GPU-based Ambient Occlusion and Indirect Illumination

László Szirmay-Kalos, Balázs Tóth, Tamás Umenhoffer, Mateu Sbert
Budapest University of Technology and Economics

1. Introduction

Computer games and real-time systems require realistically looking images at high frame rates. The realism may be provided by an accurate simulation of the physics laws, but this approach is too time consuming and is not fully supported by graphics processors (GPU). Thus in practice, we prefer approximations that can be efficiently evaluated on the GPU. A usual simplification is the *local illumination model*, which computes only the direct contribution of the light sources and adds a constant ambient term for the missing indirect illumination. However, constant ambient lighting ignores the geometry of the scene, which results in plain and unrealistic images (Figure 1).

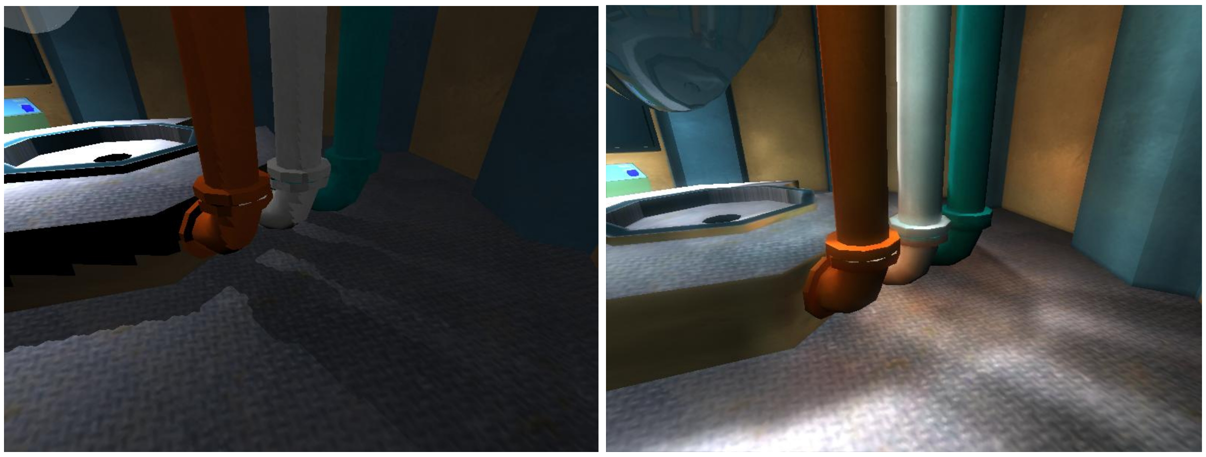


Figure 1. Comparison of the local illumination model (left) and a physically plausible global illumination solution (right).

We need better compromises that approximate indirect lighting in a more believable way but are not significantly more difficult to compute than the local illumination model. Such a flexible compromise is the *local indirect lighting* approach that examines only a neighborhood of the shaded point during illumination calculation. The *obscurances method* [Zhukov98, Iones03], which is also called the *ambient occlusion* [Landis02, Pharr04] computes just how "open" the scene is in the neighborhood of a point, and scales the ambient light accordingly. The openness of the point is proportional to the solid angle of directions where no nearby occluders exist. Obtaining the radiance of the occluders somehow, they can be treated as indirect light sources, so even color bleeding effects can be simulated [Mendez03, Umenhoffer09, Rithschel09].

Local approaches require only the consideration of the geometry in the neighborhood of the point instead of the complete geometry of the scene as happens in physically precise models, which is a great advantage. However, even the consideration of the neighborhood poses problems since the GPU processes points and triangles independently of each other, thus when a point is processed, we normally have no information about other surface elements. This problem is solved by using the depth image created by the GPU and considering it as a sampled representation of the scene geometry. This trick is known as the *screen-space method* [Mittring07].

In this article, we present a simplified illumination model which is derived from the rendering equation and develop a computation algorithm that can be efficiently executed by the GPU. The model consists of two parts, the ambient occlusion part describing the influence of the far part of the scene, and the indirect illumination part, which is responsible for local interactions. Both parts are directional integrals, and we discuss and compare different alternatives for their accurate estimation.

2. An indirect illumination model

Let us assume that the surfaces are diffuse. According to the rendering equation, the *reflected radiance* L^r in *shaded point* $\vec{s}$ can be obtained as an integral of directions $\vec{\omega}$ running over the unit hemisphere Ω above the surface:

$$L^r(\vec{s}) = \int_{\Omega} L^{in}(\vec{s},\vec{\omega}) \frac{a(\vec{s})}{\pi} \left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right)^{+} d\omega$$

where a is the *albedo* of the diffuse surface, $\vec{N}_{\vec{s}}$ is the unit normal at the shaded point, and $L^{in}(\vec{s},\vec{\omega})$ is the incident radiance of the shaded point from direction $\vec{\omega}$. If the incident angle is greater than 90 degrees, then negative scalar product $\left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right)$ should be replaced by zero, which is indicated by superscript +.

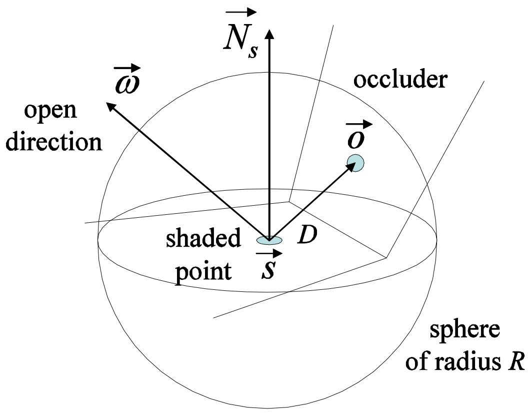


Figure 2. Shaded point $\vec{s}$ is the center of the neighborhood sphere. The radius of the sphere is R. Those directions $\vec{\omega}$ where there is no intersection closer than R are called open. Point $\vec{o}$ is an intersection closer than R.

If *occluder point* $\vec{o}$ is visible from $\vec{s}$ in direction $\vec{\omega}$ and the space is not filled with participating media, then the incident radiance is equal to exiting radiance $L^r(\vec{o})$. If no surface is seen, then shaded point $\vec{s}$ is said to be *open* in this direction, and the incident radiance is ambient intensity L^a. However, this does not meet our intuition and everyday experience that the effect of far surfaces is replaced by their average. This experience is due to that the real space is not empty but is filled by participating medium. If its extinction coefficient is σ and its albedo is 1, then the radiance along a ray of direction $\vec{\omega}$ changes according to the volumetric rendering equation:

$$L^{in}(\vec{s},\vec{\omega}) = \exp(-\sigma D) L^r(\vec{o}) + \left(1 - \exp(-\sigma D)\right) L^a ,$$

where D is the distance between the shaded and the occluder points. Note that in this equation factor $\mu(D) = (1 - \exp(-\sigma D))$ and its complement $1 - \mu(D) = \exp(-\sigma D)$ express the effects of the ambient lighting and of the occluder on the shaded point, respectively. The effect of the occluder diminishes with the distance. Function μ is a *fuzzy measure* that defines how strongly direction $\vec{\omega}$ belongs to the set of open directions based on distance D of the occlusion at this direction.

The exponential function derived from the physical analogy of participating media has a significant drawback [Iones03]. As it is non-zero for arbitrarily large distances, very distant surfaces need to be considered that otherwise have a negligible effect. Thus, for practical fuzzy measures we use functions that are non-negative, monotonically increasing from zero and reach 1 at finite distance R. This allows the consideration of only those occlusions that are nearby, i.e. in the *neighborhood sphere* of radius R.

Using the fuzzy measure, the reflected radiance of the shaded point can be expressed in the following way:

$$L^r(\vec{s}) = a(\vec{s})\left(L^a O(\vec{s}) + I(\vec{s})\right).$$

The first term of this expression is the *ambient occlusion* of the shaded point:

$$O(\vec{s}) = \frac{1}{\pi}\int_{\Omega} \mu(D(\vec{\omega}))\left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right)^{+} d\omega .$$

The second term is the irradiance due to nearby indirect lighting:

$$I(\vec{s}) = \frac{1}{\pi}\int_{\Omega} (1 - \mu(D(\vec{\omega})))L^r(\vec{o})\left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right)^{+} d\omega .$$

This integral is traced back to the ambient occlusion. Replacing occluder radiance $L^r(\vec{o})$ by the average of surface radiance values in the neighborhood of the shaded point, $\tilde{L}^r(\vec{s})$, we can express the irradiance as:

$$I(\vec{s}) \approx \frac{1}{\pi}\int_{\Omega} (1 - \mu(D(\vec{\omega})))\tilde{L}^r(\vec{s})\left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right)^{+} d\omega = \tilde{L}^r(\vec{s})\left(1 - O(\vec{s})\right).$$

Thus, our lighting model requires the calculation of the average surface radiance $\tilde{L}^r(\vec{s})$ and the ambient occlusion $O(\vec{s})$ of shaded point $\vec{s}$.

3. GPU based occlusion calculation

The ambient occlusion integral requires the determination of the distance of the close occluders in different directions. This is typically a ray tracing operation, which casts rays in different directions and tries to intersect the geometric elements inside the R-neighborhood and finally obtains the closest intersection. Unfortunately, ray-tracing is quite expensive computationally and is different from the rasterization based rendering that is fully supported by the GPU. A GPU processes elements like triangles, vertices, and fragments independently of other elements, which opens a lot of room of parallel computing. The results of different computational threads are merged in the compositing phase which uses the buffers storing the depth and the color of the closest points to the camera in each pixel, and allows summing of the color values via alpha-blending. It means that if a rendering pass already obtained the depth buffer, we have partial geometric information, i.e. the distance of the points visible from the camera.

3.1. Screen space representation of the visible geometry

Screen-space techniques assume that the height-field defined by the current content of the depth-buffer is an appropriate representation of the screen geometry, and the color buffer stores the radiance values of the represented points. Of course, the content of these buffers represents only the surfaces visible from the camera, but for local methods like the ambient occlusion, this restriction is usually acceptable. In screen-space viewing rays are parallel to axis z, which greatly simplifies calculations. However, the transformation to this space is not angle and distance preserving (it is not even affine), thus this space is not appropriate for angle and distance computation.

If we need to compute angles and distances, we should rather work in camera-space, which means that we store camera-space z values in textures and also the camera-space normal vectors of the visible points. The transformation from world-space to camera-space is angle and distance preserving since it is basically a translation and a rotation, thus these spaces are equivalent when distances and angles are computed. The disadvantage of camera-space is that in case of a perspective camera the viewing rays are not parallel, but intersect each other at the origin of the coordinate system.

Thus, to solve the distortion problem of screen-space but to keep the advantages of parallel rays, we work in camera-space but use a quasi-orthogonal approximation. When large-scale information is obtained, we follow the structure of the camera-space. However, when smaller neighborhoods are explored, which happens during the evaluation of the ambient occlusion integral, we assume that in this small neighborhood the viewing rays are parallel with axis z. This is an approximation, but is a reasonable compromise between accuracy and simplicity.

As the shaded point belongs to the set of points that are visible from the camera and the considered occluders are nearby, we may assume that the occluder point is also visible from the camera. Two points are visible from each other if the ray originated at one of the points has zero intersection with the surfaces before it arrives in the other point. The requirement of being in the visible part of the height field, on the other hand, means that the ray intersects the surface zero, two, four, etc. times. If the neighborhood is small, and at most one intersection is possible, then the two criteria are similar.

The second part of our illumination model depends on the average radiance values of nearby surface points. The color values of the frame buffer stored in the neighborhood of the current fragment can be used to obtain the average reflected radiance. By inspecting the camera-space normal, we can also check whether the surface is oriented toward the shaded point, and ignore it in the average otherwise.

3.2. Ray marching in the sampled geometry

Note that in the depth buffer we have just a discrete representation of the scene geometry as a collection of points. Instead of checking all elements, we can explore the neighborhood by ray marching that takes small steps along the ray and finds the first sample point that is occluded.

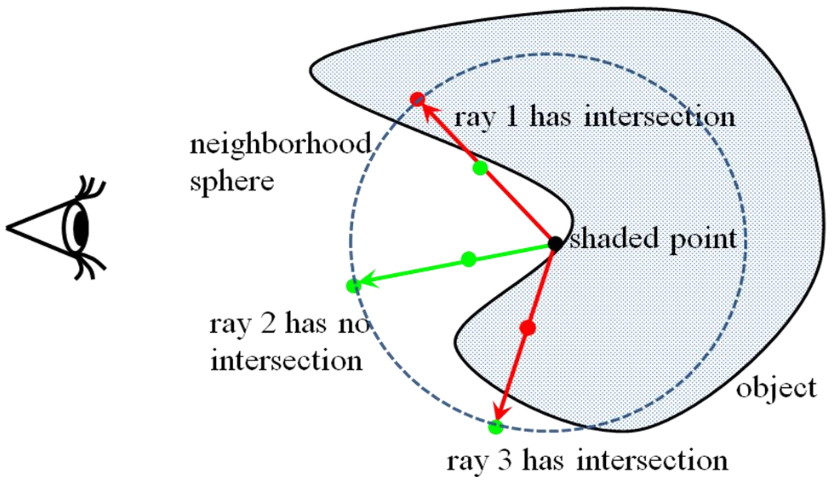


Figure 3. Computation of ambient occlusion by ray marching. Each ray is tested by points until an occlusion is found in the neighborhood sphere.

If we take *N* rays and each ray is marched with *M* steps, then altogether we have to test $N \cdot M$ points, which is prohibitive in most cases. The problem is that sample points that are uniformly distributed along rays are very non-uniformly fill the neighborhood sphere, which is responsible for high quadrature error unless the number of samples points is high. From another point of view, as we need only the first intersection, marching along a ray with small steps is the waste of computational resources since those points that are behind the first intersection do not provide useful information. We may think that breaking from the loop of ray marching when an intersection is found could solve this problem, but it does not. The problem is that the GPU is a quasi-SIMD parallel machine, thus it prefers to execute the very same machine instruction in parallel computational threads. Thus, if threads execute different number of cycles of a loop, the performance degrades.

The solution is to take very few, i.e. a single sample along each ray. First, we assume that the neighborhood is small with respect to the local curvature of the surface, so we can suppose that a ray may intersect the surface at most once. It means that if a sample point is not in the occluded region, then the ray has not intersected the surface yet. Thus, we ignore cases when the ray intersected the surface two, four, etc. in even number of times, and while starting at the unoccluded region, it visits the unoccluded region again.

If a single sample is taken per ray, and we check whether nor not this particular sample is occluded, then we can ignore the ray direction since the sample itself determines the direction, so it needs not be explicitly represented. Samples fill the volume of neighborhood sphere and their weighted sum provides the ambient occlusion estimate. Thus, these samples can also be interpreted as an estimate of a volumetric integral over visible points inside the neighborhood sphere [Szirmay10]. The questions are how the samples should be placed for effective evaluation and how the volumetric interpretation can be exploited to obtain a low-variance estimator. These questions are answered in the following two sections.

4. Volumetric ambient occlusion

In order to find a correspondence between the ambient occlusion integral and a volumetric integral inside the neighborhood sphere, we the ambient occlusion integral is extended to a three-dimensional integral. First the fuzzy measure is written as the integral of its derivative:

$$\mu(D) = \int_0^D \frac{d\mu(r)}{dr}\,dr\,.$$

The upper limit of this integral is the distance of the occlude in the given direction or R if no occluder exists in the neighborhood sphere. Including a *visibility function* $v(r,\vec{\omega})$ into the integrand, which is 1 if there is no occlusion closer than r in direction $\vec{\omega}$, and 0 otherwise:

$$\mu(D) = \int_0^D \frac{d\mu(r)}{dr}\,dr = \int_0^R \frac{d\mu(r)}{dr} v(r,\vec{\omega})dr\,.$$

Substituting this integral into the ambient occlusion formula, we get

$$O(\vec{s}) = \frac{1}{\pi}\int_\Omega\int_0^R \frac{d\mu(r)}{dr}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)^+ v(r,\vec{\omega})drd\omega\,. \qquad (1)$$

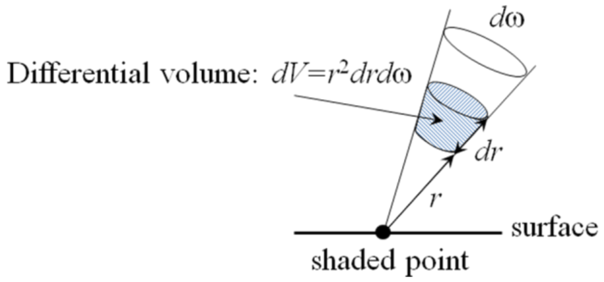


Figure 4. Computation of ambient occlusion by ray marching. Each ray is tested by points until an occlusion is found in the neighborhood sphere.

Realizing that $r^2 drd\omega = dV$ is a differential volume (Figure 4), the ambient occlusion can also be expressed as a volumetric integral

$$O(\vec{s}) = \frac{1}{\pi}\int_S \frac{d\mu(r)}{dr}\frac{1}{r^2}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)^+ v(r,\vec{\omega})dV \qquad (2)$$

where S is the points of the neighborhood sphere. Both formulations are integrals in three-dimensions, where random samples provide more accurate estimates than classical quadrature rules. The basic idea of Monte Carlo integration is to generate samples randomly in the integration domain with sampling density $p(z)$, and trace back an integral of $F(z)$ to the average of ratios $F(z_i)/p(z_i)$ obtained from n samples:

$$\int_S F(z)dz = \int_S \frac{F(z)}{p(z)}p(z)dz = E\left[\frac{F(z)}{p(z)}\right] = \frac{1}{n}\sum_{i=1}^{n}\frac{F(z_i)}{p(z_i)}\,.$$

This estimator is accurate if p mimics F, i.e. the variation of F/p is a low as possible.

4.1. Uniform sampling of the neighborhood sphere

If sample points are uniformly distributed in the neighborhood sphere, thus their density is $p(r,\vec{\omega}) = 3/(4R^3\pi)$, then the Monte Carlo estimate of the ambient occlusion integral is:

$$O(\vec{s}) = \frac{1}{\pi}\int_S \frac{d\mu(r)}{dr}\frac{1}{r^2}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)^+ v(r,\vec{\omega})dV \approx \frac{4R^3}{3N}\sum_{i=1}^{n}\frac{d\mu(r_i)}{dr}\frac{1}{r_i^2}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}_i\right)^+ v(r_i,\vec{\omega}_i)$$

where n is the number of samples, $v(r_i,\vec{\omega}_i)$ is the visibility function that is 1 if the sample point being at distance r_i and in direction $\vec{\omega}_i$ is not occluded and zero otherwise.

4.2. Uniform sampling of the visible half of the neighborhood sphere

As the scalar product of the surface normal and the direction is replaced by zero if the sample point is behind the shaded point, the contribution of this part is zero. Thus, it is better to place sample points only in the visible hemisphere with density $p(r,\vec{\omega}) = 3/(2R^3\pi)$ and use our second estimate:

$$O(\vec{s}) = \frac{1}{\pi}\int_S \frac{d\mu(r)}{dr}\frac{1}{r^2}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)^+ v(r,\vec{\omega})dV \approx \frac{2R^3}{3N}\sum_{i=1}^{n}\frac{d\mu(r_i)}{dr}\frac{1}{r_i^2}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}_i\right)v(r_i,\vec{\omega}_i).$$

Evaluating volumes by point samples has relatively high variance. More accurate estimates can be expected if a sample point identifies a larger portion of the volume to be evaluated.

4.3. Sample generation with importance sampling

According to the theory of Monte Carlo integration, the accuracy of the estimate gets higher if the sample point density mimics the integrand as much as possible. Inspecting equation 1, we need a sampling strategy where the direction is sampled with cosine distribution, and the distance with the derivative of the fuzzy membership function.

Let us now consider a unit radius sphere centered at the considered point. The sphere intersects the tangent plane *xy* in a unit circle. We find uniformly distributed points in the circle taking uniform independent random numbers (ξ_x,ξ_y,ξ_z), then transforming the first two coordinates to [-1,1] as $x = 2\xi_x - 1$, $y = 2\xi_y - 1$, and finally checking whether the point of these coordinates are inside the unit radius circle by examining $x^2 + y^2 \le 1$. Samples being outside are rejected. The accepted points are projected up to the sphere surface generating directions with cosine distribution:

$$\vec{\omega} = x\vec{T} + y\vec{B} + \sqrt{1-x^2-y^2}\,\vec{N}$$

where $\vec{T}$, $\vec{B}$, and $\vec{N}$ are the tangent, binormal, and normal vectors, respectively. Having found a cosine distributed point on the unit sphere, distance r is obtained with density $d\mu(r)/dr$ as $r = \mu^{-1}(\xi_z)$.

With these samples, the visibility is calculated, and the integral will be estimated by the integrand values divided by the density of the sample generation:

$$O(\vec{s}) = \frac{1}{\pi}\int_{\Omega}\int_{0}^{D}\frac{d\mu(r)}{dr}\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)^{+} v(r,\vec{\omega})drd\omega \approx \frac{1}{n}\sum_{i=1}^{n} v(r_i,\vec{\omega}_i)$$

4.4. Uniform sampling inside the tangent sphere

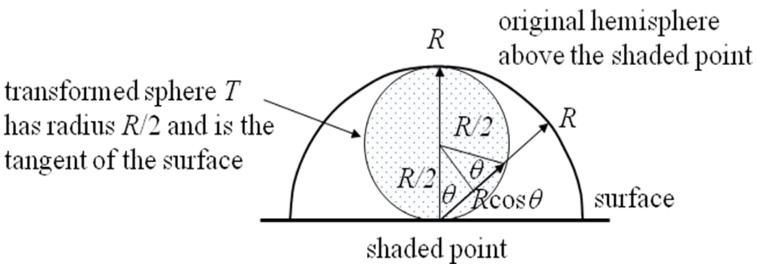


Figure 5. Transforming a hemisphere of radius R by shrinking distances by the cosine of the surface normal and the direction of interest results in another sphere of radius R/2.

Inspecting the ambient occlusion formula, we can observe that it is a double integral inside a hemisphere where the integrand includes factor $\left(\vec{N}_{\vec{s}}\cdot\vec{\omega}\right)$ that is equal to the cosine of the angle between the surface normal and the direction of the sample, thus directions enclosing a larger angle with the surface normal are less important. Instead of the multiplication, the effect of this cosine factor can also be mimicked by reducing the size of the integration domain proportionally to the cosine value. Note that this is equivalent to the original integral only if the remaining factors of the integrand are constant, and can be accepted as an approximation in other cases. If we take uniform samples inside tangent sphere T of volume $R^3\pi/6$, the Monte Carlo quadrature is:

$$O(\vec{s}) \approx \frac{1}{\pi}\int_{T}\frac{d\mu(r)}{dr}\frac{1}{r^2}dV \approx \frac{R^3}{6N}\sum_{i=1}^{n}\frac{d\mu(r_i)}{dr}\frac{1}{r_i^2}v(r_i,\vec{\omega}_i).$$

4.5. Sampling the base disk of the sphere

Using the notation of Figure 3, we can place samples on the disk perpendicular to the viewing direction and decompose the visible volume to pipes centered at the sample point and having axis parallel to the viewing direction.

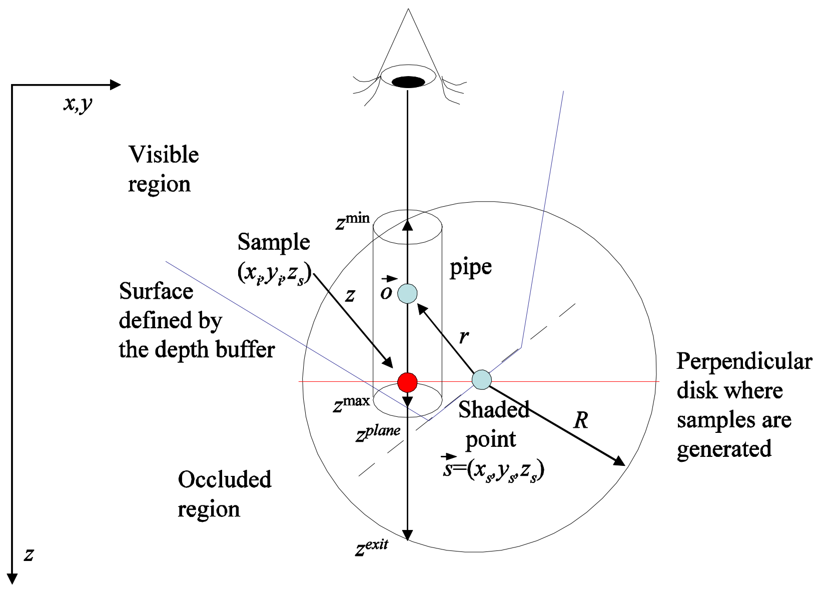


Figure 6. Evaluation of the volumetric integral in the visible part of the neighborhood sphere.

Let us compute the volumetric integral with differential elements having dz height and $dA=dxdy$ base area at (x, y) of the disk C of radius R and perpendicular to axis z (Figure 3):

$$O(\vec{s}) = \frac{1}{\pi} \int_{x,y\in C} \int_{z_{\min}}^{z_{\max}} \frac{d\mu(r)}{dr} \frac{1}{r^2} \left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right) dz dx dy = \frac{1}{\pi} \int_{x,y\in C} h(x,y,z_{\min},z_{\max}) dx dy\,,$$

where h is the integral over z:

$$h(x,y,z_{\min},z_{\max}) = \int_{z_{\min}}^{z_{\max}} \frac{d\mu(r)}{dr} \frac{1}{r^2} \left(\vec{N}_{\vec{s}} \cdot \vec{\omega}\right) dz = \int_{z_{\min}}^{z_{\max}} \frac{d\mu(r)}{dr} \frac{1}{r^2} \frac{\left((\vec{o}-\vec{s}) \cdot \vec{N}_{\vec{s}}\right)}{r} dz\,.$$

Note that the integrand is a polynomial of coordinate z if $d\mu(r)/dr \cdot 1/r$ is a polynomial of z, that is when $d\mu(r)/dr \cdot 1/r$ is a polynomial of r^2. In these cases integral h can be analytically evaluated. We have found $\mu(r) = (r/R)^4$ a particularly simple and effective choice.

The integral over the disk is evaluated with a numerical quadrature. We sample n uniformly distributed points (x_i, y_i) in the disk of radius R. Thus, a sample corresponds to disk area $\Delta A=R^2\pi/n$.

The integrand is function h whose evaluation requires minimum and maximum z values where points are visible from the camera. We initialize this interval to the range where the ray is inside the sphere, and then restrict the bounds to those points which are in front of the surface, i.e. where the z coordinate is less than the depth value stored in the depth buffer, and which are in front of the tangent plane of the shaded point. The contribution of the pipes to the volumetric integral of the ambient occlusion is

$$O(\vec{s}) = \frac{1}{\pi} \int_{x,y\in C} h(x,y,z_{\min},z_{\max}) dx dy \approx \frac{R^2}{n} \sum_{i=1}^{n} h_i\,.$$

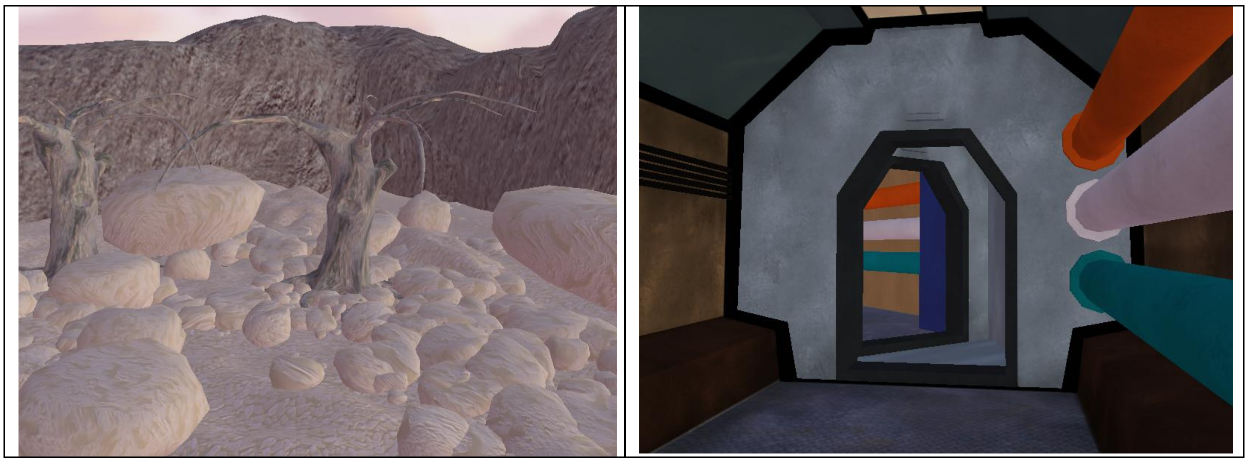
5. *Interleaved sampling*

All presented quadrature formulae are approximations and their error decreases if new sample points are added. However, computing the formula with many sample points reduces rendering speed. Thus, we consider another technique that reduces the computation error without performance degradation.

Interleaved sampling [Keller01] takes advantage of the estimates in neighboring pixels. The method discussed so far has some error in each pixel, depending on the particular samples used. If we took different random numbers in neighboring pixels, dot noise would be present. Using the same random numbers in every pixel would make the error correlated and replace dot noise with stripes. Unfortunately, both stripes and pixel noise are quite disturbing. Interleaved sampling uses different sets of samples in the pixels of a 4 × 4 pixel pattern. The errors in the pixels of a 4 × 4 pixel pattern are uncorrelated, which can be successfully reduced by a low-pass filter of the same size. When implementing the low-pass filter, we also check whether the depth difference between the current and neighboring pixels exceeds a given limit. If it does, we do not include the neighboring pixels in the averaging operation.

6. *Results*

The proposed methods have been implemented in DirectX/HLSL environment and their performance has been measured on an NVIDIA GeForce 8800 GTX GPU at 800×600 resolution. The rendering results of the harbor scene are shown by Figure 3. Taking 16 samples per pixel, the ambient occlusion and indirect lighting computation runs over 100 FPS.



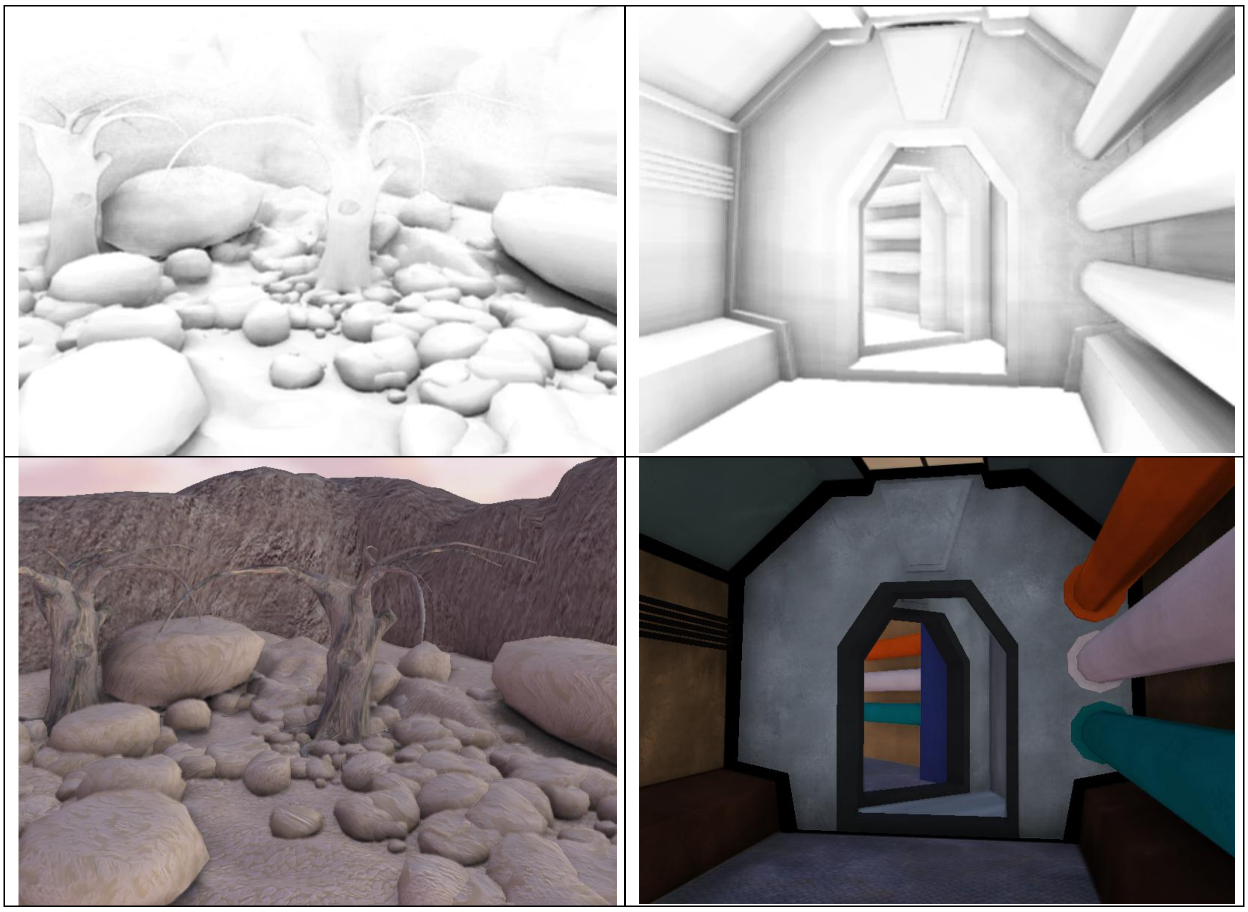


Figure 7. The Rocky Scene and the Space Station Scene rendered with classical ambient reflection model (top row), ambient occlusion only (middle row), and using the ambient occlusion and the indirect lighting. We used the method called "sample generation with importance sampling".

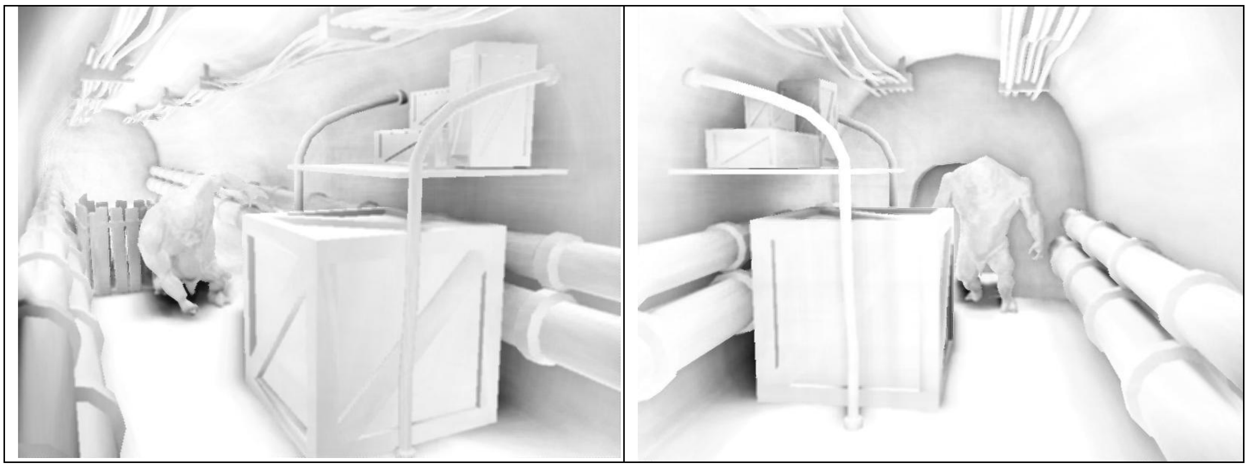
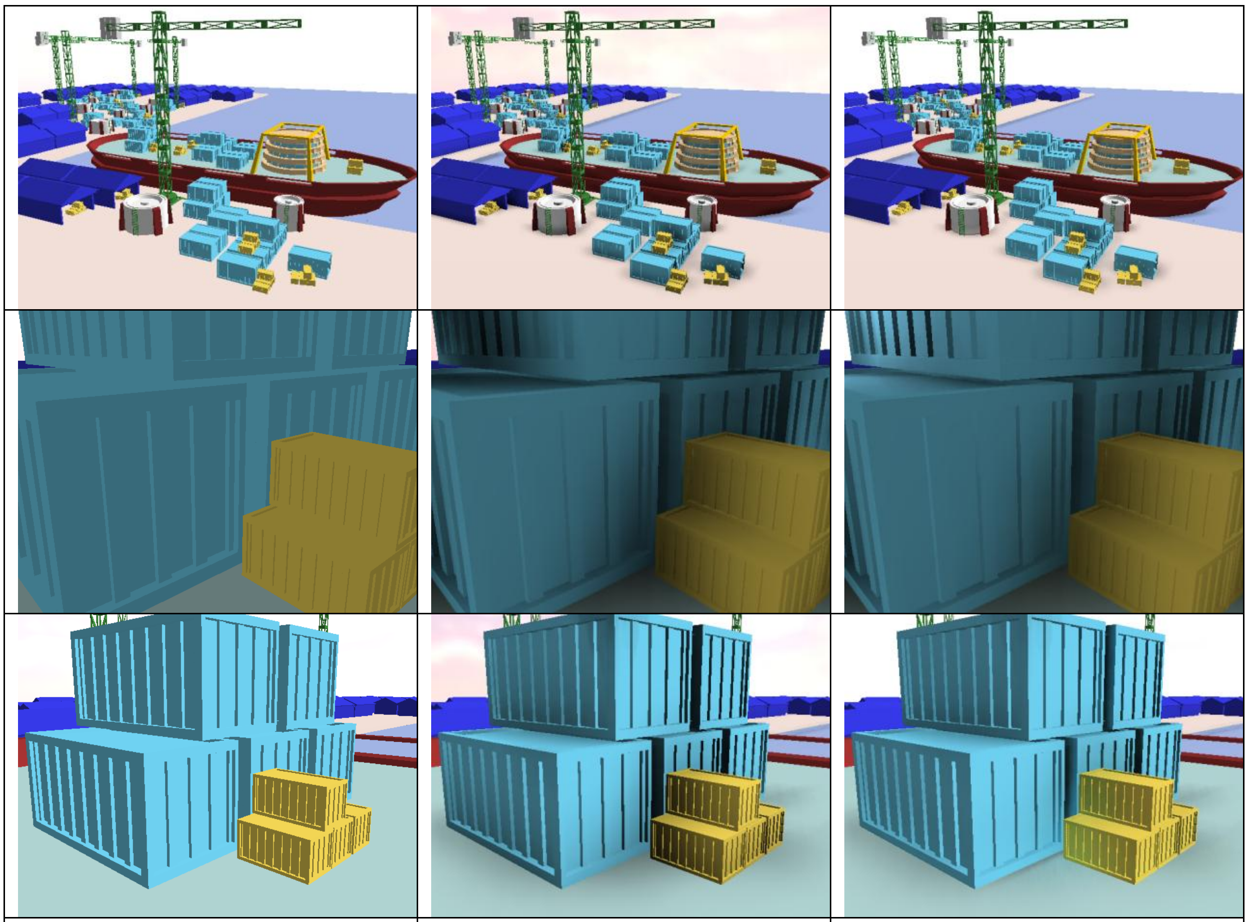


Figure 8. The Ogre in the tunnel scene rendered ambient occlusion generated with "uniform samples in the tangent sphere".



Direct lighting	Direct lighting + ambient occlusion	Direct lighting + ambient occlusion + indirect lighting

Figure 9. Rendering results of the harbor scene. We used the algorithm "sampling the base disk of the sphere".

7. Conclusions

This article proposed fast methods for the computation of ambient occlusion and indirect lighting. The new approach is based on rewriting the ambient occlusion integral to evaluate the volume of the open part of the neighborhood hemisphere. The new integral can be evaluated more accurately with the same number of samples, thus, it is more appropriate in real-time systems where low noise results are needed with just a few cheap samples. The method does not require pre-processing and runs at high frame rates, since it only needs as few as 12-16 samples per pixel.

Acknowledgements

This work has been supported by OTKA K-719922, and by the scientific program of the "Development of quality-oriented and harmonized R+D+I strategy and functional model at BME" (TÁMOP-4.2.1/B-09/1/KMR-2010-0002), by TIN 2007-68066-C04-01 from the Spanish Government.

References

[Landis02] H. Landis. Production-ready global illumination. SIGGRAPH Course notes 16, 2002. http:// www.debevec.org/HDRI2004/landis-S2002-course16-prodreadyGI.pdf

[Iones03] A. Iones, A. Krupkin, M. Sbert, and S. Zhukov. Fast realistic lighting for video games. IEEE Computer Graphics and Applications, 23(3):54–64, 2003.

[Mendez03] A. Méndez, M. Sbert, and J. Catá. Real-time obscurances with color bleeding.
In SCCG '03: Proceedings of the 19th spring conference on Computer graphics, pages 171–176, New York, NY, USA, 2003. ACM.

[Mittring07] M. Mittring. Finding next gen - CryEngine 2. In Advanced Real-Time Rendering in 3D Graphics and Games Course - Siggraph 2007, pages 97-121. 2007.

[Pharr04] M. Pharr and S. Green. Ambient occlusion. In GPU Gems, pages 279–292. Addison-Wesley, 2004.

[Shanmugam07] P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In Proceedings of the 2007 Symposium on Interactive 3D graphics, pages 73–80, 2007.

[Szirmay10] L. Szirmay-Kalos, T. Umenhoffer, B. Tóth, L. Szécsi, M. Sbert. Volumetric Ambient Occlusion for Real-Time Rendering and Games. In IEEE Computer Graphics and Applications, Vol. 30, No. 1, pp. 70-79, 2010.

[Zhukov98] S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model.
In Proceedings of the Eurographics Rendering Workshop, pages 45–56, 1998.