Guaranteed Occlusion and Visibility in Cluster Hierarchical Radiosity

Luc Leblanc Pierre Poulin
Département d'informatique et de recherche opérationnelle
Université de Montréal

Abstract.
Visibility determination is the most expensive task in cluster hierarchical radiosity. Guaranteed full occlusion and full visibility can reduce these computations without causing visibility errors. We build a hierarchy of large convex occluders using face clustering. This structure is efficiently exploited to avoid computing visibility between mutually fully occluded scene elements. Used in conjunction with a full visibility culling method, we show improvements on several scenes.

1 Introduction

Cluster hierarchical radiosity [26, 23, 24, 12, 2] possesses attractive theoretical and practical advantages to solve the problem of global illumination. Recently, adapted multiresolution surfaces [34, 6] have been introduced to better control the quantity and quality of energy transfers between surfaces rather than individual polygons or self-occluding volume clusters. While these improvements are essential to provide an increased realism for complex scenes at reduced computation cost, the major cost in these algorithms remains determining the visibility between pairs of elements, whether they are patches (subdivided or not), face clusters, or volume clusters.

In these hierarchical approaches, we can greatly benefit if we can *guarantee efficiently* at the highest possible level of the hierarchy that a pair of elements is mutually fully occluded or fully visible. Then no visibility computations would be required for any pair of children elements derived from these two parent elements. Guaranteed visibility and occlusion ensures no errors are introduced in the energy exchanges. We observed that full visibility and full occlusion can represent a significant proportion of visibility between pairs of elements in a scene. For instance in our test scenes, full visibility varies between 20 and 60% and full occlusion between 40 and 80%.

Guaranteed full visibility has been successfully exploited before by testing a shaft [13] enclosing the pair of elements against bounding volumes of the scene hierarchy. Testing full occlusion of these visibility shafts is more difficult. To be efficient, occluders should be as large as possible. Unfortunately objects in a scene can have complex concave shapes, resulting in difficulties to efficiently guarantee occlusion. Moreover in many scenes, objects are so finely meshed that no single polygon can be considered a good occluder. Algorithms handling the combined occlusion of several polygons have appeared in recent years [27, 10, 21], but they often require an important preprocessing step and a large amount of memory to store the corresponding structures.

In this paper, we fit a set of large convex polygons to handle efficiently the occlusion due to a large mesh of small polygons. Face clustering is first used to determine flat sections in surfaces. Then a small set of large convex polygons (our occluders) are fitted to each flat face cluster to efficiently represent the occlusion. The user can control the

minimum size and the number of the occluders. The full occlusion query proceeds by testing the rays bounding the visibility shaft against our convex occluders, themselves organized in a hierarchical structure.

Our results demonstrate that even with a limited number of occluders, we can find a large proportion (40 to 93%) of the full occlusions in several tested scenes. When full visibility is used in conjuntion with full occlusion, we observe reduction of the total rendering time (up to 11 times in our test scenes). Moreover even in scenes with little full occlusions, the cost of testing full occlusion is small enough that it almost never seems to penalize the visibility computations, and if it does, it will be negligible.

Although our visibility determination method is applied here to cluster hierarchical radiosity, its generality indicates it should also provide interesting results in walkthroughs [4, 16] and direct illumination from extended light sources.

In the next section, we briefly classify visibility techniques in the context of radiosity. In section 3, we describe how large convex occluders are extracted from the face clustering of the scene surfaces. Then we present the particularities of our cluster hierarchical radiosity system, before explaining in section 5 how full occlusion and full visibility are implemented in our system. The system has been tested with a wide range of scenes, and section 6 analyses the results. Finally we conclude and present directions for future improvements.

2 Related Work on Visibility Determination for Radiosity

Visibility determination is one of the fundamental problems in computer graphics, and an extensive literature has addressed various aspects of this problem. Many techniques are based on a single viewpoint, and therefore less directly applicable to specific visibility determination between pairs of elements in radiosity algorithms. Due to space constraints, we refer the interested reader to a more comprehensive survey such as the Ph.D. thesis of Durand [7].

We roughly divide the methods into sampled visibility, volumetric visibility, analytical visibility, portals, and occlusion culling. A large class of visibility methods are based on sampled visibility. The most popular of these methods simply shoots a number of rays between the pair of elements [32, 14]. They can prove efficient and their observed accuracy can usually be increased by using more samples, at the additional cost of increased computation time. However they suffer from the lack of bounds on the error of the estimated visibility, leading to potential light or shadow leakages in difficult configurations.

Volumetric visibility [22, 24] associates a density for a volumetric representation of surfaces, and the occlusion is approximated as an attenuation factor. While it provides a general sense of occlusions, the method assumes a uniform distribution of small objects, and therefore fails to accurately represent occlusions that are in nature more directional.

Analytical 3D visibility methods [20, 5, 29, 9] provide the complete and accurate solution to visibility. Unfortunately their algorithmic complexity quickly becomes a serious concern for any practical use in large scenes. Even though some extensions [8] make them more practical, the robustness problems of these methods remain.

Portals [30] offer major gains between elements distributed in separated rooms. If the information about portals is not included into the scene design, methods exist to automatically construct such portals [30, 17]. However portals become inefficient outside typical architectural scenes, within a single room, or as the rooms mutual visibility increases.

Occlusion culling based on a single large convex occluder [19, 4, 15] can eliminate

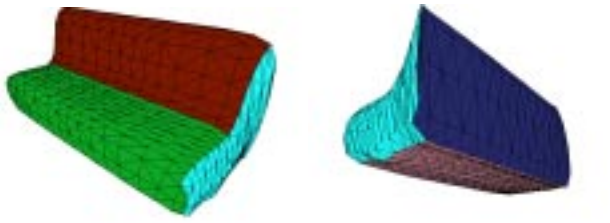


Fig. 1. Two views of a given level of face clustering applied on a sofa

an important portion of an entire scene in walkthroughs. However one must provide a list of such occluders and this list should depend on the positions of the pair of elements. The preprocessing of visibility using cells [31, 3, 33] allows to identify large occluders with respect to the current cell. However when the scene is mostly constituted of a soup of small polygons, they cannot fuse together these occluders to form a large occluder, and the expected gains are quite reduced.

Recent methods have been presented to extract large occluders from polygonal models. One approach uses simplification mesh techniques preserving occlusion [16]. The resulting occluders are not convex, which is essential in our case to efficiently guarantee full occlusion. Another approach grows octree boxes within closed geometric models [1]. This algorithm is restricted to closed models, which is not always guaranteed in many geometric modelers. Our extraction technique is similar to this one, but is based on a surface rather than a volume. We believe both solutions are useful and should be used in conjunction to identify the best occluders. In fact, these two occluder types can be integrated in our occlusion culling algorithm described in section 5. The next section details our extraction algorithm.

3 Extraction of Large Convex Occluders

Our occlusion culling structure relies on the presence of large occluders in the scene. Unfortunately for many scenes, the only information available is a list of a large number of small polygons, or if we are lucky, a list of objects, each constituted of its list of polygons.

As a preprocessing step, we first need to build a list of "good" convex occluders. To create these occluders, we use face clustering [11] to identify face clusters (a group of adjacent polygons) that are flat and as large as possible. These polygons are projected onto the supporting plane of the face cluster and large rectangles are fitted onto this projection to produce our large occluders.

3.1 Face Clustering

Face clustering [34] was introduced in cluster hierarchical radiosity in order to produce a more accurate energy transfer between clusters of polygons forming a suitable face cluster. We use a similar face cluster hierarchy to extract the list of occluders.

In the iterative construction of the face cluster hierarchy [11], a polygon or a group of polygons (face cluster) that minimizes a certain number of criteria such as adjacency, planarity, smallest perimeter, etc., is added to its corresponding face cluster. At the end of the construction, we have a list of face clusters, each organized hierarchically.

Figure 1 shows two views of a sofa with face clusters of a given level identified by different colors. Note that the face clusters in this figure are not necessarily flat; the face clusters that we use to extract our occluders are flat. We join these face clusters into a hierarchy of volume clusters similarly to the technique of Mueller *et al.* [18].

We consider a good occluder to be planar and as large as possible. Such occluders include floors, walls, doors, desktops, shelves, etc. Therefore, starting from the root of the scene cluster hierarchy, we look if the current face cluster can be a flat and sufficiently large occluder. Volume clusters are simply traversed recursively until face clusters are reached. We check the planarity of a face cluster from its associated cone of normals.[1] If the face cluster is not satisfying, we continue down the hierarchy until we find such face clusters, or until the area of a face cluster is smaller than a user specified fraction of the dimension of the scene.

If a face cluster lies on the contour of the bounding box of the entire scene, such as floors and ceilings in some scenes, this face cluster would not occlude any element, and it is simply removed from the list of occluding face clusters.

3.2 From Face Cluster to Occluder

Once a face cluster satisfies our set of conditions, we must extract a small number of suitable convex flat polygons approximating *conservatively* the occlusion of this face cluster. These polygons will reside on the supporting plane of the face cluster. Because the orientation on the plane influences the size of the extracted occluders, we need to find the 2D bounding rectangle that has the smallest surface. Currently, we simply test a number of orientations (typically 10) and keep the smallest one. More sophisticated schemes could improve this naive solution, but so far, it has proven sufficient. The contour of the face cluster is then projected orthogonally onto this supporting plane and we proceed with our extraction of 2D convex polygons. It is important to note at this point that the projected face cluster can be concave, and even contain holes.

Our extraction algorithm is similar to the extraction in 3D of volume occluders by Andujar *et al.* [1]. However geometric 3D models might not be closed, or individual polygons might be used as objects to represent ceilings or walls. Therefore to remain as general as possible, we decided to work with surfaces.

We first check some simple conditions to determine if the polygon is already simple and convex. If not, we construct a quadtree representing the 2D face cluster. Each quadtree element contains a list of the edges traversing its surface, or if it is empty, its status as interior or exterior of the face cluster.

From this quadtree, we iterate to extract occluding rectangles with the following steps:

1. Find the interior quadtree element with the largest unmarked area.
2. Expand the quadtree element along one of its two axes until the resulting rectangle intersects an edge.[2]
3. Expand similarly the rectangle along the second axis.
4. Mark the quadtree elements completely inside the rectangle so a different initial quadtree element is used in the next iteration.

[1] In our current implementation, we consider only almost planar occluders, *i.e.*, for which each polygon has an orientation difference of at most 1 degree, resulting from limited representation or numerical instabilities.

[2] In case the quadtree axes would be slightly misaligned with the face cluster, the expanded rectangle could get stuck with an edge. Shrinking the rectangle along the first axis (typically 1/1000th of its length) gives better results.

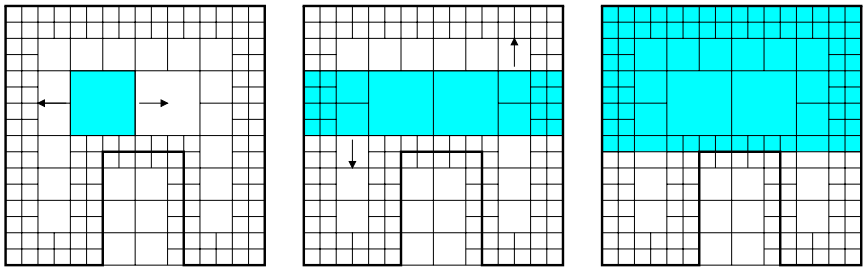


Fig. 2. On the left: largest unmarked interior quadtree element. On the middle and right: the occluding rectangle is expanded along each of its axis.

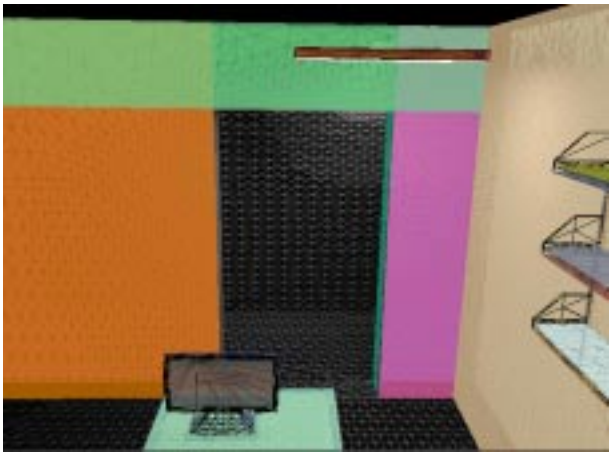


Fig. 3. Extracted large convex occluders in semi-transparent colors over the original mesh of a scene.

The process stops whenever the area of the unmarked quadtree elements is smaller than a specified threshold, or that a given number of rectangles is found. Figure 2 illustrates these steps.

Our occluders are rectangular because they are very simple convex polygons, efficient to intersect, and they are well suited in typical architectural scenes. Our occlusion testing can be easily generalized to any convex polygon, but so far, rectangles have been sufficient.

These rectangles can obviously overlap each other. This is an advantage in our case since we use each of them individually to test against shafts. Each of them should therefore be as large as possible. Another interesting advantage of this extraction technique is that whatever the polygonization of the scene, the extraction of the occluders provides a set of very similar rectangles. Figures 3 and 4 show the extracted occluders for two scenes. In all our test scenes (see Figure 5), the extraction of occluders always took less than 10 seconds on a PC Athlon 600 MHz running linux.

The extracted occluders are kept in a separate hierarchy of bounding volumes aligned on the scene axes. The hierarchy is stored in an array [25] to efficiently access these occluders.

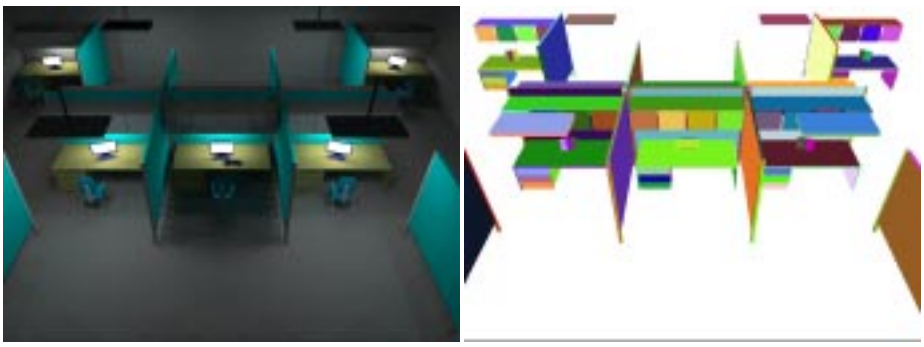


Fig. 4. Original scene and extracted large convex occluders.

4 Cluster Hierarchical Radiosity

This section describes the specificities of the radiosity algorithm we developed.

We implemented our occlusion structure within an algorithm of cluster hierarchical radiosity without links, similar to the one of Stamminger *et al.* [28]. We integrated in this algorithm a simplified version of the face clustering of Willmott *et al.* [34], postponing to a later date the introduction of the notion of "vector-based radiosity" in our system. Face clustering [11] is also used to extract the set of the largest convex occluders. It has shown useful to correct for small inaccuracies in geometric models, thus avoiding undesired cracks.

The scene hierarchy is stored as a binary tree of bounding boxes aligned with the scene axes. It is built with the criteria from Mueller *et al.* [18].

In the radiosity solution, we proceed down the scene hierarchy and shoot radiosity between scene elements starting only at the level of the face clusters or below; we do not currently exchange energy if a volume cluster is involved, although we plan to add it soon to our system. Our oracle uses a BFA refinement criterion, *i.e.*, the shooting radiosity value times the form factor times the area of the face cluster or of the polygon.

We use the technique described by Smits *et al.* [25] when rays need to be shot between two scene elements to test for full occlusion with the bounding segments of shafts or to test partial occlusion [14].

5 Hierarchical Occlusion Culling

In the context of guaranteed occlusion and visibility, we can benefit from the two extreme cases: full visibility and full occlusion.

5.1 Full Visibility

Full visibility has been exploited in a number of visibility techniques. In the case of visibility between two elements (polygons, face clusters, or volume clusters), a shaft [13] between the two elements is constructed from planes, and if no other scene elements lay within the shaft, the two elements are considered fully visible, notwithstanding self-occlusions by the elements.

Our full visibility testing algorithm between two clusters recursively traverses the

bounding boxes intersecting the shaft until:

- A bounding box is completely inside the shaft. We can then stop and consider the two elements as not fully visible.
- A bounding box contains a single polygon. We then test if the polygon intersects the shaft and if so, we consider the two elements as not fully visible.

Any bounding box outside the shaft is simply culled without any further processing.

5.2 Full Occlusion

Determining all the visibility events resulting from a combination of several occluders is a complex task. It is much easier to determine if a single convex occluder blocks the entire shaft. This simply corresponds to test all segments bounding the shaft with the occluder. If all the segments intersect the convex occluder, the two elements are guaranteed to not see any portion of each other. If a single segment does not intersect the convex occluder, although another occluder could block the remaining portion of the shaft, we cannot conclude anything except the two elements might be occluded. We then proceed by subdividing the shaft, treating the sub-shafts recursively. If we reached the lowest level in the scene hierarchy, we resort to the traditional solution of generating n rays randomly between the two elements [14] to test for occlusion.

Our occlusion testing algorithm between two clusters follows these steps:

1. A shaft [13] is built from the bounding boxes (aligned on the scene axes) of the two elements.
2. Depending on the configuration, four to eight segments on the contour of the shaft are computed.
3. The shaft is tested with the hierarchy of extracted occluders.
   (a) We cast a segment in the list of occluders.
   (b) For an occluder intersected, we test with all the remaining segments.
      - As soon as one segment does not intersect this occluder, we start again with another occluder of the list.
      - If all the rays intersect this occluder, we have guaranteed full occlusion.

Since many segments must be tested, we implemented an iterative approach [25] to efficiently traverse the scene hierarchy of bounding boxes instead of the conventional recursive approach. We observed a gain of a factor of three to four in speed for this portion of the algorithm.

5.3 Full Occlusion and Full Visibility in Cluster Hierarchical Radiosity

Our cluster hierarchical radiosity algorithm treats light exchanges by going down the hierarchy. Our oracle determines if the light exchange should be computed between the two current elements, or if we should subdivide one of the two elements and test recursively. Light is exchanged only between face clusters or polygons, never at the level of volume clusters. Visibility requests are included within this oracle. The oracle observes the following conditions:

- If the two elements (face cluster or polygon) are considered flat enough, we compute a standard BFA oracle. If the two cones of normals are entirely backfacing each other, we can stop the recursion.

- Visibility between two elements is tested only every four subdivision levels down the hierarchy. This allows the shaft to be sufficiently smaller to increase the probability of full visibility or full occlusion, and this reduces the computation costs. The jump of four levels empirically provided the best trade-offs in our test scenes. A visibility test is always computed when a light exchange is required between two elements.
- Full occlusion is less expensive to compute, so it is treated first. If we detect full occlusion, subdivision is stopped and no light exchanges occur. Because the occluder hierarchy is separated from the scene hierarchy, it becomes inefficient to attempt to traverse the two hierarchies in parallel.
- If full occlusion is not detected, the test for full visibility is computed. Obviously, one would think that if partial occlusion was detected, there is no needs to test for full visibility. However we observed that marking bounding boxes outside the shaft (explained below) is more efficient even if we know there will not be full visibility.
- In case of full visibility, this information is passed down to their children so they are not tested for visibility.
- All the bounding boxes of the scene hierarchy that are detected as outside the current shaft are marked so they are not tested if the shaft is further subdivided.

6 Results

6.1 Statistics on our Test Scenes

The results for the visibility method presented in the previous section are summarized in Table 1. We selected a set of scenes that feature a variety of visibility types. An image of these scenes appears in Figure 5, along with the number of polygons and the number of extracted occluders. Some of these images have been gamma corrected to better display their features.

Scenes A and E feature arrangements of rooms with relatively limited energy exchanges between them, while in scene B, every room can see a good proportion of the others (scene B is constructed from an array of 10×10 rooms). Scene C has a high occlusion factor but is an open room, this configuration being difficult to handle with portals. Scene D is an example of a room with very little occlusion. Finally, scene F is a mix between full visibility and full occlusion.

We tested three schemes. In the *normal* technique, the oracle decides at which level in the hierarchy of clusters the exchange happens. When this level is reached, the visibility coefficient is calculated by ray casting [14]. Eight rays were used for all scenes, except for scene C (10 rays) and scene D (20 rays). In this latter case, 20 rays were used only to compute an image with less noise. The *visibility* technique is the same as the *normal* technique with the addition of the full visibility test. The *occlusion* technique adds the full occlusion test to the visibility technique.

A single pass in radiosity consists of shooting radiosity from the scene onto itself, going down the scene hierarchy with our BFA criteria. One pass corresponds to all direct illumination, two passes to all the first reflections, etc.

The total time is in seconds, and comes from a PC Athlon 600 MHz with 256 MB of memory and running linux.

For every scene, Table 1 gives the number of energy exchanges between the elements (face clusters or polygons) and the number of them fully blocked, that is every rays shot are blocked. The number of rays shot and the number of these rays blocked by

full occlusion or partial occlusion are also shown. The number of shafts built represents the number of visibility queries for full visibility and full occlusion.

Full occlusion is divided in two parts. First, the proportion of energy exchanges fully occluded. In fact, this number represents a superset of all the fully occluded exchanges because this statistics relies only on the rays cast. Therefore when all sampling rays are blocked, we consider this exchange as fully occluded while it might not be. The second number represents the proportion of full occlusion detected in comparison with the maximum possible. For instance in scene A, 38% of all exchanges were fully occluded. We detected 73% of these full occlusions, therefore close to 28% of all exchanges. Full visibility is the proportion of exchanges that were calculated without requiring to casting rays because they were detected as fully visible.

6.2 Analysis

As one can see from the timings, the combined visibility methods resulted in speedups ranging from 1.3 to 11 for our test scenes, with a typical acceleration of approximately 2.5. Full occlusion testing is fairly efficient, and only in the second and third passes in scene D did we observe a very slight increase (about 1% of computation time) due to full occlusion testing.

One can note that the gain slightly reduces for indirect lighting. We believe this occurs because these energy exchanges have lower energy and thus are handled at higher levels in the scene hierarchy. As a consequence the relative size of our occluders with respect to the size of shafts tends to diminish, so that full occlusion and full visibility become less frequent.

Although our occluder extraction algorithm does not allow general occluder fusion, as much as 93% (scene F, pass 1) of all occlusions were accounted for by our occluders. Another interesting aspect is that the occlusion test is fairly inexpensive, so that even though scene D has very little occlusion, the full occlusion test had almost no impact on the total time.

The maximal proportion of full occlusions calculated in our test scenes can appear less than what one might expect. For instance, test scene F, pass 1 has a full occlusion factor of 38%. This is explained by the fact that most of the energy exchanges occur nearby the emitter, due to the form factor contribution to the BFA error metrics in the oracle. Therefore many of the light exchanges between distant elements (that would be good candidates for full occlusion) are never considered because of this oracle; these exchanges occur at a higher level in the scene hierarchy.

7 Conclusion

We presented a guaranteed visibility culling algorithm implemented in a cluster hierarchical radiosity algorithm with face clustering. We exploited efficiently full visibility and full occlusion, providing an early stop in visibility determination with no errors. This leads to higher quality energy exchanges at a lower computational cost.

We put emphasis on culling due to full occlusion. Face clustering is used to extract a set of rectangles representing flat sections of surfaces. These rectangles are organized in a hierarchy of occluders for efficient visibility queries with the segments bounding a shaft between two elements. The user can control the size and the number of occluders.

We tested our culling method onto a large set of different scenes, and analyzed the results. In all cases, our structure proved useful, even when full visibility and full occlusion were not apparent. The resulting visibility structures are very simple to build,

Scene Pass	Culling Technique	Total Time (sec)	Energy Exch (M)	Exch Blocked (M)	Rays Shot (M)	Rays Blocked (M)	Shafts Built (M)	Full Occlus. (%)		Full Vis. (%)
A . 1	normal	428	4.7	1.78	37.9	15.8	0	38	73	46
	visibility	271	4.7	1.78	20.3	15.8	3.0			
	occlusion	220	3.4	0.47	9.9	1.9	1.9			
B . 1	normal	382	4.8	2.91	39.0	24.5	0	61	69	29
	visibility	275	4.8	2.91	27.7	24.5	9.5			
	occlusion	173	2.8	0.86	11.3	8.1	3.4			
C . 1	normal	199	1.20	0.64	12.07	6.96	0	53	81	19
	visibility	141	1.20	0.64	9.80	6.96	0.98			
	occlusion	88	0.68	0.13	4.65	1.84	0.44			
C . 2	normal	430	2.40	1.95	23.72	20.84	0	81	75	2
	visibility	435	2.40	1.95	23.33	20.84	6.96			
	occlusion	200	0.93	0.49	8.68	6.45	2.57			
C . 3	normal	532	2.99	2.66	29.90	28.12	0	88	77	1
	visibility	553	2.99	2.66	29.66	28.07	8.99			
	occlusion	228	0.93	0.62	9.14	7.58	3.17			
D . 1	normal	93	0.32	0.05	6.60	1.29	0	16	40	56
	visibility	28	0.32	0.05	2.88	1.29	0.08			
	occlusion	25	0.30	0.02	2.31	0.73	0.07			
D . 2	normal	55	0.23	0.02	4.62	0.85	0	9	50	34
	visibility	33	0.23	0.02	3.04	0.85	0.18			
	occlusion	34	0.22	0.02	2.95	0.80	0.17			
D . 3	normal	44	0.17	0.02	3.38	0.98	0	12	50	24
	visibility	33	0.17	0.02	2.57	0.98	0.22			
	occlusion	34	0.16	0.02	2.48	0.93	0.22			
E . 1	normal	273	2.29	1.41	18.37	12.02	0	62	81	18
	visibility	158	2.29	1.41	15.14	12.02	1.02			
	occlusion	95	1.15	0.27	6.01	2.89	0.47			
E . 2	normal	118	0.98	0.40	7.84	3.74	0	41	58	26
	visibility	76	0.98	0.40	5.81	3.74	0.74			
	occlusion	69	0.75	0.17	3.95	1.92	0.49			
E . 3	normal	89	0.84	0.48	6.70	4.24	0	57	65	17
	visibility	73	0.84	0.48	5.53	4.23	0.96			
	occlusion	59	0.53	0.18	3.08	1.82	0.52			
F . 1	normal	1017	11.13	4.26	89.04	34.30	0	38	93	61
	visibility	413	11.13	4.26	34.92	34.30	2.28			
	occlusion	92	7.15	0.29	3.12	2.50	0.36			
F . 2	normal	639	5.63	2.41	45.06	19.66	0	43	66	53
	visibility	256	5.63	2.41	21.07	19.66	1.50			
	occlusion	139	4.05	0.82	8.36	6.96	1.05			
F . 3	normal	298	2.57	1.12	20.53	9.24	0	44	53	51
	visibility	126	2.57	1.12	10.11	9.24	0.78			
	occlusion	86	1.98	0.54	5.40	4.53	0.66			

Table 1. Full visibility and full occlusion on various scenes

as small as desired, provide interesting speed ups, and produce no visibility errors.

We integrated our culling method in a cluster hierarchical radiosity algorithm, but it should be simple to use it in walkthroughs, in rendering from a given viewpoint, and direct illumination.

We expect also that the method could be exploited in improved radiosity algorithms with exchanges between volume clusters, irradiance vectors, and final gathering.

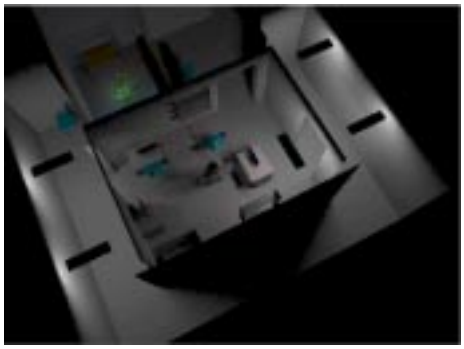
Other promising directions for future work include the use of volume occluders and directional occluders for curved surfaces.

Acknowledgments. We acknowledge financial support from NSERC and Discreet, a division of Autodesk. We would like to thank Frédo Durand and Filippo Tampieri for helpful comments.

References

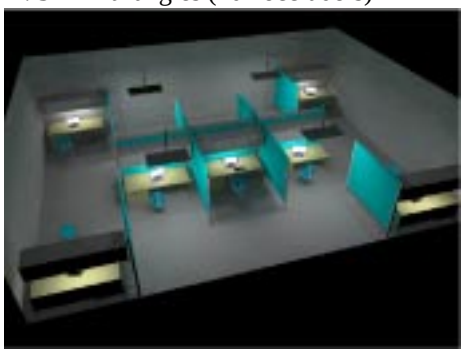
1. C. Andújar, C. Saona-Vázquez, and I. Navazo. LOD visibility culling and occluder synthesis. *Computer Aided Design*, 2000. Accepted for publication.
2. P.H. Christensen, D. Lischinski, E.J. Stollnitz, and D.H. Salesin. Clustering for glossy global illumination. *ACM Transactions on Graphics*, 16(1):3–33, January 1997.
3. D. Cohen-Or, G. Fibich, D. Halperin, and E. Zadicario. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum (Eurographics '98)*, 17(3):243–254, 1998.
4. S. Coorg and S. Teller. Real-time occlusion culling for models with large occluders. In *1997 Symposium on Interactive 3D Graphics*, pages 83–90, April 1997.
5. G. Drettakis and E. Fiume. A fast shadow algorithm for area light sources using backprojection. In *Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 223–230, July 1994.
6. R. Dumont and K. Bouatouch. Using levels of detail to speedup radiosity computation. Technical Report RR-3602, INRIA, 1999.
7. F. Durand. *3D visibility, analysis and applications*. Ph.D. thesis, Université Joseph Fourier, Grenoble, 1999.
8. F. Durand, G. Drettakis, and C. Puech. Fast and accurate hierarchical radiosity using global visibility. *ACM Transactions on Graphics*, 18(2):128–170, April 1999.
9. F. Durand, G. Drettakis, and C. Puech. The visibility skeleton: A powerful and efficient multi-purpose global visibility tool. *Proceedings of SIGGRAPH 97*, pages 89–100, August 1997.
10. F. Durand, G. Drettakis, J. Thollot, and C. Puech. Conservative visibility preprocessing using extended projections. In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, July 2000, to appear.
11. M. Garland. *Quadric-Based Polygonal Surface Simplification*. Ph.D. thesis, Carnegie Mellon University, 1999.
12. S. Gibson and R.J. Hubbold. Efficient hierarchical refinement and clustering for radiosity in complex environments. *Computer Graphics Forum*, 15(5):297–310, 1996.
13. E. Haines and J. Wallace. Shaft culling for efficient ray-traced radiosity. In *Eurographics Workshop on Rendering*, 1991.
14. P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 197–206, July 1991.
15. T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang. Accelerated occlusion culling using shadow frustra. In *Proceedings of the Thirteenth Annual Symposium on Computational Geometry*, pages 1–10, June 1997.
16. F.-A. Law and T.-S. Tan. Preprocessing occlusion for real-time selective refinement. In *1999 Symposium on Interactive 3D Graphics*, pages 47–52, 1999.

17. D. Meneveaux. *Lighting simulation in complex architectural environments: sequential and parallel approaches*. Ph.D. thesis, Université de Rennes I, 1998.
18. G. Mueller, S. Schaefer, and D. Fellner. Automatic creation of object hierarchies for radiosity clustering. In *Pacific Graphics '99*, October 1999.
19. H. Plantinga. Conservative visibility preprocessing for efficient walkthroughs of 3D scenes. In *Proceedings of Graphics Interface '93*, pages 166–173, May 1993.
20. H. Plantinga and C.R. Dyer. Visibility, occlusion, and the aspect graph. *International Journal of Computer Vision*, 5(2):137–160, 1990.
21. G. Schaufler, J. Dorsey, X. Decoret, and F.X. Sillion. Conservative volumetric visibility with occluder fusion. In *SIGGRAPH 2000 Conference Proceedings*, Annual Conference Series, July 2000, to appear.
22. F. Sillion. Clustering and volume scattering for hierarchical radiosity calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, June 1994.
23. F. Sillion, G. Drettakis, and C. Soler. A clustering algorithm for radiance calculation in general environments. In *Eurographics Rendering Workshop 1995*, pages 196–205, June 1995.
24. F.X. Sillion. A unified hierarchical algorithm for global illumination with scattering volumes and object clusters. *IEEE Transactions on Visualization and Computer Graphics*, 1(3):240–254, September 1995.
25. B. Smits. Efficiency issues for ray tracing. *Journal of Graphics Tools*, 3(2):1–14, 1998.
26. B. Smits, J. Arvo, and D. Greenberg. A clustering algorithm for radiosity in complex environments. In *Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 435–442, July 1994.
27. C. Soler and F.X. Sillion. Fast calculation of soft shadow textures using convolution. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 321–332, July 1998.
28. M. Stamminger, H. Schirmacher, P. Slusallek, and H.-P. Seidel. Getting rid of links in hierarchical radiosity. *Computer Graphics Forum (Eurographics '98)*, 17(3):165–174, 1998.
29. A.J. Stewart and S. Ghali. Fast computation of shadow boundaries using spatial coherence and backprojections. In *Proceedings of SIGGRAPH '94*, Annual Conference Series, pages 231–238, July 1994.
30. S. Teller and P. Hanrahan. Global visibility algorithms for illumination computations. In *Proceedings of SIGGRAPH '93*, Annual Conference Series, pages 239–246, 1993.
31. S.J. Teller and C.H. Séquin. Visibility preprocessing for interactive walkthroughs. In *Computer Graphics (SIGGRAPH '91 Proceedings)*, volume 25, pages 61–69, July 1991.
32. J.R. Wallace, K.A. Elmquist, and E.A. Haines. A ray tracing algorithm for progressive radiosity. In *Computer Graphics (SIGGRAPH '89 Proceedings)*, volume 23, pages 315–324, July 1989.
33. Y. Wang, H. Bao, and Q. Peng. Accelerated walkthroughs of virtual environments based on visibility processing and simplification. *Computer Graphics Forum (Eurographics '98)*, 17(3):187–194, 1998.
34. A. Willmott, P. Heckbert, and M. Garland. Face cluster radiosity. In *Eurographics Workshop on Rendering*, pages 293–304, June 1999.



A: 37424 triangles (204 occluders)

B: 83600 triangles (1140 occluders)



C: 51296 triangles (549 occluders)

D: 741 triangles (119 occluders)

E: 14453 triangles (549 occluders)

F: 642 triangles (203 occluders)

Fig. 5. Test scenes