Eurographics Symposium on Rendering 2003
Per Christensen and Daniel Cohen-Or (Editors)

Erosion Based Visibility Preprocessing

Xavier Décoret[1], Gilles Debunne[2] and François Sillion[2]

[1]MIT LCS [2]Artis, INRIA[†]

Abstract

This paper presents a novel method for computing visibility in 2.5D environments based on a novel theoretical result: the visibility from a region can be conservatively estimated by computing the visibility from a point using appropriately "shrunk" occluders and *occludees. We show how approximate, yet conservative, shrunk objects can be efficiently computed in an urban environment. The technique provides a tighter potentially visible set (PVS) compared to the original method in which only occluders are shrunk. Finally, theoretical implications of the shrinking theorem are discussed, opening new research directions.*

1. Introduction

When generating images from a given viewpoint, hidden surface removal has to be performed. The widely used *z*-buffer technique achieves this by projecting and rasterizing all primitives onto the image plane, retaining for each pixel only the closest one. Although the final result is correct, many primitives are processed only to be later discarded. If it were possible to identify these primitives, it would be possible to perform the rendering faster by safely ignoring them. A classical approach employs *from-region* visibility determination. Given a 3D model of an environment and a region of space, determine what is visible from at least a viewpoint in that region. The *potentially visible set* (PVS) associated with a region can be used for any viewpoint within that region. However, finding what is hidden from a region is a very difficult problem. In this paper we present a novel theoretical result about from-region visibility. We show that it can be estimated from one point in the region using shrunk versions of both occluders *and* occludees. The presented theorem is an extension of an existing result [26] and transforms the from-region visibility problem from a continuous domain into a discrete one that is much easier to solve.

Section 2 reviews previous work and outlines some issues. Section 3 presents the theorem and its implications. We detail in Section 4 how to compute approximate shrunk versions of both occluders and occludees. Section 5 tackles implementation issues and the results are presented in Section 6. Extensions and limitations of our method are finally discussed in Section 7.

2. Previous work

In this review, we focus on previous work related to from-region visibility. We refer the reader to Cohen-or et al. [6] and Pantazopoulos and Tzafestas [20] for a more in-depth survey. Teller et al. [23] propose pre-processing visibility through sequences of portals, an approach suitable for architectural scenes. Schaufler et al. [21] can handle scenes where occluders are watertight objects. The important class of 2.5D models has received special attention, since urban environments – which many applications consider – can be approximated by such models [13]. Wonka et al. offer efficient solutions [25, 26] for these environments. Downs et al. [9] further approximate buildings with Convex Vertical Prisms for efficient horizon culling. Bittner et al. [4] present an exact solution to the 2D version of the problem along with an effective extension to the 2.5D case. More recently, Leyvand et al. [17] have extended this class of environment and coined the term *3D*-ε scenes for those whose vertical complexity is much lower than its horizontal one.

Pre-processing *vs* dynamic visibility. Initially, from-region visibility was proposed as a pre-processing framework which can be seen as a form of caching. The navigable space is covered with a finite set of regions, called *viewcells*. the PVS for those viewcells are determined during a pre-processed computation. At run-time, the viewcell containing the current viewpoint is retrieved, and its PVS is ren-

[†] ARTIS is team of the GRAVIR/IMAG laboratory, a joint effort of CNRS, INRIA, INPG and UJF.

dered in place of the whole scene. In that context, the time required to compute visibility is not the main issue. The problem is the computation and the storage of the pre-computed PVS. More recently, Wonka et al. [27] have shown that when the computation of the PVS is fast enough, it can be used for *instant visibility*: the PVS is dynamically computed for a region surrounding the current viewpoint. The computation cost is amortized over the set of frames where the user remains in this region. In such an approach, the PVS corresponding to rather large regions should be computed quickly.

Conservativity. The methods discussed here guarantee that any object actually visible is classified as potentially visible. In other words, the PVS contains the *visible set* (VS). This property is referred to as *conservativity*. The PVS might contain objects that are actually not visible, which will be handled by the final hidden surface removal. However, the PVS must be computed as tightly as possible in order to optimize this removal. Comparing the *over-conservativity* of different methods is a difficult task since implementations are not always available. However, it is valuable to estimate the over-conservativity by analyzing the type of occlusion that the method is able to detect, as well as the worst-case and best-case scenarios.

Occluder fusion. Although early work has focused on the occlusion caused by single, large, convex occluders [8, 15, 7], current research focuses on the combined action of several potentially small occluders. Convex occluders have indeed been shown to be useful only if they are larger than the region for which visibility is determined [18]. *Occluder fusion* must be done in order to detect significant occlusion. Note that occluder fusion is implicitly performed by most from-points methods, such as the z-buffer, where projected occluders are aggregated to form an *occlusion map* against which objects can be tested for visibility. The extended projection of Durand et al. [10] defines a projection of occluders and occludees that conservatively maintains the visibility properties from a region. Wonka et al. [26] showed that this can also be achieved by shrinking the occluders. Schaufler et al. [21] work in object space, constructing large shadow volumes by extending discretized occluders through occluded regions. Brunet et al. [5] extract convex silhouettes from generic non-convex occluders in the case of from-point visibility. Koltun et al. [16] construct virtual occluders that are equivalent to a set of occluders from the point of view of a region. Other methods consider the ray space directly. The set of rays blocked by occluders have to be fused. If the resulting set of rays contains all the rays joining the region and an object, the latter can be declared hidden. The work by Nirenstein [19] proposes such an exact approach. Other works rely on a parameterization and a conservative discretization of the ray space (either in 2D [4] or in 3D-ε environments [17]).

The method presented here achieves a conservative from-region visibility pre-computation, computed in image space.

3. Shrinking based visibility

Wonka et al. [26] introduced the idea of shrunk occluders. They observe that if a ray SR is blocked by a "shrunk" occluder $\mathcal{O}_d$, then any ray $S'R$ with $|SS'| < d$ is blocked by the original object $\mathcal{O}$ (notations are those of Fig. 1). The d-shrinking is defined by $\mathcal{O}_d = \{M \mid \mathcal{B}_d(M) \subset \mathcal{O}\}$ where $B_d(M)$ is a sphere of radius d and center M. To determine wether an object is occluded by some occluders from a given viewcell, the approach fits the viewcell into a sphere $\mathcal{S}(S,d)$ and simply tests if the object is occluded by d-shrunk occluders from point S. The from-region query is therefore reduced to a from-point query, for which many existing methods can be used. This approach is very appealing since it can handle many types of occluder fusion, and since from-point methods are easier to implement. Our work is built on this foundation and overcomes its main limitation.

The use of a sphere to shrink occluders implies that the viewcell should not be too long in a given direction in order to avoid over-shrinking occluders by a large sphere. An interesting extreme case is a line segment viewcell. Another issue is that spheres shrink occluders in "every direction". Consider the case of a thin wall: when the viewcell's half radius is larger than the wall thickness, the wall is shrunk to the empty set and causes no occlusion, though it obviously hides everything behind it.

To address this issue of *over-shrinking*, we generalize the shrinking by a sphere to the erosion by a *convex shape*. Doing so, we establish a theorem that reduces the from-region visibility query to a *finite* number of ray/scene intersections (Sec. 7.1). Wonka et al. reduced the region-region visibility queries to point-region queries. We go one step further by reducing them to point-point queries.

3.1. Erosion theorem

We first introduce some notation. Consider a volumetric object $\mathcal{O}$, and a set of vectors $\mathbf{X}$. The dilation of $\mathcal{O}$ by $\mathbf{X}$, also known as the Minkowski sum of both sets [22], is defined by the equation:

$$\mathcal{O} \oplus \mathbf{X} = \{M + \mathbf{x},\ M \in \mathcal{O} \text{ and } \mathbf{x} \in \mathbf{X}\}$$

$\mathbf{X}$ is commonly called the *structuring element.* Similarly, the erosion of $\mathcal{O}$ by $\mathbf{X}$ is defined by:

$$\begin{aligned}\mathcal{O} \ominus \mathbf{X} &= \{V \text{ such that } \forall \mathbf{x} \in \mathbf{X},\ V + \mathbf{x} \in \mathcal{O}\} \\ &= \{V \text{ such that } \{V\} \oplus \mathbf{X} \subset \mathcal{O}\}\end{aligned}$$

The erosion of $\mathcal{O}$ is the complement set of the dilation of $\mathcal{O}$'s complementary. This can be written:

$$\mathcal{O} \ominus \mathbf{X} = (\mathcal{O}^C \oplus (-\mathbf{X}))^C \qquad (1)$$

Given this notation, we prove (see Appendix A) the following theorem, illustrated by Fig. 1:

Theorem 1 (Occluder erosion) *If a segment* $[SR]$ *intersects*

$\mathcal{O} \ominus \mathbf{X}$, where $\mathbf{X}$ is a convex set of vectors, then any segment $[S'R']$ intersects $\mathcal{O}$ with $S' \in \{S\} \oplus \mathbf{X}, R' \in \{R\} \oplus \mathbf{X}$.

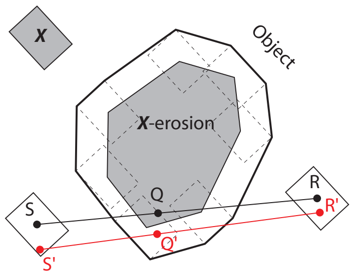

Figure 1: *If a ray $[SR]$ is blocked by the $\mathbf{X}$-erosion of an object, then any ray $[S'R']$ joining two points located in the convex $\mathbf{X}$-shaped regions around S and R is blocked by the object.*

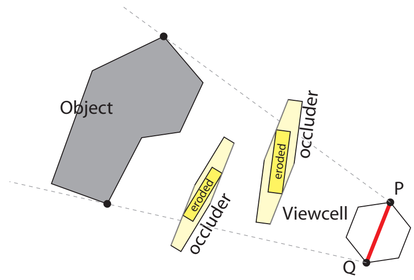

Figure 2: *Testing the visibility of the object from the viewcell is equivalent to testing visibility from segment $[PQ]$. But eroding occluders by the segment will not make them vanish as it would with the hexagon. The object will hence correctly be classified as invisible from the viewcell.*

3.2. Application to from-region visibility

Consider a convex viewcell, decomposed as $\{S\} \oplus \mathbf{X}$ (centered on S, with an $\mathbf{X}$ shape). Theorem 1 states that if object $\mathcal{R}$ is hidden by $\bigcup_i (\mathcal{O}_i \ominus \mathbf{X})$ from S then $\mathcal{R} \oplus \mathbf{X}$ is hidden by $\bigcup_i \mathcal{O}_i$ from $\{S\} \oplus \mathbf{X}$. In other words, whether an occludee is hidden by a set of occluders from the viewcell can conservatively be determined by testing if the "shrunk" occludee is hidden by the "shrunk" occluders.

This may seem odd at first since we are used to *over-estimating* occludees (see for example extended projections [10]), whereas in our case the tested occludee $\mathcal{R}$ is actually "smaller" than the occludee $\mathcal{R} \oplus \mathbf{X}$ for which we determine visibility. The occluder/occludee distinction is however still present since their "shrunk" versions differ (as will be detailed in Section 4).

The main difference between this theorem and the one used by Wonka et al [26] are that objects are eroded by the actual viewcell shape (which no longer needs to be included in a bounding sphere) and that *all* the objects of the scene (occludees *and* occluders) can be shrunk.

3.3. Tighter PVS using segment erosion

Theorem 1 allows us to significantly reduce the amount of over-shrinking. Let us consider a convex viewcell and the viewcell-object shaft as shown on Fig. 2. The shaft always rests on a maximal supporting segment $[PQ]$, and if any ray between $[PQ]$ and the object is blocked, then any ray between the viewcell and the object is alos blocked. Therefore, testing the visibility of the object from the viewcell is the same as testing its visibility from the segment. In the latter case, however, Theorem 1 would require eroding occluders only by a segment instead of an hexagon, preventing the two thin occluders of Fig. 2 from disappearing.

3.4. Zonotope decomposition

The viewcells used for erosion in Theorem 1 must be convex (otherwise they can be replaced by their convex hulls). However the actual shape of the viewcell may lead to several potential optimizations. If the viewcell is a 2D flat region, then Theorem 1 states that no erosion of the objects has to be performed along the "vertical" direction in order to compute from-region visibility. This result is not obvious when thinking about all the parallax effects that can happen along this direction as the user moves.

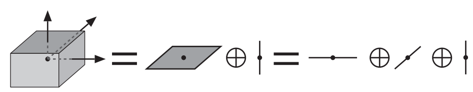

Figure 3: *Minkowski decomposition of a parallelepiped as the (commutative) sum of 3 segments.*

Another interesting and common case is when the viewcell is a *zonotope*, or the Minkowski sum of line segments (such as parallelepipedic boxes, as illustrated by Fig. 3). The erosion by the viewcell can then be replaced by *successive* erosions by line segments, making the process much easier. Indeed, and as proven in Appendix B, we have:

$$\mathcal{O} \ominus (\mathbf{X} \oplus \mathbf{Y}) = (\mathcal{O} \ominus \mathbf{X}) \ominus \mathbf{Y}$$

If the viewcell is the extrusion of a 2D contour, the same decomposition can be applied to separate the horizontal erosion by the contour and the vertical one by a segment (as is done in Section 4.3).

4. Shrinking objects

Computing exact erosion for general objects is a very difficult task [11,3]. We detail in this section how to compute approximate erosions for both the occluder and the occludee, so that the visibility conservativity property is satisfied.

4.1. Urban environments

Urban environments are a major field of application for PVS pre-computation algorithms. The viewcells will pave the streets of the city, and the potentially visible set of buildings that has to be displayed will be updated depending on the observer's current viewcell. Buildings can in general be quite tightly approximated by 2.5D blocks [13]. In our case, we suppose that these parts can be described by a polygonal "contour" and a height. Note that a contour does not need to be at ground level (*aka* footprint), as we can deal with arches or buildings made of superposed blocks, for example.

4.2. Bounding volumes

Instead of working with the objects' true geometry, it is possible to work with simplified versions. This idea of using levels of detail for visibility computation is common, since small geometric features rarely influence visibility. If for each object $\mathcal{O}$ of the scene we have an inner $\underline{\mathcal{O}}$ and an outer $\overline{\mathcal{O}}$ bound such that $\underline{\mathcal{O}} \subset \mathcal{O} \subset \overline{\mathcal{O}}$, then the following result holds: an occludee is hidden by occluders if (from the center of the viewcell) the "shrunk" version of its *outer* bound is hidden by the erosions of the occluders' *inner* bounds.

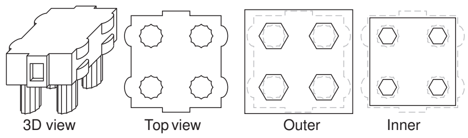

Figure 4: *Inner and outer 2.5D approximations of a building.*

Fig. 4 shows a complex building which is not itself 2.5D and contains facade details (windows, balcony), together with its inner and outer approximations by 2.5D polyhedra. For the purpose of clarity, we now consider that buildings are actually composed of such 2.5D polyhedra, but the reader will remember that we actually refer to its inner polyhedra when it is used as an occluder (Section 4.3) and to its outer polyhedra when the building is used as an occludee (Section 4.4).

4.3. Shrinking occluders

As stated in Section 3.4, when the viewcell is an extrusion (or is flat), we can separate the erosion using the contour as a 2D structuring element from the vertical erosion of the building. We consider this case in this section, focusing on the contour erosion since vertical erosion is simply a modification of the building height.

To compute the $\mathcal{O} \ominus \mathbf{X}$ erosion of $\mathcal{O}$, we rely on equation (1). We translate the two end points of each edge of the contour by the opposite ($-\mathbf{X}$) of the structuring element and compute the resulting convex hull. These "edge convex hulls" are unioned and the desired erosion is the polygon difference with the original contour (computed using the *kbool* library†). Fig. 5 illustrates this method for two examples.

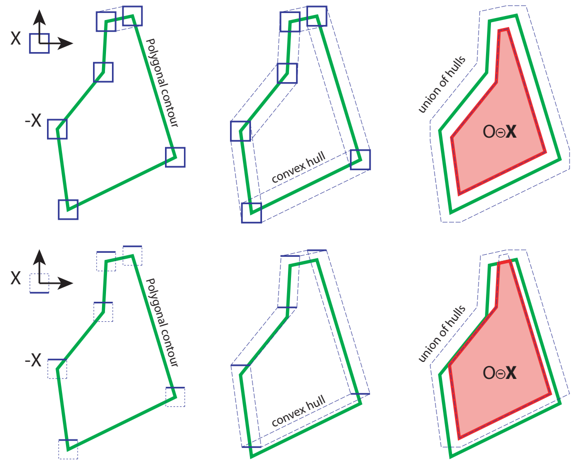

Figure 5: *Computing the erosion of a contour by a square or a segment not containing the origin. You can see from the bottom line that eroding by the top or the bottom segment of the viewcell yields different erosions.*

It is worth noting that the computed erosion is exactbecause we use a polygonal structuring element. Erosions by a sphere would also include rounded parts that need to be approximated by line segments [26]. We can speak here of exact mathematical "erosion" as opposed to the term "shrinking" that is used in the rest of this article and will be explained in the next section.

4.4. Shrinking occludees

The shrinking of occludees is slightly more complex. In order to be able to apply Theorem 1, we need to ensure that every point of the original occludee $\mathcal{O}$ is in the $\mathbf{X}$-neighboring of the shrunk occludee $\mathcal{R}$. In other words, we need to have $\mathcal{O} \subset \mathcal{R} \oplus \mathbf{X}$. Using $\mathcal{R} = \mathcal{O} \ominus \mathbf{X}$ generally does not satisfy this property as shown in Fig. 6.

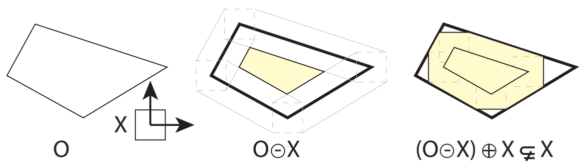

Figure 6: *Using $\mathcal{O} \ominus \mathbf{X}$ as a shrunk version ($\mathcal{R}$) of the object is not adequate as each point inside $\mathcal{O}$ is not guaranteed to be within a $\mathbf{X}$-neighboring of $\mathcal{R}$. Indeed, $\mathcal{R}$ has to satisfy $\mathcal{O} \subset \mathcal{R} \oplus \mathbf{X}$.*

Analytical exact shrinking. Computing a shrunk version $\mathcal{R}$ of an arbitrary 2D contour $\mathcal{O}$, such that $\mathcal{O} \subset \mathcal{R} \oplus \mathbf{X}$, can be done analytically when $\mathbf{X}$ is a segment. The algorithm is

† `www.xs4all.nl/~kholwerd/bool.html`

similar to the ones used to compute convex hulls. The vertices of $\mathcal{O}$ need to be sorted along a direction perpendicular to $\mathbf{X}$ and then swept while the width of the object is updated. A single line is created when this width is lower than $|\mathbf{X}|$, and it is split in two lines otherwise (see Fig. 7).

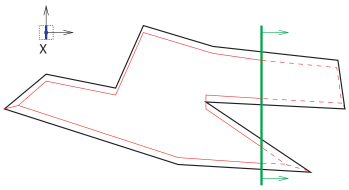

Figure 7: *Illustration of the analytical shrinking algorithm on a contour using a sweeping line parallel to the eroding segment.*

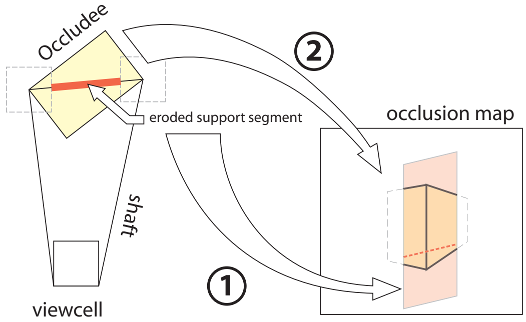

Figure 8: *Eroding in image space using a stencil mask. The supporting segment of the viewcell/occludee shaft is eroded and rendered as a quad in the stencil buffer (step 1). The occludee is then rendered and clipped by the stencil mask (step 2).*

This exact segment-shrunk occludee (extruded along the vertical direction) can be pre-computed. However, since it may be composed of numerous polygons, we propose an image based alternative.

Image based erosion using the stencil test. The shaft between the viewcell and the occludee is constructed. If this shaft is not degenerate, it is supported by a segment located on the occludee. This segment is $\mathbf{X}$-eroded and then transformed into a quad by raising it to a sufficient height. This quad is rendered in the stencil buffer to create a mask (see Fig. 8). The occludee itself is then rendered using this mask to clip it “on the sides”, thus creating a valid shrunk representation in image space.

Proof. The key to see that this computation is conservative is noticing that it is equivalent to testing the visibility of the occludee and the visibility of its projection onto a vertical quad raised from the supporting segment. Indeed, both have the same shaft and the erosion of the segment, hence of the quad, creates an $\mathbf{X}$-clipped projection of the occludee.

Two precautions must be taken. First, the eroded supporting segment must not be null to satisfy $\mathcal{O} \subset \mathcal{R} \oplus \mathbf{X}$. If it is null, a point centered on the supporting segment is considered instead. In other words, a vertical line instead of a vertical quad is rendered in the stencil buffer. Actually, this line is always rendered to overcome resolution issues when rendering the quad. The second precaution must be taken when the shaft degenerates. This occurs when, for example, we test the visibility of a street containing the viewcell. In this case, we simply use the occludee as a valid shrunk representation of itself.

This image-based algorithm works for flat viewcells. For extruded non-flat viewcells, and as stated in Section 3.4, all that needs to be done is to reduce the height of the rendered occludee by one half of the viewcell height.

For general 3D objects, this Minkowski segment decomposition, combined with a voxelized representation of the object, allows us to easily compute an approximated shrunk version. The voxels are successively and independently eroded along the three axis directions.

5. Implementation

The algorithm has two passes. During the first pass, each building is considered as an occluder. Its eroded (by the viewcell or by the shaft supporting segment) version is rendered in the frame buffer to create the occlusion map. From the center of the viewcell, all the possible viewing directions are sampled using 6 camera orientations directed toward the faces of a cube, each with a field of view of 90°.

The second pass determines the PVS by rendering all the buildings shrunk as occludees using the stencil buffer mask. The occludees are tested against the map, optionally using hardware occlusion queries. In that case, the number of visible *pixels* of each object can be retrieved. This image-based visibility estimation can be used for pruning large PVS or rendering buildings from a priority queue in a true real-time application [1].

5.1. Optimizations

All the buildings must be considered twice: first as potential occluders and then as possible occludees. However, we use hierarchical frustum culling [12, 2] to limit the number of rendered buildings. An important advantage of our image-based visibility algorithm is that any classical rendering optimization algorithm can be plugged in.

If all the viewcells have exactly the same shape (or a few different shapes), which is a fairly common case, then the eroded versions of the occluders can be pre-computed once and reused. Indeed, the number of different segments that can be used to erode the occluders is determined from the number of possible viewcell shaft-supporting segments. In the case of square viewcells, only 6 segments (the four sides and the two diagonals) need to be considered to represent all the shaft supporting segments. This leads to only 6 compact eroded representations of each building when they are considered as occluders. The one used depends on the observer's position. The number of representations depends on the shape of the viewcell (3 for a triangle, for instance) but is usually small as the shapes of the viewcells are typically simple.

Note that for square viewcells, the "left" and "right" segments of the square cannot be considered as identical eroding elements (same for the "top" and "bottom" segments). For consistency reasons, all the buildings are rendered from the *center* of the viewcell, and these two segments are hence different structuring elements (see Fig. 5).

We do not perform any occluder selection, and each object is rendered to create the occlusion map and then tested against this map. Since this is done 6 times per viewcell, hierarchical occlusion queries could clearly accelerate this process a lot for very large environments, since distant parts could very quickly be culled as a whole. However, hierarchical queries require serialization whereas hardware accelerated occlusion queries are more efficient when performed in parallel. This trade-off is still an open issue. Moreover, the occlusion map construction could be interleaved with the occlusion queries so that hidden objects do not need to be rendered. A front-to-back rendering technique could be used, for example, a Kd-tree as in Leyvand et al. [17].

6. Results

It is uncertain which numbers are relevant to present visibility results. The amount of geometry culled away can for example be made arbitrarily high by adding hidden primitives. A more relevant figure is the *over-conservativity* of the method – that is the ratio between the PVS size and the actual VS size. Measuring these sizes in terms of number of polygons is biased by the objects' discretization, so we chose to count the number of visible *objects*.

Different algorithms were tested on a city model of Vienna made of 458 buildings, 1182 streets and 726 crossings. Computed PVS are subsets of these entities. Over-conservativity is estimated and averaged from a representative set of square viewcells. We compared different methods, for increasing sizes of viewcells:

1. Occluders are eroded by a sphere (approximated by an octagon), as in [26]. Occludees are rendered as is.
2. Occluders are eroded by the viewcell (a square).
3. Occluders are eroded by segments as described in 3.3.
4. Same as (3), but occludees are also reduced (see 4.3).

As expected, Fig. 9 shows that the over-conservativity grows with the viewcell-size. However, we can see that this growth is slower using our approach: when both occluders *and* occludee are shrunk, a tighter PVS is obtained as predicted. The bottom curve on Fig. 9 shows the ratio of the PVS sizes for methods 1 and 4, exhibiting the gain of our method. The PVS reduction is maximum when segments (method 3) are used instead of viewcells (method 2) to shrink objects.

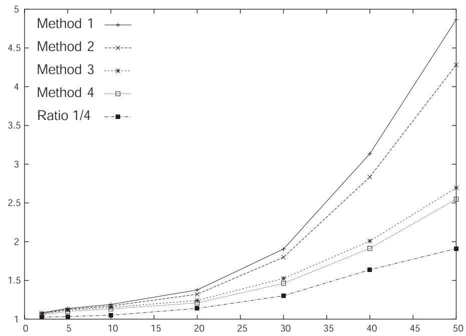

Figure 9: *Comparison of over-conservativity for the different erosion-based methods, according to the viewcell size. The curves show the averaged ratio between PVS and VS sizes. The VS was computed from a dense sampling of the viewcell. The lower curve is the ratio between method 1 and method 4.*

The average viewcell PVS computation time is 0.5s on a 2 GHz Pentium 4 processor with a 128 Mb NVidia GeForce 4 Ti4600 graphics card, making overnight pre-computation (100,000 viewcells) possible for typical urban environments (rendering was not optimized). Pre-computation of eroded buildings takes negligible time compared to rendering.

The 3D models used for pre-computation were simple, but remember that we compute the visibility using bounding volumes of the objects and that the complexity of the displayed geometry can be made arbitrarily high.

7. Discussion

We now discuss some of the strengths and limitations of the proposed method.

Sampling. Our approach relies on sampling, since visibility is estimated by using a rasterization of the occluders in the frame buffer. However this sampling is not an issue since it is performed in image space, where its results will be used. The impact of the discretization is not always clear for methods that sample a different parameter space. By choosing a frame buffer resolution at least equal to r times the resolution at which the scene will be navigated, where r is a ratio dependent on the viewcell's size and accounting for the fact that we sample from the center, we can ensure the conservativity of the approach for the center of the viewcell. The r ratio can be determined by a simple geometric calculation on the considered viewcell shape.

On-the-fly queries. The computational cost is not much of an issue when visibility is pre-computed. However, for very large environments, viewcells tend to become unusable since they require a large amount of storage. As seen in previous work, from-region visibility computations can, however, be used in the context of *instant visibility* [27], in which case the timing and the ability to handle large viewcells becomes crucial. We believe that our approach is suitable for these cases since there is ample room for optimization and large viewcells still provide a good PVS.

Scaling to large environments. The database size has a great influence on PVS pre-computation times. However, once the horizon is completely occluded, further buildings, no matter their number, are all occluded. Our algorithm is linear in complexity as each object is considered exactly twice. Using hierarchical approaches, either in image space [14, 28] or in object space (Kd-tree such as in [17]), could lower this complexity to scale to very large environments.

Bad scenario. Though we showed that larger viewcells can be used with our approach, the Theorem 1 still suffers from the fact that the viewcell cannot be too big compared to the average size of the occluders. In that case, the occluders have empty erosions and no occlusion is detected.

Viewcells When determining the optimal sizes and positions of viewcells as well as the trade-off between their number and the over-conservativity we can afford are important issues. Adjacent and small viewcells that (almost) share the same PVS can recursively be merged. This can be done afterward using PVS compression techniques as described by Panne et al. [24]. Another possible hierarchical algorithm could start with large viewcells, which are subdivided when the union of the PVS of children differs too much from the parent's. This post treatment will benefit from the conservativity and the tighter PVS of our method.

7.1. X-sampling objects

We finally introduce the notion of $\mathbf{X}$-*sampling* of an object $\mathcal{O}$ as a set of points whose $\mathbf{X}$-dilation contains $\mathcal{O}$ (see Fig. 10). As noted before (see Fig. 6) $\mathcal{O} \ominus \mathbf{X}$ is generally *not* an $\mathbf{X}$-sampling, but $\mathcal{O}$ itself is trivially one. We then have the following theorem:

Theorem 2 (X-sampling) *If $\mathcal{O}$ is bounded and $\mathbf{X}$ is a convex open set, then there exists an* $\mathbf{X}$-sampling *of $\mathcal{O}$ composed of a finite set of points $\mathcal{I} = \{I_1, \ldots, I_n\}$*

A naive tiling of space proves the result, however finding an *optimal* set of points $\mathcal{I}$ is a complex task in the general case.

The two theorems yield a very surprising result. To test if $\mathcal{O}$ is hidden by a set of occluders $(\mathcal{O}_k)$ from any viewpoint in the viewcell, it is sufficient to test if the *rays* $[SI_j]$ (see Fig. 10) are blocked by $(\mathcal{O}_k \ominus \mathbf{X})$. As surprising as it may sound, the from-region visibility query can be conservatively answered by computing the intersection of a finite number of sampling rays with eroded occluders! This result may prove very interesting for complex 3D visibility queries and opens new research directions.

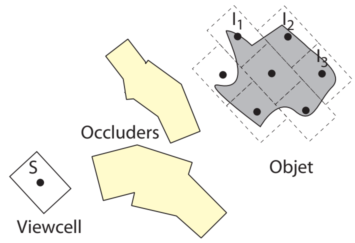

Figure 10: *Testing the visibility of the object from the viewcell can be done by testing intersection of rays SI_j with eroded occluders.*

8. Conclusion & future work

We introduced a novel theoretical result on visibility, stating that occluders *and* occludees can be both shrunk to compute from-region visibility. Some erosion and shrinking properties were proven and used to conservatively compute the shrunk representations of the objects. We demonstrated that tighter PVS are generated with this approach.

We believe that the theoretical results presented offer possibilities for further applications. As shown in Section 7.1, conservative visibility can be computed for a region by performing a finite number of ray-scene intersections. We plan to take advantage of this in a ray-tracer for accelerating the shadow computations of an extended light source. Another application of this ray testing would be for small moving objects (such as cars). If their bounding box is smaller than the current viewcell, visibility can be conservatively determined by testing a single ray joining their centers.

Another field of investigation is determining how to compute erosions and $\mathbf{X}$-samplings in 3D. For convex objects, this can be done in image space using hardware-accelerated convolutions. "Apparently convex" occluders could also be used as in Hoops 3D [5]. Another interesting property is that in some cases, $\mathcal{O} \ominus \mathbf{X}$ is a valid shrunk occludee. We are investigating this property for segment erosion. If this property is true for each object in the scene, then a theoretical one-pass algorithm is possible where each eroded object is rendered in the frame buffer from the center of the viewcell. Those that are visible in the final image form the PVS.

References

1. Carlos Andújar, Carlos Saona-Vázquez, Isabel Navazo, and Pere Brunet. Integrating occlusion culling with levels of detail through hardly-visible sets. In *Comp. Graphics Forum (Proc. of Eurographics)*, volume 19(3), pages 499–506, 2000.

2. Ulf Assarsson and Tomas Möller. Optimized view frustum culling algorithms for bounding boxes. *Journal of Graphics Tools*, 5(1):9–22, 2000.

3. Henk Bekker and Jos B.T.M. Roerdink. An efficient algorithm to calculate the Minkowski sum of convex 3d polyhedra. In *Proc. Computational Science - ICCS 2001*, pages 619–628. Springer, 2001.

4. Jiří Bittner, Peter Wonka, and Michael Wimmer. Visibility preprocessing for urban scenes using line space subdivision. In *Proc. of Pacific Graphics*, pages 276–284, 2001.

5. Pere Brunet, I. Navazo, C. Saona-Vázquez, and J. Rossignac. Hoops: 3d curves as conservative occluders for cell-visibility. In *Computer Graphics Forum (Proc. of Eurographics 2001)*, volume 20(3), 2001.

6. Daniel Cohen-Or, Y. Chrysanthou, Claudio Silva, and Fredo Durand. A survey of visibility for walkthrough applications. *IEEE Trans. on Visualization and Comp. Graphics*, 2002.

7. Daniel Cohen-Or, Gadi Fibich, Dan Halperin, and Eyal Zadicerio. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. *Computer Graphics Forum*, 17(3):243–255, 1998.

8. Satyan Coorg and Seth Teller. Real-time occlusion culling for models with large occluders. In *Proc. of the 1997 Symposium on Interactive 3D graphics*, pages 83–90, 1997.

9. Laura Downs, Tomas Möller, and Carlo H. Séquin. Occlusion horizons for driving through urban scenery. In *Proc. of the 2001 Symposium on Interactive 3D Graphics*, pages 121–124, 2001.

10. Frédo Durand, George Drettakis, Joëlle Thollot, and Claude Puech. Conservative visibility preprocessing using extended projections. In *Proc. of SIGGRAPH'00*, pages 239–248, 2000.

11. E. Flato, E. Fogel, D. Halperin, and E. Leiserowitz. Video: Exact Minkowski sums and applications. In *Proc. 18th ACM Symposium on Computational Geometry*, pages 273–274, 2002.

12. B. Garlick, D. Baum, and J. Winget. Interactive viewing of large geometric data bases using multiprocessor graphics workstations. In *Parallel Algorithms and Architectures for 3D Image Generation*, 1990. Siggraph '90 Course Notes.

13. Rick Germs and Frederik W. Jansen. Geometric simplification for efficient occlusion culling in urban scenes. In V. Skala, editor, *Proc. of WSCG 2001*, 2001.

14. Ned Greene, Michael Kass, and Gavin Miller. Hierarchical z-buffer visibility. In *Proc. of SIGGRAPH'93*, pages 231–238, 1993.

15. T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang. Accelerated occlusion culling using shadow frusta. In *Proc. 13th ACM Symposium on Computational Geometry*, pages 1–10, 1997.

16. Vladlen Koltun, Yiorgos Chrysanthou, and Daniel Cohen-Or. Virtual occluders: An efficient intermediate PVS representation. In *Proc. of Eurographic Rendering Workshop*, 2000.

17. Tommer Leyvand, Olga Sorkine, and Daniel Cohen-Or. Ray space factorization for from-region visibility. In *ACM TOG, Proceedings of SIGGRAPH)*, volume 21(3), 2003.

18. Boaz Nadler, Gadi Fibich, Shuly Lev-Yehudi, and D. Cohen-Or. A qualitative and quantitative visibility analysis in urban scenes. *Computers and Graphics*, 23(5):655–666, 1999.

19. S. Nirenstein, E. Blake, and J. Gain. Exact from-region visibility culling. In *Proc. of the 13th Workshop on Rendering*, pages 191–202, 2002.

20. Ioannis Pantazopoulos and Spyros Tzafestas. Occlusion culling algorithms: A comprehensive survey. *Journal of Intelligent and Robotic Systems*, 35(2):123–156, 2002.

21. Gernot Schaufler, Julie Dorsey, Xavier Décoret, and François Sillion. Conservative volumetric visibility with occluder fusion. In *Proc. of SIGGRAPH*, pages 229–238, 2000.

22. Michel Schmitt and Juliette Mattioli. *Morphologie Mathématique*. Masson, Paris, 1993.

23. Seth J. Teller and Carlo Séquin. Visibility preprocessing for interactive walkthroughs. In *Computer Graphics (Proc. of SIGGRAPH'91)*, volume 25, pages 61–69, 1991.

24. M. van de Panne and J. Stewart. Efficient compression techniques for precomputed visibility. In *Proc. of Eurographics Rendering Workshop*, pages 305–316, 1999.

25. P. Wonka and D. Schmalstieg. Occluder shadows for fast walkthroughs of urban environments. In *Comp. Graphics Forum (Proc. of Eurographics)*, volume 18(3), pages 51–60, 1999.

26. Peter Wonka, Michael Wimmer, and D. Schmalstieg. Visibility preprocessing with occluder fusion for urban walkthroughs. In *Proc. of Eurographic Rendering Workshop*, 2000.

27. Peter Wonka, Michael Wimmer, and François Sillion. Instant visibility. In *Computer Graphics Forum (Proc. of Eurographics 2001)*, volume 20(3), pages 411–421, 2001.

28. Hansong Zhang, Dinesh Manocha, Tom Hudson, and Kenneth E. Hoff. Visibility culling using hierarchical occlusion maps. In *Proc. of SIGGRAPH'97*, pages 77–88, 1997.

Appendix A: Proof of the shrinking Theorem 1

Notation is that of Fig. 1. Let $S' \in \{S\} \oplus \mathbf{X}$ and $R' \in \{R\} \oplus \mathbf{X}$. By definition, there exists $\mathbf{x}_S \in \mathbf{X}$ and $\mathbf{x}_R \in \mathbf{X}$ such as:

$$S' = S + \mathbf{x}_S \qquad R' = R + \mathbf{x}_R$$

By hypothesis, there is a point Q on segment $[SR]$ which is in $\mathcal{O} \ominus \mathbf{X}$. For this point, there is $t \in [0,1]$ such as:

$$Q = (1-t)S + tR$$

We construct $Q' = (1-t)S' + tR'$. It is a point in $[S'R']$, and we have:

$$\begin{aligned} Q' &= (1-t)(S+\mathbf{x}_S) + t(R+\mathbf{x}_R) \\ &= (1-t)S + tR + (1-t)\mathbf{x}_S + t\mathbf{x}_R \\ &= Q + \mathbf{x}' \end{aligned}$$

with $\mathbf{x} = (1-t)\mathbf{x}_S + t\mathbf{x}_R$. Since $\mathbf{X}$ is convex, we clearly have $\mathbf{x} \in \mathbf{X}$. Since $Q \in \mathcal{O} \ominus \mathbf{X}$, we also have $Q + \mathbf{x} \in \mathcal{O}$. Thus segment $[S'R']$ intersects $\mathcal{O}$ in Q', which makes the proof.

Appendix B: Proof of associativity

Given a point M in space, we have the following equivalences, which proves $\mathcal{O} \ominus (\mathbf{X} \oplus \mathbf{Y}) = (\mathcal{O} \ominus \mathbf{X}) \ominus \mathbf{Y}$:

$$\begin{aligned} M \in (\mathcal{O} \ominus \mathbf{X}) \ominus \mathbf{Y} &\iff \forall \mathbf{y} \in \mathbf{Y}\ M + \mathbf{y} \in \mathcal{O} \ominus X \\ &\iff \forall \mathbf{y} \in \mathbf{Y}\ \forall \mathbf{x} \in \mathbf{X}\ M + \mathbf{y} + \mathbf{x} \in \mathcal{O} \\ &\iff \forall \mathbf{z} \in \mathbf{X} \oplus \mathbf{Y}\ M + \mathbf{z} \in \mathcal{O} \\ &\iff M \in \mathcal{O} \ominus (\mathbf{X} \oplus \mathbf{Y}) \end{aligned}$$