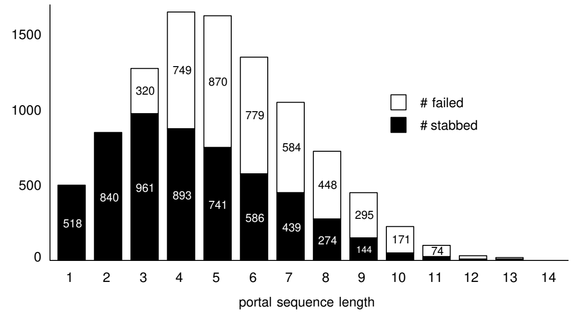
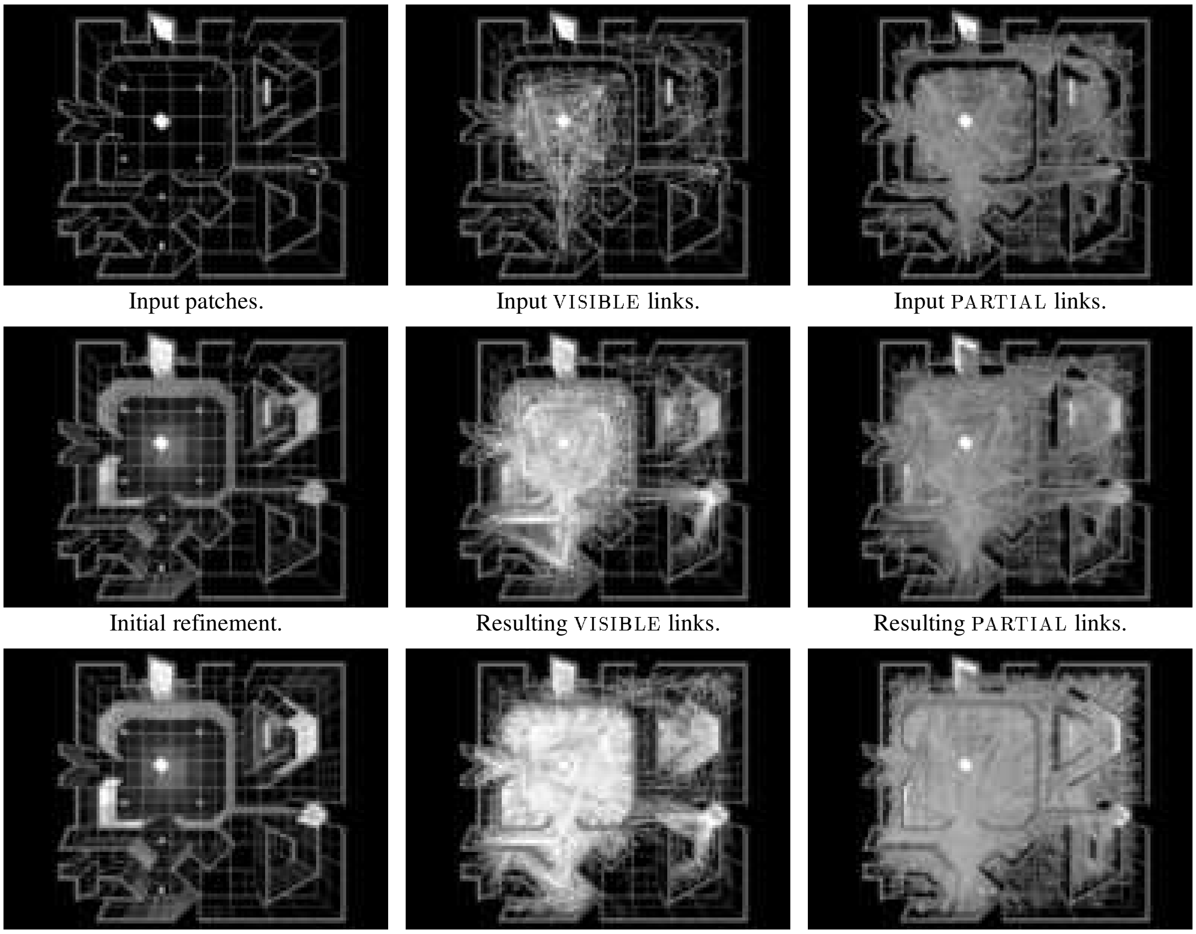


Figure 13: Stabbing successes and failures, by sequence length.

The inter-cell visibility determination uses a depth-first-search through the cell adjacency graph, applying an incremental stabbing predicate and visible volume computation at each step. The incremental operation expends linear time in the number of portals currently in the sequence, assuming a constant number of edges per portal, and so requires $O(n^2)$ time to stab a sequence of n portals. In practice, this seems not to prohibit use of the algorithms on real data sets, since most portal sequences are short (less than ten portals), and the algorithmic constants are therefore more important than the asymptotic complexity measure.



Input patches. Input VISIBLE links. Input PARTIAL links.

Initial refinement. Resulting VISIBLE links. Resulting PARTIAL links.

Second refinement. Resulting VISIBLE links. Resulting PARTIAL links.

Figure 14: Refinement of the input patches (left), VISIBLE links (middle), and PARTIAL links (right).

7.3 Blocker Detection

There were 5 blockers between partially visible patches, on average, reached through portal sequences of average length five. The subdivision heuristic was effective; the BSP tree did not suffer from excessive ''free-space'' splitting, or regions in which subdivision planes were induced due to far away polygons.

The visibility analysis communicated its results to the radiosity computation via an ASCII file. Each file line recorded an interaction $I(S, R)$ between two polygons, the length of the associated blocker list $B(S, R)$, and the blockers themselves. A zero-length blocker list implied total visibility between S and R, i.e., $V(S, R) =$VISIBLE; otherwise the visibility status was PARTIAL.

7.4 Blocker Maintenance

The model input to the radiosity computation is shown at the upper left of Figure 14. The input patches form the radiosity program's initial mesh. The 4,391 initial VISIBLE and PARTIAL links are shown, respectively, in white (middle column) and green (right column). Two iterations of patch-patch refinement were performed. The resulting model mesh, PARTIAL and VISIBLE interactions are displayed in the second and third rows of Figure 14. The number of VISIBLE links drastically increases after the first iteration. Their increased density naturally indicates unoccluded regions of the model. Similarly, the green PARTIAL links indicate occlusion. INVISIBLE links are not shown, as they were discarded by the radiosity program upon detection.

Using the results of the visibility preprocessing, the initial refine took only 11 seconds, performing 145,846 interactions. The second refinement stage required 50 seconds, and performed 186,703 interactions. About 90% of the refined interactions were VISIBLE, thus requiring no sampling for form-factor estimation. Table 1 charts the evolution of each link type, the number of elements, and the number of interactions at each refine.

Input Links	Refine	Links	Refine	Links
V 1,577 (35.9%)	170,440 (87.6%)	V 176,474	218,420 (87.6%)	V 225,763
	V 6,034 (3.1%)		V 7,343 (2.9%)	
P 2,814 (64.1%)	P 16,889 (8.7%)	P 16,889	P 22,157 (8.9%)	P 22,157
	I 1,096 (0.6%)		I 1,339 (0.5%)	
4,391	194,459	193,363	249,259	247,920
403 patches		15,039 patches		17,347 patches
4,391 interactions		145,846 interactions		186,703 interactions

Table 1: Link evolution by type, with patch and interaction counts.

Since the time complexity of the radiosity algorithm is proportional to the number of interactions, the visibility preprocessing significantly decreased the computation done by the radiosity algorithm. Moreover, the modified radiosity algorithm was more accurate, since no partially visible interactions were missed due to sampling errors (as in [11]).

7.5 Blocker Visualization

All of the algorithms described in this paper were implemented using visualization tools that allowed interactive inspection of complex data structures. Figures 15 through 17 depict the use of this tool to investigate some interesting PARTIAL interactions. Again, the white and green line segments represent VISIBLE and PARTIAL interactions, respectively; for a particular partial interaction in each figure, the tube is shown (in red), and the blockers for the interaction

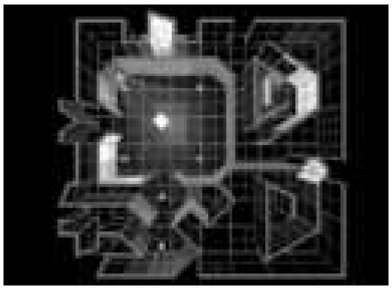


Figure 15: The tube data structure (red, orange) for an ordinary PARTIAL interaction.

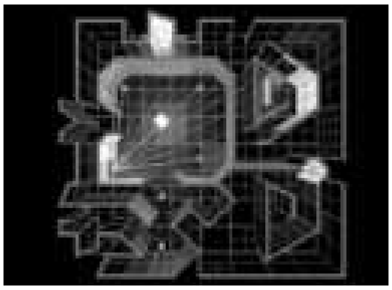


Figure 16: Spatially incident polygons that have not been classified as blockers.

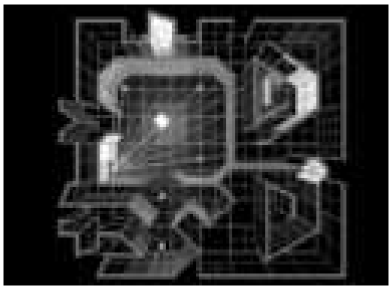


Figure 17: A PARTIAL interaction that could be classified as INVISIBLE.

are highlighted in orange. Figure 15 depicts an ordinary PARTIAL interaction. Figure 16 depicts spatially incident polygons that (correctly) have not been classified as blockers. Figure 17 depicts a PARTIAL interaction for which no single blocker occludes the source and receiver; a more sophisticated algorithm could classify this interaction as INVISIBLE.

8 Summary and Conclusion

We have presented several novel algorithms that represent an effective application of global visibility analysis to radiosity computations, an important problem in image synthesis. Given the complexity of both the visibility and radiosity approaches used, it was surprisingly easy to couple the two processes. We did so using an abstraction in which interactions between polygons were maintained along with all potentially blocking polygons. We argue that, for an interesting class of large models, inter-polygon visibility has roughly constant complexity throughout the interior of the model. After construction of a spatial subdivision for the model, the visibility algorithms we present are output sensitive; they expend work proportional to the amount of inter-polygon visibility present.

None of the visibility algorithms attempt to compute exact visibility information. However, they achieve precision in a different sense, by reporting all visibilities conservatively; potentially visible interactions are always reported.

Only blockers can occlude a specified source from a specified emitter. Thus, the blocker list formulation is applicable to the problem of discontinuity meshing in the presence of area light sources [12, 13, 14, 21], as well as to the construction of an ‘‘oracle’’ to decide which, if any, among a collection of discontinuities should be meshed upon earliest.

We showed that the visibility analysis significantly accelerated a radiosity computation in a polygonal environment. Finally, we demonstrated the successful application of some elegant concepts such as linespace and algorithmic triage to the concrete problem of construction and incremental maintenance of blocker lists.

Acknowledgments

The authors are grateful to David Laur for his assistance with the geometric model and the color plates, and to Dani Lischinski for his valuable comments. This work was begun during a visit to the NSF Science and Technology Center for the Visualization of Geometric Structures, in Minneapolis, and partially supported by Apple, Silicon Graphics Computer Systems, and the National Science Foundation (CCR 9207966).

References

[1] Airey, J. M. *Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations*. PhD thesis, UNC Chapel Hill, 1990.

[2] Baum, D. R., Mann, S., Smith, K. P., and Winget, J. M. Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 51–60.

[3] Campbell III, A., and Fussell, D. S. An Analytic Approach to Illumination with Area Light Sources. Tech. Rep. TR-91-25, Department of Computer Sciences, UT Austin, 1991.

[4] Chin, N., and Feiner, S. Fast Object-Precision Shadow Generation for Area Light Sources Using BSP Trees. In *Proc. 1992 Symposium on Interactive 3D Graphics* (1992), pp. 21–30.

[5] Cohen, M. F., Chen, S. E., Wallace, J. R., and Greenberg, D. P. A Progressive Refinement Approach to Fast Radiosity Image Generation. *Computer Graphics (Proc. SIGGRAPH '88) 22*, 4 (1988), 75–84.

[6] Cohen, M. F., and Greenberg, D. P. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics (Proc. SIGGRAPH '85) 19*, 3 (1985), 31–40.

[7] Fuchs, H., Kedem, Z., and Naylor, B. On visible surface generation by a priori tree structures. *Computer Graphics (Proc. SIGGRAPH '80) 14*, 3 (1980), 124–133.

[8] Funkhouser, T. A., Séquin, C. H., and Teller, S. Management of Large Amounts of Data in Interactive Building Walkthroughs. In *Proc. 1992 Workshop on Interactive 3D Graphics* (1992), pp. 11–20.

[9] Gigus, Z., Canny, J., and Seidel, R. Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence 13*, 6 (1991), 542–551.

[10] Haines, E. A., and Wallace, J. R. Shaft Culling for Efficient Ray-Traced Radiosity. In *Proc. 2^{nd} Eurographics Workshop on Rendering* (May 1991).

[11] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 197–206.

[12] Heckbert, P. S. *Simulating Global Illumination Using Adaptive Meshing*. PhD thesis, Computer Sciences Department, UC Berkeley, June 1991.

[13] Lischinski, D., Tampieri, F., and Greenberg, D. P. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12*, 6 (1992), 25–39.

[14] Lischinski, D., Tampieri, F., and Greenberg, D. P. Combining Hierarchical Radiosity and Discontinuity Meshing. *Computer Graphics (Proc. SIGGRAPH '93) 27* (1993).

[15] Megiddo, N. Linear programming in linear time when the dimension is fixed. *Journal of the ACM 31* (1984), 114–127.

[16] Nishita, T., and Nakamae, E. Half-Tone Representation of 3-D Objects Illuminated by Area Sources or Polyhedron Sources. In *Proc. IEEE COMPSAC, 1983* (1983), pp. 237–242.

[17] Plantinga, H. An algorithm for finding the weakly visible faces from a polygon in 3D. Tech. Rep. 92–11, U of Pittsburgh, 1992.

[18] Plantinga, W., and Dyer, C. An algorithm for constructing the aspect graph. In *Proc. 27^{th} Annual IEEE Symposium on Foundations of Computer Science* (1986), pp. 123–131.

[19] Seidel, R. Small-dimensional linear programming and convex hulls made easy. *Discrete and Computational Geometry* (1991), 423–434.

[20] Sommerville, D. *Analytical Geometry of Three Dimensions*. Cambridge University Press, 1959.

[21] Teller, S. Computing the Antipenumbra Cast by an Area Light Source. *Computer Graphics (Proc. SIGGRAPH '92) 26*, 2 (1992), 139–148.

[22] Teller, S. *Visibility Computations in Densely Occluded Polyhedral Environments*. PhD thesis, CS Dept., UC Berkeley, 1992.

[23] Teller, S., and Hohmeyer, M. E. Computing the Lines Piercing Four Lines. Tech. Rep. UCB/CSD 91/665, Computer Science Department, UC Berkeley, 1991.

[24] Teller, S., and Séquin, C. H. Visibility Preprocessing for Interactive Walkthroughs. *Computer Graphics (Proc. SIGGRAPH '91) 25*, 4 (1991), 61–69.

[25] Zhao, J., and Dobkin, D. Personal communication, 1992.