EUROGRAPHICS '98 / N. Ferreira and M. Göbel
(Guest Editors)

Volume 17, (*1998*), *Number 3*

Conservative Visibility and Strong Occlusion for Viewspace Partitioning of Densely Occluded Scenes

Daniel Cohen-Or, Gadi Fibich, Dan Halperin and Eyal Zadicario

School of Mathematical Sciences
Tel-Aviv University, Ramat-Aviv 69978, Israel
Email: daniel@math.tau.ac.il

Abstract

Computing the visibility of out-door scenes is often much harder than of in-door scenes. A typical urban scene, for example, is densely occluded, and it is effective to precompute its visibility space, since from a given point only a small fraction of the scene is visible. The difficulty is that although the majority of objects are hidden, some parts might be visible at a distance in an arbitrary location, and it is not clear how to detect them quickly. In this paper we present a method to partition the viewspace into cells containing a conservative superset of the visible objects. For a given cell the method tests the visibility of all the objects in the scene. For each object it searches for a strong occluder which guarantees that the object is not visible from any point within the cell. We show analytically that in a densely occluded scene, the vast majority of objects are strongly occluded, and the overhead of using conservative visibility (rather than visibility) is small. These results are further supported by our experimental results. We also analyze the cost of the method and discuss its effectiveness.

1. Introduction

Consider a scene consisting of a large number of polyhedral objects. Given a viewpoint *s*, the answer to an ε-*visibility query* is the set of all polygons visible from *s* or from an ε neighborhood of *s*. The ε-visibility query is a basic tool for viewspace partitioning which has many applications in computer graphics, including ray tracing and Monte-Carlo based global illumination algorithms[11], shaft-culling for radiosity[12] and interactive walkthroughs of complex scenes[1, 10, 16]. In all these cases a precomputed viewspace partition saves further processing of objects which are not visible from a given point or region.

Visibility queries can be answered by constructing an *aspect graph*. Aspect graphs are tools to encode all the possible two-dimensional images of a three-dimensional scene[13]. They enumerate all possible appearances of a polyhedral scene, by partitioning the viewspace into qualitatively maximal regions in which the viewpoints have the same view or aspect. The aspect graph is an extremely expensive data structure; under perspective projection its size can be $\Omega(n^9)$ in the worst case for polyhedra with a total of *n* vertices. In their raw form, aspect graphs may require prohibitively large storage space and preprocessing time.

Furthermore, the aspect graph contains more information than is required for an ε-visibility query. An efficient viewspace partition should not necessarily be maximal, that is, the space partition can be coarser with each cell including several aspect regions. Such a relaxation allows the ε-visibility set to be a conservative superset which includes at least all the visible polygons from a given region, plus possibly some occluded polygons.

Most previously developed methods for precomputing the visibility of in-door scenes take advantage of the inherent properties of such scenes. Computing the visibility of out-door scenes is much harder. An urban scene (see Figure 1), for example, is densely occluded, and it is quite tempting to precompute its visibility space, since from a given point only a small fraction of the scene is visible. The difficulty is that although the majority of objects are hidden, some parts may be visible at a distance in an arbitrary location, and it is not clear how to detect them quickly. Visibility problems are notoriously complicated. This is reflected in the fact that a small change

Figure 1: *A densely occluded urban model.*

in the viewpoint might cause large changes in the visibility. Having the visibility solved at one point does not help much in solving the visibility at a nearby view. This further justifies the need for a precomputed viewspace partition.

In this paper we present a method to partition the viewspace into cells, where each cell contains a conservative superset of visible objects. For a given cell the method tests the visibility of all the objects in the scene. For each object it searches for a *strong occluder* which guarantees that the object is not visible from any point within the cell. In Section 3 we define the problem and outline the algorithm which extends our preliminary work[4]. In Section 4 we develop a probabilistic model to quantify the visibility and we show that in a densely occluded scene, the vast majority of objects is occluded and for most of them there is a strong occluder. This is further supported by our experimental results, reported in Section 6. We also analyze the cost of the method and present several optimizations.

2. Related Work

The aspect graph has been an ongoing theme in the computer vision community, motivated by problems in object recognition[8, 13]. Later, researchers in computational geometry joined in to point out the existence of natural three-dimensional scenes whose induced aspect graphs are more compact than those induced by general three-dimensional scenes[2, 3]. Recently, the computer graphics community found the most interest in aspect graphs, with the goal of using them to allow for fast interactive graphic visualization on low-end graphics machines[6] and over the net[4].

Shimshoni and Ponce[15] presented an approximation to the aspect graph by generating a *finite-resolution* aspect graph. By ignoring edges whose visible portions project onto a segment of length smaller than some fixed ε, the approximate aspect graph has fewer nodes. However, the time complexity for generating the finite-resolution aspect graph is even higher than for a infinite-resolution aspect graph.

Coorg and Teller[5] presented an algorithm to incrementally maintain a conservative visibility set during a walkthrough by detecting a subset of the changes in the aspect. However, their visibility set is valid only for the current frame at a given viewpoint and not necessarily for its ε-neighborhood. Airey et al.[1] and Teller[9] described methods for interactive walkthroughs of complex buildings that compute the potentially visible set of surfaces for each room in a building. These methods take advantage of the inherent property of indoor scenes which partition the space into “cells” and “portals”. Then the cell-to-cell visibility can be precomputed. Such methods are very effective for building interiors, but are not necessarily suited for other types of virtual worlds.

Plantinga[14] presented a method for partitioning the viewspace into 2D cells, each containing a conservative visibility set. The method assumes the existence of an a priory set of effective occluders, like the walls in the interior of buildings. The method is based on the computation of all the visual events among the occluders. This method is effective only if a small set of occluders contributes the majority of the occlusion.

The methods developed by Hudson et al.[7] and by Coorg and Teller[5, 6] also assume the existence of a small set of effective occluders determined in a preprocessing stage. These methods accelerate the rendering from a given viewpoint by creating or updating a conservative visibility set. However, the set is valid only for the current position of the viewpoint, and not for a region.

Other related work deals with shadow algorithms[17]. Regarding the viewpoint and its ε-neighborhood as an area light source, occlusion culling algorithms search for the primitives which are in the umbra. However, shadow algorithms are usually analytical and much too expensive to simply cull away the primitives in the umbra.

3. Strong Occluders and Conservative Visibility

Let's assume an urban model consisting of a large number of buildings. For simplicity we first assume that each building is a single box. Later we will discuss the more general case. Each box is made of a number of triangles.

The viewspace is partitioned into *cells* which are sometimes referred to as *viewspace cells*. Each cell defines a *set of visible triangles*, that is, a set of triangles, each of which is at least **partially** visible from some point in the cell. For a given cell our method computes a conservative superset of the visible triangles. Instead of looking for the visible triangles we look for triangles which are guaranteed to be hidden from any viewpoint within the cell.

Given a viewspace cell C and two objects S and T, we say that T is a *strong occluder* of S with respect to C, if T fully occludes S from every viewpoint in C. Since the cell C will be clear from the context we will simply say that T is a strong occluder for S. Given a polyhedral object S our algorithm looks for a convex strong occluder for S. If no strong occluder for S is found , we say that S is *potentially visible*. If a potentially visible object is found to be occluded inside our viewpoint cell, we call it a *weakly occluded* object. We call the set of potentially visible objects for a given viewspace cell, a *conservative visibility set*.

We will show the effectiveness of our method for densely occluded sets by showing that: (i) the number of weakly occluded objects in each of our conservative visibility sets is relatively small, and (ii) we can construct the potentially visible sets efficiently (far more efficiently than it would have taken to compute the visibility set precisely).

Given a viewpoint s and a convex polyhedron P with vertices v_i, if all the rays connecting s and v_i intersect a single convex occluder O, then O is a strong occluder of P with respect to s. That is, for the viewpoint s, O is guaranteed to occlude any point v' in P[5]. Now, given a viewspace cell C defined by its vertices s_j, if O is a strong occluder of P with respect to all s_j, then O is easily verified to be a strong occluder of P with respect to C.

The definition of a strong occluder directly implies an algorithm to detect the strong occluder of a given polyhedron P with respect to a visibility cell[4]. A naive algorithm casts rays from every vertex of the cell towards each vertex of the polyhedron P. Each ray reports a list of potential occluders which is the set of convex objects that have been intersected by the ray. If the intersection of all these lists is not empty then P is reported as being strongly occluded. The conservative visibility set of a given cell is constructed by traversing all the objects of the scene. Each object that is strongly occluded is culled away and not included in the conservative visibility set.

We partition the viewspace into cubic cells for simplicity; a tetrahedron may be advantageous for certain applications[4] since it has less vertices. Note that the occludee can be either the individual triangles or the entire object, or its bounding box. One effective optimization is to try to find a strong occluder to the bounding box before processing the individual triangles.

Remark: Even if O is non-convex, it may still be a strong occluder. However, determining that O is a strong occluder becomes rather complicated. As we will show below, part of the strength of our method comes from using only very simple operations, which will no longer hold if we have to carry out such tests. In the conclusions section we propose a way to extend our method to non-convex occluders. Also the shape of the cell need not be convex; convex shapes, however, can be handled more efficiently.

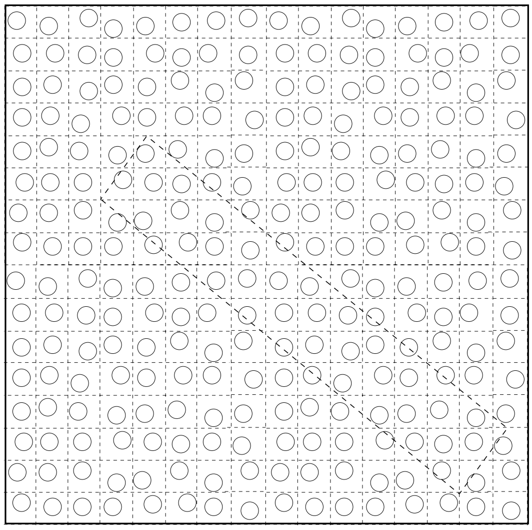


Figure 2: *The distribution model; the number of spheres contained in an arbitrary rectangle is proportional to its size.*

4. Visibility Analysis

The purpose of our algorithm is to find in a given scene the visibility set A_{vis}, which consists of all objects that are visible from a given viewspace cell C_ε. However, what our algorithm can do efficiently is to eliminate all the strongly occluded objects. This leaves us with the conservative visibility set A_{c-vis} which consists of two subsets: (i) A_{vis} and (ii) the set of all weakly occluded objects. In this section we provide an analytical support for this approach by proving that:

1. The overhead of computing A_{c-vis} (rather than A_{vis}) is small.
2. The probabilities of an object to belong to A_{c-vis} or to A_{vis} decrease exponentially with its distance from C_ε.

To simplify the presentation, we consider a scene consisting of spheres distributed randomly in space. All spheres have the same radius R ($0 < R \leq 1$) and are assumed to have

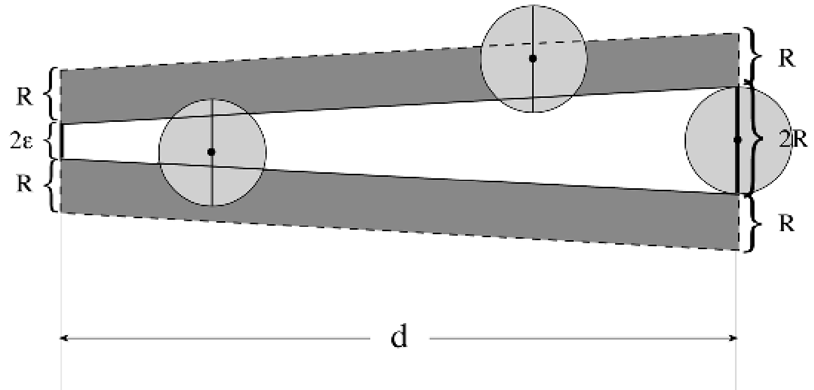


Figure 3: *A sphere is (partially) occluding the sphere on the right from C_ε (the 2ε segment on the left) $\iff$ it intersects with the smaller trapezoid $\iff$ its center lies within the larger trapezoid. Here, only the left sphere is a strong occluder.*

a *minimum-distance* Poisson distribution in which no pair of spheres is closer than a fixed minimum distance (see Figure 2). In our simulations we generated their location by *jittering*, that is, perturbing sample locations that are initially spaced on a regular grid. Each sphere can be associated with its initial source grid position, and is located within the bounds of the cube centered around the grid point. In terms of the visibility between two remote spheres, jittering yields a distribution which approximates a Poisson distribution. The cubes have fixed dimensions of 2×2 in the 2D case and $2\times 2\times 2$ in the 3D case.

4.1. The 2D case

We first present results for the 2D case, for which the calculations are simpler. In this case, the terms 'spheres' and 'cubes' correspond to circles and squares, respectively. Let us consider a visibility cell C_ε of size $2\varepsilon\times 2\varepsilon$ and a (potentially occluded) sphere S_1 at a distance d from C_ε. In the following, we estimate the probabilities:

$$\begin{aligned} P_1 &= P(S_1 \in A_{vis}) \ , \\ P_2 &= P(S_1 \in A_{c-vis}) \ . \end{aligned}$$

To do that, we construct the trapezoid T_1 whose bases are the two diameters 2ε and $2R$ and whose height is d (Fig. 3). Let K be the number of (partially) occluding spheres between C_ε and S_1, i.e. those that intersect with T_1. To find K, we note that it is equal to the number of sphere centers located within T_2, which is a larger trapezoid formed by extending the bases of T_1 by R in both directions (Figure 3). Since we have one sphere per 2×2 square, on average, the number of sphere centers within any volume V is $V/2^2$. Therefore,

$$K \sim \frac{V(T_2)}{4} = \frac{(3R+\varepsilon)}{4} d \ . \qquad (1)$$

Since we are interested in the case where $d \gg R$, we can approximate the trapezoid T_1 with a rectangle with height d and base $2b$, where $\varepsilon \le b \le R$. The sphere S_1 is visible from C_ε if the union of the projected images of the K potentially occluding spheres on the basis of the rectangle does not cover it completely (strictly speaking, the converse is not true i.e. even if the basis is completely covered, S_1 may still be visible. However, for $d \gg R$ the probabilities of these two events are almost the same). In the appendix, we prove the following Lemma:

Lemma 4.1 Let K intervals I_k $(k = 1,\ldots,K)$ of length $2R$ be uniformly distributed in the interval $[-2R-b \quad 2R+b]$ and let I_b be the interval $[-b \quad b]$. Then

$$\begin{aligned} \hat{P}_1 &:= P(I_b \not\subset \cup_k I_k) = (K+1)\left(\frac{1}{m_b+1}\right)^K \ , \\ \hat{P}_2 &:= P(\nexists k_0, I_b \subset I_{k_0}) = \left(\frac{2}{m_b+1}\right)^K \ , \end{aligned}$$

where

$$m_b := \frac{R}{b} \ge 1 \ .$$

From this lemma we see that the two key parameters for visibility at a distance are m_b and K. Although we did not give an exact definition of b, it is clear that b is some average of R and ε and for our qualitative analysis we can take $b = (R+\varepsilon)/2$. Since K is proportional to d (the constant of proportionality, according to Equation (1), is $(3R+\varepsilon)/4 \sim \delta^{1/2}$, where δ is the area density of the spheres), visibility decreases exponentially with distance. However, there is a marked difference between the decrease in visibility set and in potential visibility set, since the bases of the exponent for visibility and potential visibility are $1/(m_b+1)$ and $2/(m_b+1)$, respectively. Therefore, visibility decreases with distance for all ε, with faster decay for smaller ε. However, potential visibility decreases exponentially with distance only when the viewspace cell is smaller than the objects. The borderline case is $\varepsilon = R$, where $b = R$ and $m_b = 1$. In this case, all cells are potentially visible (yet visibility still decreases like 2^{-K}). Since for our algorithm to work efficiently, strong occlusion should be a close approximation to being hidden, we see from Lemma 4.1 that the viewspace cell should be truly smaller than the objects.

The analogy between the P_i's and the $\hat{P}_i$'s allows us to draw the following conclusions: In a densely occluded scene, like a city, if the visibility cell is truly smaller than the objects:

- The vast majority of hidden objects are strongly occluded.
- The vast majority of all distant objects are strongly occluded.

We remark that the difference between the probability distribution used in Lemma 4.1 and the one used in the simulations is insignificant with regard to these qualitative statements.

We can also use the Lemma to calculate upper bounds for the total number of spheres in A_{vis} and in A_{c-vis}, denoted by N_{vis} and N_{c-vis}, respectively. Let us assume that all spheres whose distance from C_ε is at most r_0 are visible. There are, clearly, less than $\pi(\varepsilon + r_0)^2/4$ such spheres. In addition, the number of spheres whose distance from C_ε is between r and $r + dr$ is approximately $2\pi r\,dr/4$, the average number of occluding spheres between this annulus and C_ε is

$$K = \bar{b}r \quad \text{where} \quad \bar{b} = \frac{3R + \varepsilon}{4}$$

(see Equation (1)), and the probabilities for each of these spheres to be in A_{vis} and in A_{c-vis} are given by $\hat{P}_1$ and by $\hat{P}_2$, respectively. Thus,

$$N_{vis} \leq \frac{\pi(\varepsilon + r_0)^2}{4} + \int_{r=r_0}^{\infty} \frac{2\pi r\,dr}{4}(\bar{b}r + 1)\left(\frac{1}{m_b + 1}\right)^{\bar{b}r}$$

and

$$N_{c-vis} \leq \frac{\pi(\varepsilon + r_0)^2}{4} + \int_{r=r_0}^{\infty} \frac{2\pi r\,dr}{4}\left(\frac{2}{m_b + 1}\right)^{\bar{b}r} .$$

These integrals can be evaluated exactly, giving

$$N_{vis} \leq \frac{\pi(\varepsilon + r_0)^2}{4} + \frac{\pi}{2\bar{b}^2 \ln^3(m_b + 1)}\left(\frac{1}{m_b + 1}\right)^{r_0\bar{b}} \times$$
$$\left[2 + (2\bar{b}r_0 + 1)\ln(m_b + 1) + (r_0^2\bar{b}^2 + r_0\bar{b})\ln^2(m_b + 1)\right] ,$$

$N_{c-vis} \leq$

$$\frac{\pi(\varepsilon + r_0)^2}{4} + \frac{\pi}{2}\,\frac{1 + r_0\bar{b}\ln\dfrac{m_b + 1}{2}}{\bar{b}^2 \ln\dfrac{m_b + 1}{2}}\left(\frac{2}{m_b + 1}\right)^{r_0\bar{b}}$$

for $m_b > 1$, and $N_{c-vis} \leq \infty$ for $m_b \leq 1$. If necessary, these bounds can be further lowered by finding the optimal value of r_0.

The last bound for N_{c-vis} shows again that as $\varepsilon \nearrow R$, the number of potentially visible spheres becomes unbounded. Note that as $\varepsilon \searrow 0$, the number of visible spheres, as well as the number of potentially visible ones, goes down to a non-zero constant, which is all spheres (potentially) visible from a point. Therefore, there is no advantage in using viewspace cells significantly smaller than the size of the objects.

4.2. Extension to 3D

The picture in the 3D case is qualitatively similar to the one in the 2D case. The main difference is that for the algorithm to be efficient, C_ε should be relatively smaller than in the 2D case, as we shall now see.

Let us estimate the probability $1 - \hat{P}_2$ for a sphere S_1 in 3D to be strongly occluded from C_ε which is at a distance d. To do that, we construct a truncated cone H_1 with small base with radius ε, large base with radius R and height d. All K occluding spheres between S_1 and C_ε intersect H_1, and their centers lie within an extended truncated cone H_2 of the same height and with bases whose radii are $\varepsilon + R$ and $2R$. Therefore, the number K is, on average,

$$K = \frac{V(H_2)}{2^3} = \frac{\pi d}{24}(7R^2 + 4\varepsilon R + \varepsilon^2) .$$

As before, for d large, we can replace H_1 with a cylinder with height d and base with radius b, where $\varepsilon \leq b \leq R$. The probability for conservative visibility is given by the following lemma (whose proof is given in the Appendix):

Lemma 4.2 Let B_r denote a circle with radius r centered at the origin. If K circles of radius R are uniformly distributed in the circle $B_{2R+\varepsilon}$, the probability that none of the K circles completely covers the circle B_b is:

$$\hat{P}_2 = \left(\frac{4m_b}{(1 + m_b)^2}\right)^K .$$

This result is qualitatively similar to the one in the 2D case. However, ε should be much smaller than in the 2D case for the algorithm to work efficiently. For example, if we want $\hat{P}_2$ to decay like 2^{-K}, then $m_b = 3$ in the 2D case but $m_b \sim 6$ in the 3D case. In addition, the constant of proportionality between K and d is approximately $R^2 \sim \delta^{2/3}$, where δ is the volume density of the spheres.

5. Cost Analysis and Optimizations

Assume for simplicity that the n objects of the scene are boxes, and we are constructing the set of conservative visible bounding boxes, without refining them down to the triangle level. For each one of the n boxes the algorithm casts 8×8 rays, where each ray tests its intersection with all the $n - 1$ boxes. However, in a typical model, like an urban model,

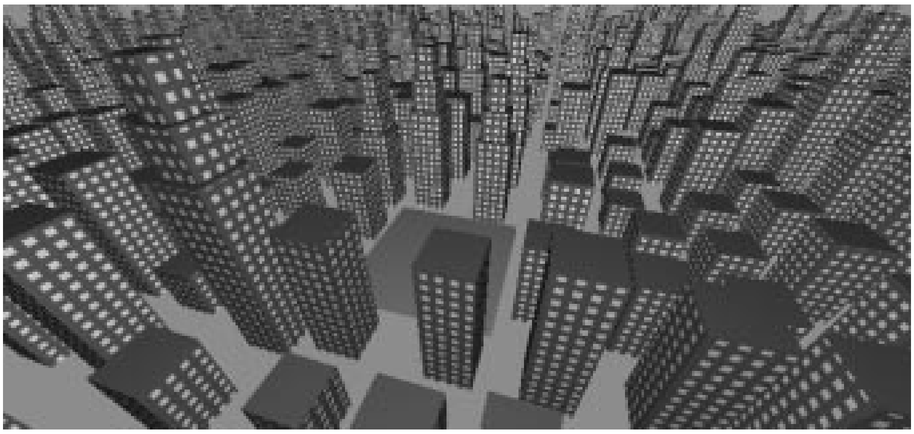


Figure 4: *The 400 $\times$ 400 green square for which we have computed the visibility.*

where the objects are distributed rather uniformly, it is common to use some space subdivision techniques by which a ray needs to test its intersection with only roughly $c\sqrt{n}$ objects. Thus, the cost of generating the conservative visible set of a given cell is roughly $64nc\sqrt{n}$. This estimate is for a naive implementation. In the following we will discuss some optimizations by which the cost in practice becomes linear in n.

1. The first and most effective optimization tries to reduce the number of objects each ray has to be intersected with. Indeed, as will be shown later, in most cases, the strong occluder is found within the first five objects or trials. Thus, it would be better to test one occluder at a time. One ray, the *leading ray*, is cast from one of the cell vertices, and retrieves the first candidate encountered by the ray. The rest of the rays test their intersection with the candidate hoping to classify the candidate as a strong occluder. If one of the rays fails to intersect the candidate, then the leading ray retrieves the next candidate, until either a strong occluder is found or the leading ray reaches the object.
   For visible objects it would be better to first cast each ray all the way through the occluders to quickly detect a visible vertex. However, the cost of processing visible objects is insignificant since the vast majority is occluded and most of the visible objects are located very close to the viewspace cell, and have few, if any, potential occluders.
2. An interesting observation is that a strongly occluded object is a redundant occluder. By processing the objects close to far, the population of candidate occluders can be diluted, without losing the effective strong occluders. Although this optimization sounds promising, in most cases, the effectiveness of the first optimization dominates it, since the effective strong occluders are tested first. However, its implementation requires only to mark strongly occluded objects and to skip their further processing for a given cell.
3. Another optimization can reduce the number of rays cast from the cell to the object. Instead of using all the rays connecting all the vertices of the cell and all the vertices of the candidate object, it is enough to use only the subset of the rays, since rays which do not lie on the convex hull defined by the vertices of the cell and the vertices of the candidate are redundant.
4. A different type of optimization is to exploit the coherence among the objects of the scene. One way is to test the visibility of a group of objects together. The objects can be organized in some hierarchical structure of bounding boxes, so that a group of nearby objects can have a common strong occluder. This is a common acceleration technique in ray tracing. Indeed, in some sense our technique is an extension of ray tracing since here we test for a simultaneous intersection of several rays.
5. Another possible way to exploit the coherence among the objects is by maintaining a cache of "potential strong occluders". This assumes that if an object has a strong occluder then the same occluder has a good chance of being a strong occluder to its neighbor.
6. A different type of optimization is to exploit the coherence among the cells. The visibility of the cells can be generated by a top-down computation. First, large cells from the top of the hierarchy are computed. Then these coarse viewspace cells are refined. The smaller subcells are computed with less effort, since most of the primitives have already been removed. However, care should be taken to avoid redundant computation. If a cell is too wide then the overhead is too costly, and if the overhead is small enough the cost of refining it might be too costly.

The effectiveness of these optimizations is model dependent, and they can be ineffective for some cell sizes. Moreover, they are interdependent, namely using certain combinations of optimizations may yield unexpected results since the overhead of some tests can dominate the process. Therefore, in the following section we present our results implemented with only the first optimization (1) above, so the cost and effectiveness are better understood. We also report on the effectiveness of the hierarchical scheme.

6. Experimental Results

The algorithm was tested on a city model consisting of 2500 textured buildings, 75% of them simple boxes, and the rest made up of three boxes each. The base of each building is a 100×100 square, its height is a random value between 200 and 900. In total, there are 3324 boxes, each of which consists of ten polygons, so the total number of polygons is 33240. In the center of the city we have removed four buildings and for this region we are computing the visibility space (see the green square in Figure 4). The size of the viewspace region is 400×400 units.

Tables 1 and 2 report on the potentially visibility sets computed for different cell sizes. Only ground level cells have been computed (height of 50). Figure 6 visualizes the results by rendering the city from a camera vertically above the city. The strongly occluded buildings are colored in red, while the potentially visible buildings are colored in green.

In the tables we distinguish between strongly occluded boxes and strongly occluded buildings. The latter are buildings which are not occluded as a whole, but every piece (the individual polygons) is strongly occluded so that their aggregate defines the building as being strongly occluded. Note that the number of ray/box intersections is linear to the number of tested boxes. In some cases a box can be determined as being potentially visible by just one or two ray/box intersections and in others by over twenty. In our current implementation the strong occluder is determined by 64 intersections. The tables also show that in smaller cells more boxes are strongly occluded. This suggests that it might be effective to use the upper levels just to cull boxes, and the finest level to cull the polygons.

That is why we have separated the ray/box intersections from the ray/polygon intersections. This is important because it is not clear how to estimate the cost of the calculation. Using different optimizations can yield different running times which are all scene dependent. In the following we use the number of ray/box intersections as the cost estimate for estimating the cost effectiveness of the technique. As we have seen above, the occlusion factor is a function of the ratio between the cell size and the occluder size. In our tested scene, the occluders have a 100^2 base; thus, as expected, only cells smaller than 100 are cost effective. According to Figure 5(a) the cost effectiveness gets higher as the cell size decreases. However, we need to account also for the size of the cell relative to the entire viewspace, since small cells require more computation to cover the entire viewspace. Figure 5(b) shows the results of dividing the cost effectiveness by the relative cell size. We see that cells smaller than 100 and larger than 50 are effective with a clear preference for a cell size of around 80.

Another interesting statistics is the "distance" of the strong occluder among the set of potential occluders. That is, for a given cell and a given object T, all the objects intersected by at least one of the rays emanating from the cell vertices towards the T's vertices are considered potential occluders. We have sorted the potential occluders and assigned a distance rank to the strong occluder. Table 3 is a histogram of the frequencies of the ranks with respect to a given cell size. This reflects the probabilities developed in Section 4. We can see that when the cell is small enough the strong occluder tends to be very close to the cell, and the probability of not having a strong occluder drops exponentially.

From our experience with our models we learned that constructing the visibility with a hierarchy of two levels of refinement is most efficient. The results are, of course, model dependent. However, from Figure 5(b) we know that computing cells of size 80×80 is most efficient for the given buildings in the model. Assuming we are interested in partitioning the space into cells of size 20. First computing the 80×80 cells and then refining them down to 20×20 cells is more cost effective than computing the 20×20 cells directly from the entire model. The following experiment confirms this (see Table 4). We have partitioned the above 400×400 playground into cells of size 20×20 and 80×80 directly from the entire model, and we have also refined the visibility of the 80×80 cells down to 20×20 in a second step. According to Table 4, the cost (in terms of ray/box intersections) of the two-level computation, accounting the cells relative size gives $400 \times 57207 + 25 \times 245076 \simeq 2.9 * 10^7$; three times better than a single level computation.

We have also tested the method over a 3D model of spheres. A 3D array of $30 \times 30 \times 30$ spheres was jittered to form a model of 27000 spheres. The radius of the spheres is 50 units. In the center of the model we removed one sphere and computed the visibility of different cells located in that region. The results are shown in Table 5. Here the visibility tests are applied to the bounding boxes of the spheres rather than directly to the spheres, since the algorithm requires a set of vertices for the ray shooting. On the other hand, the occluders are the exact spheres. Note that the depth complexity of the sphere's model is not as high as in the urban model. It is expected that in a model with a higher depth complexity the percentage of the strongly occluded spheres will increase significantly.

7. Conclusions

We have presented an efficient method for constructing the viewspace partition of densely occluded scenes. The resulting partition can be regarded as an approximation to the aspect graph in the following sense. When using aspect graphs

Table 1: The visibility as a function of the cell size

cell size	200	150	125	100	50	10
potentially visible polygons	24534	20470	16995	11338	2540	1447
strongly occluded building	89	241	438	974	2947	3126
strongly occluded boxes	11	39	100	337	2918	3116
ray/box intersections	132703	171463	208280	273029	222611	211715
ray/polygons intersections	1175813	1346399	1385075	1204630	109210	48002

Table 2: The visibility as a function of the cell size (cont'd)

cell size	90	80	70	60	50	40	30	20
potentially visible polygons	7745	5464	4090	3128	2540	2115	1833	1615
strongly occluded building	1806	2329	2623	2829	2947	3019	3067	3102
strongly occluded boxes	1456	2120	2521	2778	2918	2994	3052	3088
ray/box intersections	255315	245076	234380	227892	222611	218785	215838	212943
ray/polygons intersections	707462	425117	259442	159599	109210	85313	64846	55621
time (sec)	47	33	25	18	16	15	14	13

for image synthesis we do not need to maintain the two-dimensional view (aspect) in each cell, but rather the set of visible objects from that cell. The actual construction of the view can be efficiently carried out by a special purpose mechanism (e.g., z-buffer). Exact aspect graphs are difficult to construct and extremely costly in space and time. Our viewspace partition could be used for the same purposes as the aspect graph, but its construction is far more efficient and requires considerably less storage and preprocessing time. Since we are allowed to overestimate, we can use very simple operations, and in densely occluded scenes the overestimation is relatively very small.

As mentioned above, our method requires the occluders to be convex[5, 6, 7]. However, in practice most objects are not convex, but one can decompose a general polyhedral into convex parts. Since the decomposition need not be exact, we are now investigating several techniques by which an object can be represented as a union of convex shapes. The goal is to automatically find the union of a small number of convex bodies, which on the one hand can be intersected by rays easily, and on the other hand their union is as large as possible, but still included in the interior of the original object.

Acknowledgment

This work was supported in part by a grant from the Israeli Ministry of Science and the French Ministry of Research and Technology (AFIRST). The authors would like to thank Anat Sakov and Gernot Schaufler for reading the manuscript and providing useful suggestions. Work on this paper by Dan Halperin has been supported in part by an Alon Fellowship, by ESPRIT IV LTR Project No. 21957 (CGAL), by the USA-Israel Binational Science Foundation, by The Israel Science Foundation founded by the Israel Academy of Sciences and Humanities, and by the Hermann Minkowski – Minerva Center for Geometry at Tel Aviv University.

References

1. J.M. Airey, J.H. Rohlf, and F.P. Brooks, Jr.. Towards image realism with interactive update rates in complex virtual building environments. *Computer Graphics (1990 Symposium on Interactive 3D Graphics)*, 24(2):41–50, March 1990.

2. P. K. Agarwal and M. Sharir, On the number of views of polyhedral terrains, *Discrete and Computational Geometry* **12**, 177–182, 1994.

3. M. de Berg, D. Halperin, M.H. Overmars and M. van Kreveld, Sparse arrangements and the number of views of polyhedral scenes, *International Journal of Computational Geometry and Applications*, **7**, 175–195, 1997.

4. D. Cohen-Or and E. Zadicario, Visibility Streaming for Network-based Walkthroughs, Graphics Interface'98, June 1998.

5. S. Coorg and S. Teller. Temporally coherent conservative visibility. In the *Proceedings of the 12th ACM Symposium on Computational Geometry*, 1996.

6. S. Coorg and S. Teller. Real-time occlusion culling for models with large occluders. In the *Proceedings of the 1997 Symposium on Interactive Graphics.*.

7. T. Hudson et al. Accelerated occlusion culling using shadow volumes. In *the Proceedings of 13th ACM Symposium on Computational Geometry*, Nice, France, June 4–6, 1997

Table 3: A histogram of the distance of the strong occluder as a function of the cell size

	Distance										
Cell size	1	2	3	4	5	6	7	8	9	10	> 10
200	0	0	1	0	0	0	1	2	1	2	5
150	0	0	2	1	1	3	6	5	3	5	13
100	32	30	46	51	51	44	29	23	15	6	10
70	829	571	359	284	184	103	78	37	22	14	40
50	1470	635	377	185	111	68	32	15	11	1	13
30	2045	630	231	86	32	10	7	4	4	2	1
20	2345	504	160	45	19	5	5	2	2	1	0

Table 4: A two-step hierarchical computation of the visibility sets is more cost effective than a direct computation. Average numbers are displayed (M = Million).

cell size	20x20	80x80	20x20(80x80)
potentially visible polygons	1615	4090	1615
strongly occluded boxes	3102	2329	3102
ray/box intersections	212943	245076	57207
ray/polygon intersections	55621	425117	41076
time (sec)	15	26	5
total number of cells	400	25	400 + 25
total cost	85.2M	6.1M	22.9M + 6.1M

8. D.J. Kriegman and J. Ponce, Computing exact aspect graphs of curved objects: Solids of revolution, *International Journal of Computer Vision* **5** 119–135, 1990.

9. S. Teller and C.H. Sequin. Visibility preprocessing for interactive walkthroughs. In *Proc. of ACM Siggraph*, Pages 61–69, 1991.

10. F. Sillion, G. Drettakis, and B. Bodelet, Efficient impostor manipulation for real-time visualization of urban scenery. In *Eurographics '97*, 207–218, September 1997.

11. A.S. Glassner. *An Introduction to Ray Tracing.* Academic Press, Inc, San Diego, California, 1989.

12. E.A. Haines and J.R. Wallace. Shaft culling for efficient ray-traced radiosity. *Proceedings of the second Eurographics Workshop on Rendering,* Barcelona, Spain, May 13–15, 1991.

13. H. Plantinga and R. Dyer. Visibility, occlusion, and aspect graphs. The international journal of computer vision, 5(2):137–160, 1990.

14. H. Plantinga. Conservative visibility preprocessing for efficient walkthroughs of 3D scenes. *Graphics Interface'93*, 166–173, 1993.

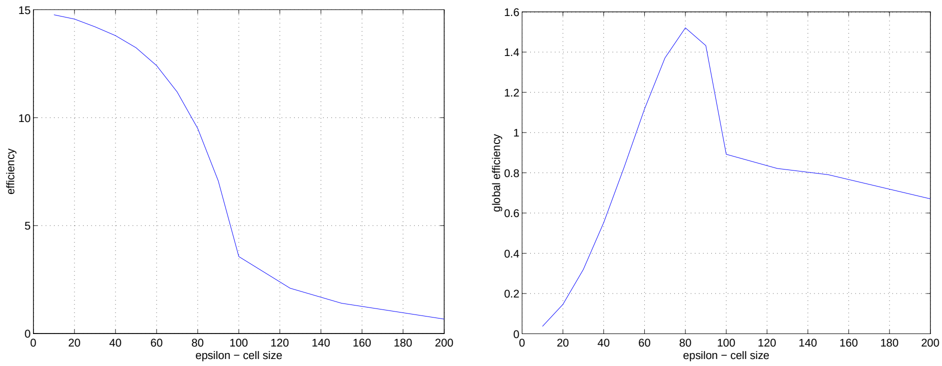
15. I. Shimshoni and J. Ponce. Finite resolution aspect graphs of polyhedral objects. in *IEEE Trans. Patt. Anal. Mach. Intell.* Vol 19(4), 315–327, April 1997.

16. R. Yagel and W. Ray. Visibility computations for efficient walkthrough of complex environments. *Presence*, 5(1):1–16, 1996.

17. Y. Chrysanthou, Shadow Computation for 3D interactive and animation, Ph.D. thesis, *Department of Computer Science, College University of London*, January 1996.

Table 5: The visibility of the 3D sphere scene.

cell size	50	40	30	20	10	5
ray/sphere intersections	463204	558874	678895	819622	971410	1052048
strongly occluded spheres	119	811	2334	4611	7481	9169
potentially visible spheres	26880	26188	24665	22388	19518	17830



(a) (b)

Figure 5: *(a) the cost effectiveness of a single cell size, and (b) the cost effectiveness of the aggregate of cells.*

Appendix A: Proof of Lemma 4.1

To simplify the presentation, we normalize all lengths by setting $b = 1$ and calculate the corresponding probabilities for the interval $I_1 := [-1 \quad 1]$ to be covered by K intervals of length $2m_b$ which are uniformly distributed in $[-2m_b - 1 \quad 2m_b + 1]$. Let the k-th interval be $[a_k \quad b_k]$ where $k = 1, \ldots, K$. Then $\hat{P}_1$ can be broken into three disjoint components:

1. All K intervals are to the left of 1. Since the distributions of the intervals locations are uniform and independent, the probability of this event is
$$P(\max b_k < 1) = (P(b_1 < 1))^K = \left(\frac{1}{m_b + 1}\right)^K .$$
2. All K intervals are to the right of -1. The probability of this event is
$$P(\min_k a_k > -1) = \left(\frac{1}{m_b + 1}\right)^K .$$
3. At least one interval is to the left of 1, another interval is to the right of -1 and I_1 is not fully covered. We note that since $b_k \in [-1 \quad 1 + 2m_b]$, the probability density for b_{k_0} to be located at α is $1/(2m_b + 2)$. Therefore, the probability of this event is
$$P\left(\begin{matrix} \exists \quad -1 < \alpha < \beta < 1, \exists k_0 \neq k_1, \\ b_{k_0} = \alpha, a_{k_1} = \beta, (\alpha \quad \beta) \cap (\cup_k [a_k \quad b_k]) = \emptyset \end{matrix}\right) =$$
$$\int_{-1}^{1} d\beta \int_{-1}^{\beta} d\alpha \binom{K}{1} \binom{K-1}{1} \left(\frac{1}{2m_b + 2}\right)^2 \left(\frac{2 - (\beta - \alpha)}{2m_b + 2}\right)^{K-2}$$
$$= (K-1)\left(\frac{1}{m_b + 1}\right)^K .$$

Summing the above probabilities gives
$$\hat{P}_1 = (K+1)\left(\frac{1}{m_b + 1}\right)^K .$$
The calculation of $\hat{P}_2$ is much simpler, as
$$\hat{P}_2 = [P(a_1 > -1 \text{ or } b_1 < 1)]^K = \left(\frac{4}{2m_b + 2}\right)^K .$$

Appendix B: Proof of Lemma 4.2

As in the 2D case,
$$\hat{P}_2 = [P(\text{first sphere does not fully cover } B_b \;)]^K .$$
In addition, a circle with radius R completely covers $B_b \iff$ its center is inside the circle B_{R-b}. Since the center of the circle is uniformly distributed in B_{b+R}, the probability for it not to fully cover B_b is
$$1 - \frac{\pi(R-b)^2}{\pi(R+b)^2} = 1 - \frac{(m_b - 1)^2}{(m_b + 1)^2} = \frac{4m_b}{(1 + m_b)^2} .$$

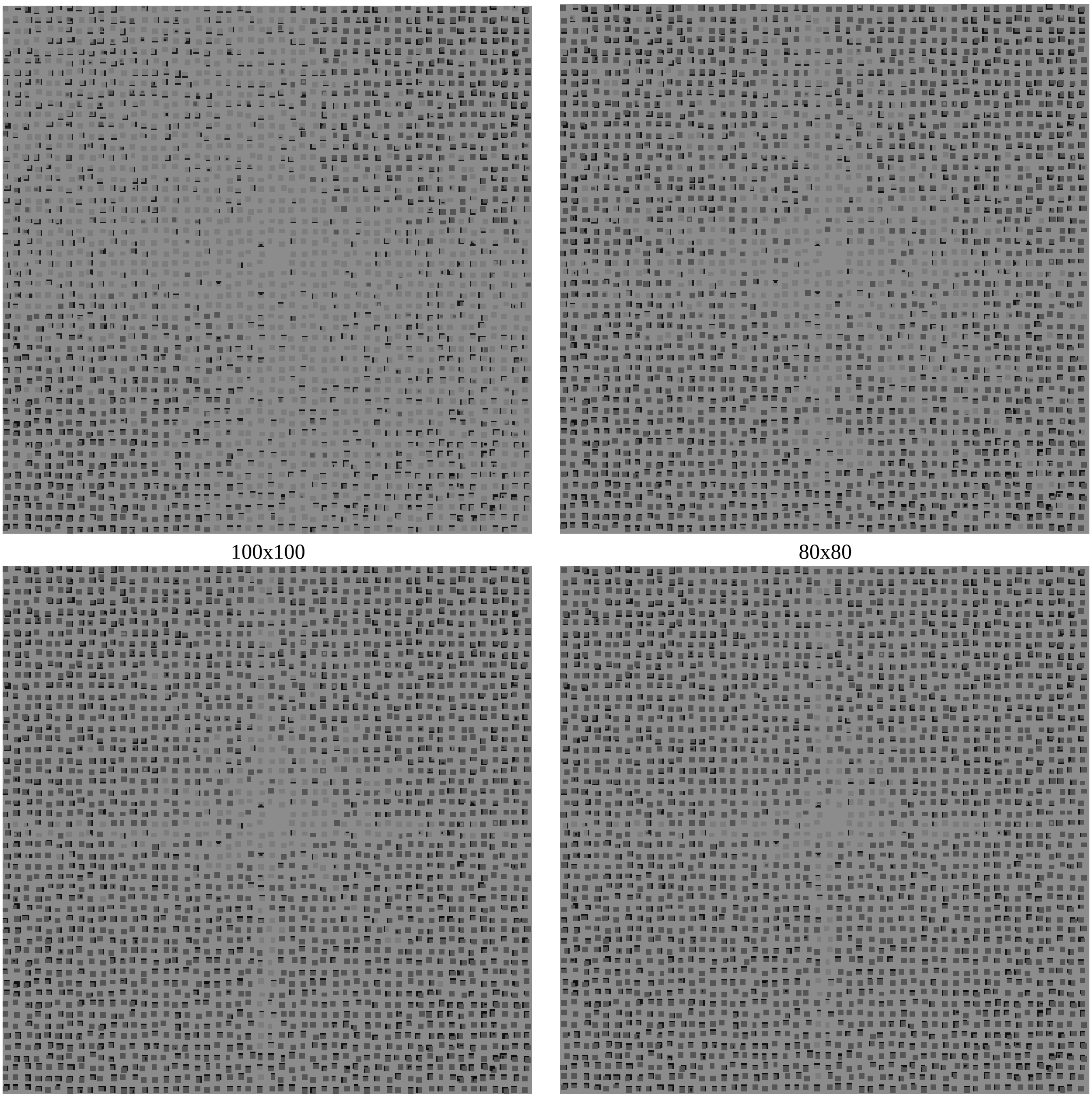


Figure 6: *A top view of the city model. The potentially visible buildings are colored in green, and the strongly occluded buildings are colored in red. See the replicated figures in the color section.*