EFFICIENT APPROXIMATE VISIBILITY TESTING USING OCCLUDING SPHERES[1]

László Szécsi, László Szirmay-Kalos

Budapest University of Technology and Economics,
Magyar Tudósok krt. 2., Budapest, H-1117, Hungary
E-mail: szecsi@iit.bme.hu

ABSTRACT

Practically every ray-casting based image synthesis algorithm relies heavily on the tracing of shadow rays to determine visibility between a sample point on a light source and a surface point. If the light source is relatively large, or indirect illumination is sampled, the necessary amount of such visibility tests may be several magnitudes more then the number of primary rays. In this paper, we discuss an approach using simple spherical occluders to approximate the shadowing effect of scene objects. We elaborate on how spheres allow for an extremely fast evaluation of visibility, and offer a complex automatic method for generating good sphere sets. The virtual light sources or indirect photon mapping technique, and the handling of large light sources are improved using the new method. Results featuring fast global illumination are presented.

Keywords: global illumination, virtual light sources, indirect photon mapping, occluders, Delaunay

1 INTRODUCTION

Taking global illumination algorithms to interactive speeds is a prevailing objective of recent research. Solutions either require immense computing power, or limit themselves to simple geometric scenes and light transport phenomena. Finite element approaches feature tessellation artifacts that are not easily removed and, with exceptions, are limited to diffuse materials. Performance of ray-casting based algorithms is restrained by the number of rays we are able to trace for an image. This number is too small even for the purposes of real-time ray-tracing, and very far from the needs of proper global illumination.

In this paper, we will deal with global illumination problems focusing on large light sources and indirect illumination. Radiation of both light sources and reflective surfaces will be represented by point samples. These point samples for indirect illumination are generated using the indirect photon mapping method[WKB+02].

In photon mapping techniques, light transfer information is stored for paths originating from light sources in the form of photon hits. These photon hits compose a random approximation of the radiance within the scene. Direct techniques try to estimate the radiance at a point using nearby photon hits. Indirect photon mapping, on the other hand, completes the photon paths to reach the point of interest in order to tell the radiance there. This means, that photon hits work as point-like light sources, also referred to as *virtual light sources*. The method needs relatively few photons, but it is necessary to check if our point of interest and the photon hits can be connected. In indirect photon mapping, practically, all surfaces within the scene act as area light sources, sampled at the photon hit points.

The problem is very much similar to that of the large area light source, where a large number of samples is necessary to obtain a pleasing image. In both cases, in order to determine the illumination of a single surface point, numerous visibility tests between the point and the light samples have to be evaluated. In ray-casting based algorithms this is done by tracing a shadow ray from the point to the light source to tell if there is an occluding object in between. However, this means that the most time will be spent on shadow rays. On the other hand, both for large light sources or indirect photon hits, the resulting shadows will be smeared. Furthermore, accurate shadow contours corresponding to one light sample should never appear in the final image, as they would be perceived as disturbing artifacts. Fine details of the occluding objects will not influence the resulting image. Therefore, accurate shadow calculation is too much time wasted for superfluous information. Visually pleasing results can be achieved faster if shadows are approximated.

In this paper, we cut down the cost of visibility tests using a simplified representation of possible shadowing objects. Although this basic idea of using less detailed models for less important computations is commonplace in computer graphics, the question what model would suit the given task at best and how to find such a representation for the original model are always intriguing. We describe a strikingly fast algorithm to evaluate visibility based on a sphere set model, and a rather complicated technique involving optimisation heuristics and mesh generation to generate such sphere sets. Research done by Chrysanthou, Cohen-Or and Lischinski had a similar motivation[CCOL98]. Their concept is based on the adaptive discretisation of the 6 dimensional space of rays. That approach may be more general, as it is applicable for linear or flat objects not easily modelled by spheres. However, it requires the storage and traversal of a huge data structure. Furthermore, it appears to be difficult to extend the method to support dynamic scenes and animation.

2 SPHERICAL OCCLUDERS

In our choice of occluders, the most important factor is how fast the visibility test can be carried out for them. One of the natural choices is using spheres. They are not unrivalled in this field, using boxes or ellipsoids would also be possible. However, spheres have a feature no other geometric object has: when looked upon for any direction, they appear to be disks facing the viewer. Telling if a point light source is behind such a disk can be very easy.

Whenever the illumination of a surface point is to be found, visibility between that single point and every sample point on the light sources has to be checked. It is beneficial to calculate some kind of useful values for every occluder in advance. These values should allow for faster execution of those calculations that need to be carried out for every light sample. Using spheres as occluders allows for effective pre-processing. For every sphere, we may calculate the direction vector pointing to the centre, and the cosine of the angle γ (Figure 1):

$$\cos\gamma = \frac{\sqrt{t^2 - R^2}}{t}.$$

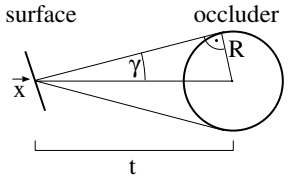

Figure 1: Nomenclature for pre-processing occluders

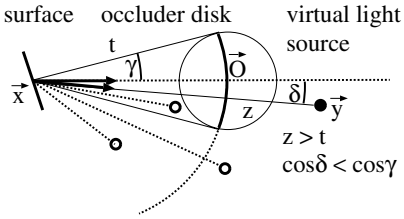

Figure 2: Occlusion test for occluder disk

For every virtual light source, occlusion testing will be fast (Figure 2). We need to calculate the direction towards the light source and its distance, also needed in illumination calculations and ray-casting. We take the dot product of the direction to the centre of the sphere and the direction to the light source:

$$\cos\delta = \frac{(\vec{O} - \vec{x}) \cdot (\vec{y} - \vec{x})}{zt}.$$

If $\cos\delta$ is less than $\cos\gamma$, the virtual light source is within the cone containing the sphere. As no accurate results are needed, the distance of the light source z may simply be compared to the distance t of the centre of the sphere. This actually means, that we use slightly bent, view-dependent

occluder disks instead of spheres. That way, we are able to carry out visibility tests at the cost of one dot product per occluder, instead of the cost of tracing a complete ray.

3 SPHERE GENERATION

It is not always a simple task to find a given number of spheres that represent the shadowing characteristics of a complex object. Even in case of a single sphere, it may not be trivial to find the perfectly matching radius. In order to develop an automated process, we first need to define what we mean by similar shadowing. Generally, we suppose that light rays may arrive from any point of space and any direction with equal probability. We try to find such a set of occluders that is hit by light rays with the same probability as the original object. In the convex case, this is proportional to the *surface area* of the object[Sto75]. However, neither the set of spheres, nor the original object is guaranteed to be convex.

Our approach is to create a very accurate spherical representation, and simplify it gradually until a desired number of occluders has been reached. The initial set has practically the same shadowing characteristics as the original object. Simplification is carried out in such a manner that the probability of the set being hit by a general light ray is preserved.

How to create a spherical representation for an arbitrary object using surface points is described by Hubbard[Hub96]. An ideal continuous set of infinite number of spheres could match the volume of any object perfectly. The set of their centres defines the mid-surface of an object. Centres of approximating spheres should be placed on this surface to obtain a well-fitting representation. However, finding the mid-surface is also not easy. Three-dimensional Delaunay meshing provides the solution. A Delaunay mesh is a decomposition of space into tetrahedra, the circumspheres of which cannot contain any node of the mesh. Using the Bowyer-Watson incremental algorithm[Fil96], we can build a mesh whose nodes are surface points of the object. This is an incremental algorithm, starting off form an initial enclosing tessellation, adding points sequentially. As a result, we should obtain a set of tetrahedra. Some of these are within the boundaries of the object, and some are just part of the initial tessellation we used for the algorithm, or correspond to concavities of the object surface. The inner ones are making up the object. How to select them is discussed in Section 3.1. Circumspheres of these tetrahedra do not include any of the specified surface points, but touch at least four of them. If surface points used for meshing were dense enough, these circumspheres are a very tight enclosing set of spheres, suitable to serve as an initial representation for our object.

3.1 Identifying inner tetrahedra

Unfortunately, the result of the mesh generation process will include tetrahedra external to the object. The circumspheres of external elements are usually large and their volumes do not coincide with the object. These spheres should be excluded from the representative set. Figure 3 explains the criterion used to select them. We calculate the surface normal for every point inserted into the mesh. We use the relation of these normals and the normal of the circumsphere at the same point to tell for every sphere if it is plausibly within the bounds of the surface. Whenever the angle between the sphere normal and the surface normal is not more than $\pi/2$ for all four nodes of the tetrahedron, it can safely be regarded to be interior. However, this approach supposes a consistent, gradually changing set of surface normals. In case of non-manifold surfaces where the normals may be poorly defined or have discontinuities, further or different filtering may be necessary.

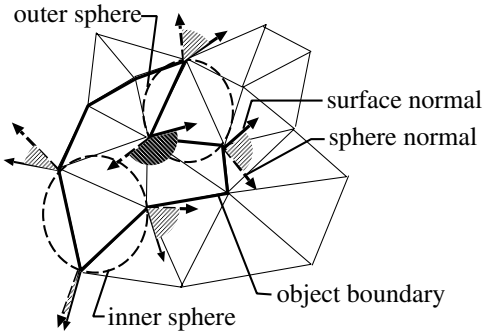

Figure 3: The obtuse angle between the surface and sphere normals identifies an outer sphere

3.2 Sphere merging

The article by Hubbard[Hub96] proposes a gradual refinement approach for the simplification of a sphere set, but for the purpose of collision detection. QSplat[RL00] also uses a hierarchy of

bounding spheres. Both methods generate an enclosing set of spheres that tends to grow large as spheres decrease in number. For our needs, such a representation is not suitable, as it would cast larger shadows than the original object. However, our strategy is similar: identify an ideal pair of spheres, and merge them into one. There are two problems to be solved. First, a criterion for choosing pairs must be found. Second, a formula for the size and position of the substitute sphere of equal shadowing capacity must be provided.

An ideal merger should cause minimal change to the geometry. The two spheres should be close to each other in some sense, so that a single sphere is able to substitute them. There are two practical definitions for this distance of two spheres:

$$D_1 = d, D_2 = d + r - R,$$

where d is the distance of the centres, R and r are the radii, and $R > r$. D_2 can be given an intuitive interpretation. The radius of the larger sphere should be extended by D_2 to make the sphere include the smaller one. It is also equal to the Hausdorff distance between the larger sphere and the enclosing sphere of both spheres. While D_2 seems to be better established, it can be argued that D_1 promotes merger of small spheres better, and thus avoids that a single continuously growing sphere absorbs the smaller ones. Practically, there is little difference. Both heuristics choose suitable pairs. In both cases, a best pair for every sphere can be efficiently located using a spatial proximity search structure like a k-d tree.

Knowing the best pair for every sphere, the pair with minimum distance can be chosen, and merged into one sphere. Repeating this process will eventually take us to the desired number of occluders. However, executing a proximity search for every sphere, after every merger would make the algorithm very slow. Therefore, the best pair for every sphere is remembered, and updated only if it is necessary. This means, that the best pairs have to be found only for the newly created sphere, and those spheres whose best pairs were just merged. The spatial search acceleration structure should also be updated.

3.3 Radius of the new sphere

The new sphere substituting for the pair should have the same shadowing capability. Spheres are usually overlapping with other occluders, or they are surrounded by them. However, we may assume that other spheres influence the shadowing effect of the new sphere to the same extent as they did with its predecessors. Numerous configurations can be listed when this is not exactly true, but for practical cases it proves to work satisfactory. That way, the problem is simplified to the following: the new sphere should be hit by rays with the same probability as the union of its two predecessors. Still, the union of spheres is usually not a convex object, for which the surface area[Sto75] would be proportional to the probability. As a closed form integral geometry formula is ponderous to derive, it has to be approximated in some way.

3.3.1 Shadowing capability

Let $A(R, r, d)$ be the *generalised shadowing surface area* of the union of two spheres of radius R and r, with their centres located at a distance of d from each other. Generalised shadowing surface area, or GSSA, is defined as the surface area of any enclosing convex object multiplied by the conditional probability of a ray hitting the shadowing object, provided it hits the enclosing one. For convex objects, it is the same as the surface area. When merging two spheres, the new sphere should have the same shadowing capability as the union of its predecessors. Therefore it has to have the same GSSA:

$$4\pi R_{new}^2 = A(R, r, d), \quad R_{new} = \sqrt{\frac{A(R, r, d)}{4\pi}}.$$

In the following sections, we will present several methods to approximate $A(R, r, d)$.

3.3.2 Sum of surfaces

A straightforward idea is to add the surface areas of the spheres. This is always an overestimation, because it considers rays that hit both spheres twice. It may be applied if the two spheres are far from each other.

$$A_{\sum}(R, r, d) = 4\pi R^2 + 4\pi r^2.$$

3.3.3 Surface area of the union

For non-intersecting spheres it is identical to the previous estimate. For intersecting spheres:

$$a = \frac{R^2 - r^2 + d^2}{2d}, \quad b = \frac{r^2 - R^2 + d^2}{2d}$$

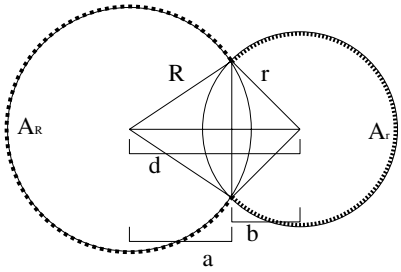

Figure 4: Calculation of the surface area of the union

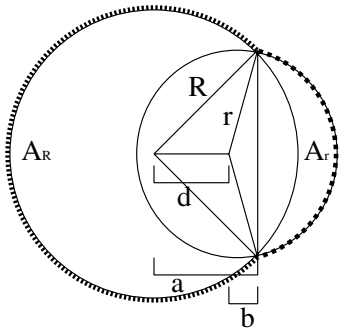

Figure 5: Calculation of the surface area of the union

There are two possible cases. If $a < d$, spheres are located like in Figure 4, and surface areas are as follows:

$$A_R = 2\pi R(R + a), \quad A_r = 2\pi r(r + b).$$

If $a > d$, spheres are located as in Figure 5, and surface areas are:

$$A_R = 2\pi R(R + a), \quad A_r = 2\pi r(r - b).$$

The surface area of the union is the sum of the individual sphere segment areas.

$$A_\cup(R, r, d) = A_R + A_r.$$

This is also an overestimation. It may be obtained by subtracting the surface area of the common volume from the summed surface area of the spheres. However, rays crossing both spheres but not the intersection are still accounted for twice.

3.3.4 Surface area of convex hull

When spheres are near, their convex hull fits the union tightly. This is also an overestimation, as rays crossing the hull but not the union are considered to be hitting. See Figure 6 for nomenclature:

$$S_r = 2\pi r^2(1 - \cos\alpha), \quad S_R = 2\pi R^2(1 + \cos\alpha),$$

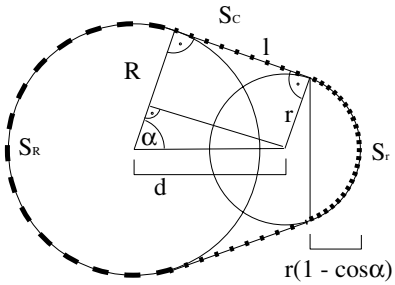

Figure 6: Calculation of the surface area of the convex hull

$$l = \sqrt{d^2 - (R - r)^2}.$$

The surface area of the conical segment is:

$$S_C = \pi l(r + R)\sin\alpha,$$

$$A_{convex}(R, r, d) = S_R + S_C + S_r.$$

For intersecting or almost touching spheres, this estimator fits more tightly than either above. To handle every case, the minimum of the three should be taken. Note that the first two will be identical for non-intersecting spheres.

3.3.5 Measurement using Monte-Carlo integration

The algorithm may tolerate overestimation well, but the result could be slightly larger occluders than needed. This could force us to correct the radii by some arbitrary factor. Therefore, we opted to construct an approximate function for $A(R, r, d)$. The surface area of an object scaled by s changes by the factor of s^2. To see if this is true for GSSA, let us consider its definition in Section 3.3.1. If an object is scaled by s, its enclosing convex object may also be scaled by the same factor, retaining the enclosing property. The surface area of the enclosing object will be scaled by s^2, just like the GSSA of the original object. Therefore, it is true that:

$$A(R \cdot s, r \cdot s, d \cdot s) = s^2 \cdot A(R, r, d).$$

Using this we may state:

$$A(R, r, d) = R^2 \cdot P(1, r/R, d/R),$$

where P denotes the probability of the spheres being hit by a ray. Conclusively, it would be enough to know values of $P(1, r/R, d/R)$, where $r/R \in (0, 1]$ and $d/R \in [0, \infty)$. Furthermore, cases where $d/R + r/R < 1$ are trivial, because the large sphere contains the small one.

We carried out measurements for $P(1, r/R, d/R)$, for a large number of different values of r/R and d/R. Random rays were generated by selecting two random points on the surface of the enclosing sphere of the two spheres, and connecting them. The number of rays that had hit either of the spheres was recorded, and the probability was approximated as the ratio of the number of hits and the number of rays shot. The GSSA of the two spheres is:

$$A_{approx} = \frac{nHits}{nRays} \cdot \text{surface area of enclosing sphere.}$$

100000 rays for every pair of parameters were shot. We used a curve fitting utility, to find a closed form approximating function:

$$A_{approx}(1, r/R, d/R) =$$

$$1.1824 \cdot 0.42615^{R/d} \cdot r/R^{1.972} + 0.991$$

This makes the general estimator:

$$A_{approx}(R, r, d) = R^2 A_{approx}(1, r/R, d/R).$$

3.4 Centre of the new sphere

The substitute should be placed so that it is near the union of the spheres, or between the two if they are at a distance. The new centre is a weighted average of the two previous centres, where the weighting corresponds to shadowing capability of the respective spheres. The quantities S_R and S_r defined in Section 3.3.4 are meaningful measures of this, and can be used as weights.

3.5 Occluder generation results

Using the algorithms described above, we were able to generate visually plausible sphere representations. In Figure 7 and Figure 8, different detail level occluder representations of two models are shown. As it would be less challenging to find a sphere set for a stocky convex object, both the dragon and the torus knot are concave and exhibit a complicated shadow pattern. In the figures, shadows cast by small light sources may be compared to see differences between full ray-casting and the occluder solution. While actual contours are irrelevant, the area of the shadows is preserved as much as possible. The images were rendered at $600x600$ pixels resolution, with the indirect photon mapping method, but using shadow rays for visibility tests. 500 photons and 100 direct light samples per pixel were used.

For the dragon scene (108588 triangles), the reference image was rendered in 26.95 seconds, and sphere generation took 2.28 seconds irrespective of the number of occluders. For the torus knot scene (2880 triangles), the reference image was rendered in 43.7 seconds, and sphere generation took 3.92 seconds irrespective of the number of occluders. Ray-casting was accelerated using a k-d tree. An AMD Athlon XP 2600+ running at 2.08 GHz with 1 Gbyte RAM was used.

Figure 7: Original dragon model, and occluder representations using 1000, 250, 100, 25 and 10 spheres

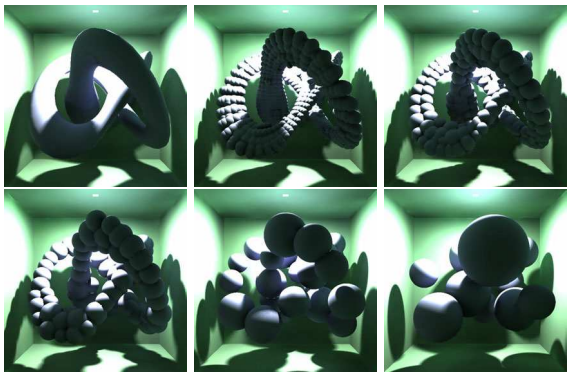

Figure 8: Original torus knot model, and occluder representations using 1000, 250, 100, 25 and 10 spheres

4 RENDERING

4.1 Occluder search structures

The basic mechanism of using occluders in visibility tests was introduced in Section 2. Following the idea of spatial subdivision schemes for ray-casting, the number of tests may be reduced by checking possible candidates only. However, as a test itself is nothing more than calculating a dot product and a comparison, the support methods should also be very lightweight to pay off.

There are two classes of occluders that can be trivially neglected: those behind the plane of the surface element on which our point of interest is and those further than the light. The rest may be processed linearly. Although linear complexity is obviously not the best that is algorithmically possible, it may be suitable for a low number of occluders. It is very much likely that typical applications will trade accuracy for speed and use few occluders.

If occluders are arranged into a spatial subdivision hierarchy like a k-d tree, the complexity of traversal is tuned down to a logarithmic level. However, the overhead cost of searching the structure may be comparable with the time gain. Our measurements, using a generic k-d tree implementation versus a fine-tuned occluder code, showed that it needs about 800 spheres for the logarithmic method to be on par with the simple linear one. The results are depicted in Figure 9. The same test scene as in Figure 8 was used, the 600×600 pixels image was rendered using 500 photons in the indirect photon mapping method, without direct illumination. The test machine in this case was an AMD Athlon 950 MHz with 128 Mbyte memory.

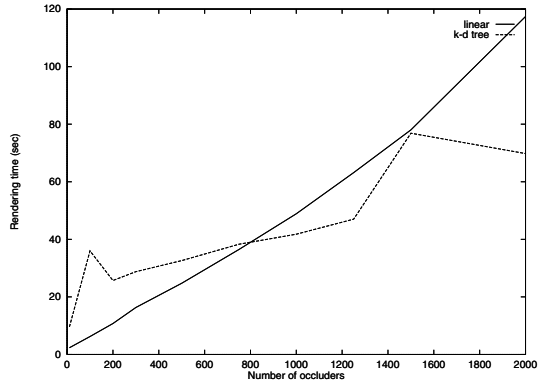

Figure 9: The performance of the simple linear and the k-d tree based methods

4.2 Self shadowing

Although the use of occluders may stay unnoticeable in most cases, it can cause very disturbing artifacts when it comes to self-shadowing. Because spheres may intersect the surface of the original object, sharp edges may appear even on completely smooth surfaces. There are several options to handle the problem. First, self-shadowing may simply be neglected. In order to achieve this, the occluders of the object on which our point of interest is should be excluded from the visibility tests. This may be a legal decision if the object is convex, or we know that self-shadowing is negligible. Even for a complex object, especially when only indirect illumination is concerned, the introduced error may remain unnoticeable.

The second, absolutely conservative approach is to carry out the intersection test for the object and the shadow ray. In case of a complex model, this may still be very expensive. If the object were the only complex object within the scene, we would practically return to the classic shadow ray solution.

The method offering both proper self-shadows and retaining the speed of the occluder approach has to select which spheres actually represent a part of the object that would cast a shadow, and which spheres are blocking the light just because they do not follow the object surface. The choice can be based on the distance of the sphere. It proved to be a proper heuristic to exclude spheres the centre of which was at a smaller distance from the surface point than the double of their radius. At such a distance the shadowing effect of the sphere should be meagre enough to allow for seamless clipping, but real occluding parts of the object are probably further away. Naturally, the accuracy of self-shadowing depends on the accuracy of the sphere representation, and, consequently, on the number of spheres. Figure 10 compares the shadows produced by the conservative approach, and the heuristic one at different numbers of occluders. Self shadowing errors are most obvious within the mouth of the dragon. The images were rendered at $600x600$ pixels resolution, with the indirect photon mapping method. 500 photons and 100 direct light samples per pixel were used. The model consisted of 108588 triangles. The image in the upper left was rendered in 52.25 seconds, using classical ray-casting for direct, and occluders for indirect illumination. The rest of the images were rendered using occluders for both direct and indirect illumination, applying the double radius heuristics. They feature 200, 150 and 50 occluders, and were rendered in 46.32, 23.00 and 16.20 seconds, respectively. An AMD Athlon XP 2600+ running at 2.08 GHz with 1 Gbyte RAM was used.

4.3 Rendering results

Using occluders, rendering of scenes with the indirect photon mapping method could be accelerated by an order of magnitude. Furthermore, significant change of image quality was only observable

Figure 10: Dragon with approximated direct illumination and self shadowing.

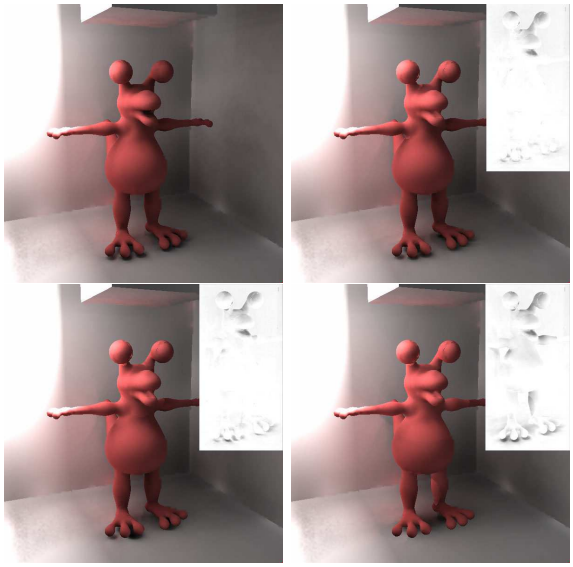

Figure 11: Indirect photon mapping without and with occluders, and inverted difference images

when it came to precise self-shadowing or casting shadows on close surfaces. Figure 11 compares result images. The image in the upper left was rendered using shadow rays, and the rest were rendered using 150, 50 and 10 occluders. The model consisted of 33094 triangles. Only the indirect visibility tests were accelerated. The four images were rendered in 52.08, 21.6, 10.34 and 5.48 seconds, respectively. Other circumstances were identical to those listed above. As the number of virtual light sources, and, consequently, the ratio of shadows rays was selected to be extremely high, the time cost of tracing primary rays is negligible. Therefore, results of measurements considering the shadow computation only would be very similar.

5 FUTURE POSSIBILITIES

Our current implementation uses the virtual light sources method in itself. However, in case of specular or refractive surfaces continuing the gathering random walk could be more appropriate. Furthermore, caustic effects should be handled by direct photon mapping. The composition of different techniques could lead to a fast, but general rendering solution.

The enormous power of recent graphics hardware is not easily used to support ray-casting algorithms. However, visibility tests for occluder disks are simple enough for the algorithm to be compiled onto a graphics accelerator.

ACKNOWLEDGEMENT

This work has been supported by the National Scientific Research Fund (OTKA ref. No.: T042735), and Bolyai Scholarship.

REFERENCES

[CCOL98] Yiorgos Chrysanthou, Daneil Cohen-Or, and Dani Lischinki. Fast approximate quantitative visibility for complex scenes. In *Proceedings of CGI '98*, pages 220–227, 1998.

[Fil96] M. Filipiak. Mesh generation. Technical report, Edinburgh Parallel Computing Centre, 1996.

[Hub96] M. P. Hubbard. Approximating polyhedra with spheres for time-critical collision detection. *ACM Transactions on Graphics*, 15(3):179–210, 1996.

[RL00] Symon Rusinkiewicz and Mark Levoy. QSplat: A multiresolution point rendering system for large meshes. *Siggraph 2000 Proceedings*, pages 343–352. 2000.

[Sto75] L. Stone. *Theory of Optimal Search*. Academic Press, New York, 1975.

[WKB+02] Wald, Kollig, Benthin, Keller, and Slusallek. Interactive global illumination. Technical report, Saarland University, 2002. http://www.openrt.de/Publications.