Volumetric Ambient Occlusion for Real-Time Rendering and Games

László Szirmay-Kalos, Tamás Umenhoffer, Balázs Tóth, and László Szécsi ■ *Budapest University of Technology and Economics*

Mateu Sbert ■ *University of Girona*

This new GPU-based algorithm can compute ambient occlusion to inexpensively approximate global-illumination effects in real-time systems and games. The algorithm is based on a novel interpretation of ambient occlusion that measures the relative volume of the visible part of the surface's tangent sphere.

Realistically computing the illumination of ambient lighting from the sky or large-area light sources is essential in high-quality rendering because most of the illumination in real scenes comes from such sources. Because the physically plausible rendering equation is too complex to solve, real-time system and game developers usually prefer alternative approaches. Ambient occlusion has become particularly popular because it promises a good trade-off between the computational cost and the resulting image quality.

The classical ambient-occlusion model considers only near occlusions and, for each shaded point, evaluates a directional integral to determine how strongly the distant ambient lighting can take effect. Because this model isn't physically based, it can't predict the difference between real and synthetic images. On the other hand, the classical definition and evaluation of ambient occlusion is based on ray tracing, so its GPU implementation is neither simple nor optimal.

To address these problems, we establish links between the physically based rendering equation and the ambient-occlusion model, and we propose an ambient-occlusion algorithm to make the method more appropriate for GPU evaluation. This algorithm replaces the directional integral with a volumetric one, which can be more efficiently and accurately evaluated on the GPU. The resulting method

- runs at high frame rates on current GPUs,
- requires no preprocessing and so applies to general dynamic models, and
- can provide smooth shading with only a few samples, thanks to partial analytic integration and interleaved sampling.

These properties make the algorithm suitable for real-time rendering and games.

Deriving Ambient Occlusion from the Rendering Equation

Our new algorithm provides a way to quickly compute the reflection of ambient light. (If the scene contains other sources—for example, directional or point lights—we could add their effects to the ambient light's illumination as well.) We assume that illumination in the scene comes primarily from a homogeneous skylight radiance source, which we denote as ambient radiance L^a. For simplicity, we consider only diffuse surfaces. According to the rendering equation, we can obtain reflected radiance L^r in shaded point $\vec{x}$ as

$$L^r(\vec{x}) = \frac{a(\vec{x})}{\pi} \int_{\Omega} L^{in}(\vec{x}, \vec{\omega}) \cos^+(\theta) d\omega, \quad (1)$$

where Ω is the set of directions in the hemisphere above $\vec{x}$, $L^{in}(\vec{x}, \vec{\omega})$ is the incident radiance from direction $\vec{\omega}$, $a(\vec{x})$ is the surface's albedo, and θ is the angle between the surface normal and the illumination direction $\vec{\omega}$. If incident angle θ is greater than 90 degrees, the object casts a shadow on itself, so we should replace the cosine value with 0, which the superscript in $\cos^+$ conveys.

If no surface appears from $\vec{x}$ in direction $\vec{\omega}$,

 Published by the IEEE Computer Society

shaded point $\vec{x}$ is *open* in this direction, and incident radiance L^{in} is equal to ambient radiance L^a. If an occluder is nearby, the point is *closed* in this direction, and the incident radiance is that of the occluder surface. The exact determination of this radiance would require a global-illumination solution, which is too costly in real applications. So, we assume the radiance is proportional to the ambient radiance and to a factor expressing the point's *openness* (or *accessibility*).

The theory of classical ambient occlusion considers an occluder to be nearby if its distance is smaller than a predefined threshold R. However, to handle a subjective property such as closeness, a *fuzzy measure*, $\mu\left(D\left(\vec{\omega}\right)\right)$, is more suitable. This measure defines how strongly direction $\vec{\omega}$ belongs to the set of open directions, on the basis of the occlusion's distance D in direction $\vec{\omega}$. In other words, this fuzzy measure expresses how an occluder at distance D lets the ambient lighting take effect. Relying on the physics analogy of the nonscattering participating medium, a possible fuzzy measure could be $\mu(D) = 1 - \exp(-\tau D)$, where τ is the medium's *absorption coefficient*.[1,2] Unfortunately, this interpretation requires the distance of far occlusions as well, whereas classical ambient occlusion need not compute occlusions farther than R, localizing and thus simplifying the shading process.

So, for practical fuzzy measures, we use nonnegative functions that monotonously increase from 0 until they reach 1 at distance R. The application developer can set the particular value of R. When we increase this value, shadows due to ambient occlusion get larger and softer.

Using the fuzzy measure of openness, the incident radiance is

$$L^{in}\left(\vec{x},\vec{\omega}\right) = L^a \mu\left(D\left(\vec{\omega}\right)\right).$$

So, we can write the reflected radiance (Equation 1) in this form:

$$L^r\left(\vec{x}\right) = a\left(\vec{x}\right) \cdot L^a \cdot O\left(\vec{x}\right),$$

where

$$O\left(\vec{x}\right) = \frac{1}{\pi} \int_{\Omega} \mu\left(D\left(\vec{\omega}\right)\right) \cos^{+}(\theta)\, d\omega \qquad (2)$$

is the *ambient occlusion* representing the local geometry and expresses how strongly the ambient lighting can take effect at point $\vec{x}$. (The "Other Ambient-Lighting Models" sidebar discusses other approaches to modeling ambient lighting.)

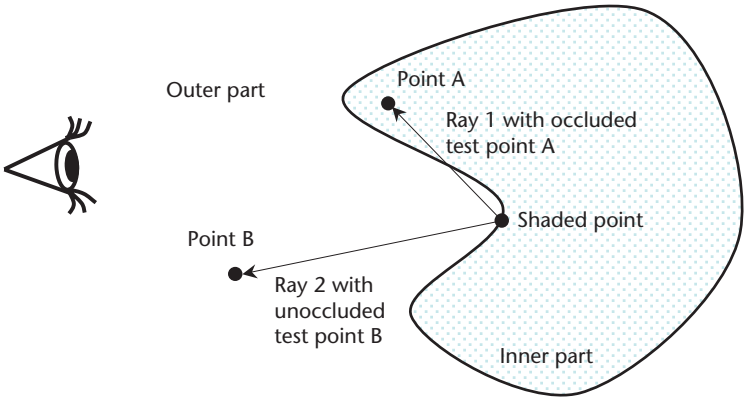


Figure 1. Replacing ray tracing with containment tests. If test point B along a ray starting at the shaded point in the outer region or at the region boundary is also in the outer region, then the ray either has not intersected the surface or has intersected it at least twice. If the ray is short enough to intersect the surface at most once, the condition of being in the outer region is equivalent to the condition of no intersection occurring.

Our GPU-Based Algorithm for Ambient Occlusion

Evaluating the directional integral in the ambient-occlusion formula (Equation 2) requires tracing rays in many directions, which is costly and requires complex GPU shaders. So, we transform this directional integral to a volumetric one that we can efficiently evaluate on the GPU. The integral transformation involves two steps.

First, considering the fuzzy membership function's derivative instead of the fuzzy membership function, we replace the expensive ray-tracing operation with a simple containment test. This replacement is valid if neighborhood R is small enough for us to assume a ray intersects the surface at most once in interval [0, R].

Second, we compensate for factor $\cos^{+}(\theta)$ in Equation 2 by transforming the integration domain mapping the hemisphere above the surface to the tangent sphere. This makes ambient occlusion depend on the volume of that portion of the tangent sphere belonging to the free space unoccupied by objects.

Replacing Ray Tracing with Containment Tests

Assume the surfaces subdivide the space into an *outer part*, where the camera is, and *inner parts*, which can't be reached from the camera without crossing a surface. The *characteristic function* $I\left(\vec{p}\right)$ is 1 if point $\vec{p}$ is an outer point, and 0 otherwise. The boundary between the inner and outer parts (that is, the surfaces) belongs to the outer part by definition (see Figure 1).

The form of indicator function $I\left(\vec{p}\right)$ depends on the surface type or representation (see Figure 2):

Other Ambient-Lighting Models

The oldest ambient-lighting model assumes that incident radiance L^{in} is equal to ambient radiance L^a at all points and directions. A point reflects $k_a L^a$ intensity, where k_a is a surface's ambient reflectivity. Because this model ignores the scene's geometry, the resulting images are plain and don't have a 3D appearance. A physically correct approach would be a rendering-equation solution that takes into account all factors missing in the classic ambient-lighting model. However, this approach is too computationally expensive for rendering dynamic scenes in real time.

Rather than working with the rendering equation, local approaches examine only a neighborhood of the shaded point. Ambient occlusion[1–3] and obscurance[4,5] methods compute only how *open* the scene is in a point's neighborhood (openness is measured by the solid angle in which no nearby occluder is seen), and they scale the ambient light accordingly. Originally, the neighborhood had a sharp boundary in ambient occlusion and a fuzzy boundary in the obscurance method, but today these terms refer to similar techniques. Because the term "ambient occlusion" has become more popular, we also use this term in this article.

If, in addition to computing the points' openness, we obtain the average of open directions, this extra directional information lets ambient occlusion modulate not only ambient light but also environment maps.[2] Spectral extensions also take into account the neighborhood's and the entire scene's average spectral reflectivity, letting us inexpensively simulate even color-bleeding effects.[6,7]

Because ambient occlusion is the sky's "local invisibility," real-time methods rely on scene representations that easily determine the visibility. These include the approximation of surfaces by disks[7,8] or spheres.[9] The method proposed by Petrik Clarberg and Tomas Akenine-Möller directly approximates the visibility function.[10] A cube map or a depth map rendered from the camera can also serve as a sampled representation of the scene.[11,12] Because these maps are already in the GPU's texture memory, a fragment shader program can check the visibility for many directions. *Screen space ambient occlusion* computes the difference between the depth values.[11] *Horizon-split ambient occlusion* generates and evaluates a horizon map on the fly.[12]

References

1. H. Landis "Production-Ready Global Illumination," *Siggraph 2002 Course Notes*, Course 16: Renderman in Production, ACM Press, 2002, pp. 87–102; www.debevec.org/HDRI2004/landis-S2002-course16-prodreadyGI.pdf.
2. M. Pharr and S. Green, "Ambient Occlusion," *GPU Gems*, R. Fernando, ed., Addison-Wesley, 2004, pp. 279–292.
3. J. Kontkanen and T. Aila, "Ambient Occlusion for Animated Characters," *Proc. 17th Eurographics Symp. Rendering*, Eurographics Assoc., 2006, pp. 343–348.
4. S. Zhukov, A. Iones, and G. Kronin, "An Ambient Light Illumination Model," *Proc. Eurographics Workshop Rendering Techniques*, Springer, 1998, pp. 45–56.
5. A. Iones et al., "Fast Realistic Lighting for Video Games," *IEEE Computer Graphics and Applications*, vol. 23, no. 3, 2003, pp. 54–64.
6. A. Méndez, M. Sbert, and J. Catá, "Real-Time Obscurances with Color Bleeding," *Proc. 19th Spring Conf. Computer Graphics* (SCCG 03), ACM Press, 2003, pp. 171–176.
7. M. Bunnell, "Dynamic Ambient Occlusion and Indirect Lighting," *GPU Gems 2*, M. Pharr, ed., Addison-Wesley, 2005, pp. 223–233.
8. J. Hoberock and Y. Jia, "High-Quality Ambient Occlusion," *GPU Gems 3*, H. Nguyen, ed., Addison-Wesley, 2007, pp. 257–274.
9. P. Shanmugam and O. Arikan, "Hardware Accelerated Ambient Occlusion Techniques on GPUs," *Proc. Symp. Interactive 3D Graphics and Games*, ACM Press, 2007, pp. 73–80.
10. P. Clarberg and T. Akenine-Möller, "Exploiting Visibility Correlation in Direct Illumination," *Computer Graphics Forum*, vol. 27, no. 4, 2008, pp. 1125–1136.
11. M. Mittring, "Finding Next Gen: CryEngine 2," *Siggraph 2007 Course Notes*, Course 28: Advanced Real-Time Rendering in 3D Graphics and Games, ACM Press, 2007, pp. 97–121.
12. M. Sainz, "Real-Time Depth Buffer Based Ambient Occlusion," *Proc. Games Developers Conf.* (GDC 08), GDC, 2008; http://developer.download.nvidia.com/presentations/2008/GDC/GDC08_Ambient_Occlusion.pdf.

- *Implicit surfaces* are defined by $f(\vec{p}) = 0$. In this case, $I(\vec{p})$ is 1 if $f(\vec{p})$ is positive or 0 (the point is outside the object or on its surface), and is 0 otherwise.
- *Height fields* represent an important special case of implicit surfaces. Points (x, y, z) satisfying equation $z = h(x, y)$ form a height field. Because the eye is usually above the height field, the characteristic function should indicate the case when, for point $\vec{p} = (p_x, p_y, p_z)$, relation $p_z \geq h(p_x, p_y)$ holds. We can also consider a *displacement-mapped* mesh's polygon as a height field in the macrostructure polygon's tangent space.
- The *z-buffer* can provide an inner/outer distinction, as in screen space ambient-occlusion methods.[3,4] If the depth map reports a point as occluded, then surfaces separate this point from the eye, and its characteristic value is 0. If the point passes the depth test, it's in the same region as the eye, so it gets the indicator value 1.

In all three cases, we classify a point as inner or outer and determine the distance between point (p_x, p_y, p_z) and the surface along the z-axis. When

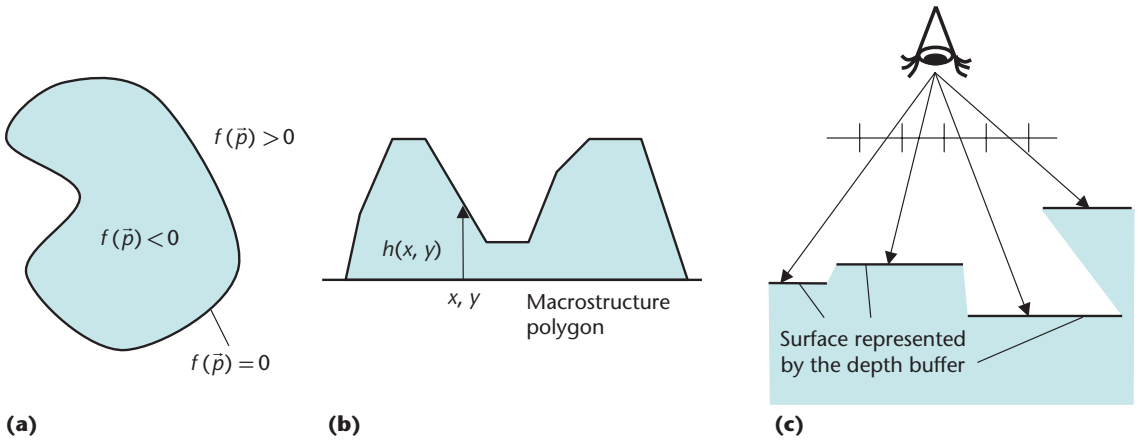


Figure 2. Examples of three different ways to define the characteristic function: (a) implicit surface, (b) height field, and (c) *z*-buffer. Outer regions, where $I = 1$, are blank; inner regions, where $I = 0$, are filled.

the *z*-buffer's content defines the separation, *z* is the viewing direction in the clipping space. Reading the depth value z^* with the point's (p_x, p_y) coordinates, we can express the distance between the point and the surface as $z^* - p_z$. If height field $h(x, y)$ defines the characteristic function, the distance along the *z*-axis is $p_z - h(p_x, p_y)$. Finally, when implicit equation $f(x, y, z) = 0$ defines the surface, we can use Taylor's approximation to estimate the distance between point (p_x, p_y, p_z) and point (p_x, p_y, z^*) on the surface:

$$f\left(p_x, p_y, p_z + \left(z^* - p_z\right)\right) = 0 \Rightarrow z^* - p_z \approx -\frac{f\left(p_x, p_y, p_z\right)}{\partial f / \partial z}.$$

To evaluate the ambient occlusion using containment tests, we express it as a 3D integral. For a point at distance *D*, we find fuzzy measure $\mu(D)$ by integrating its derivative from 0 to *D*, because $\mu(0) = 0$. Then, we can extend the integration domain from *D* to *R* by multiplying the integrand by a step function that replaces the derivative with 0 when the distance is greater than *D*:

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

where $\varepsilon(x)$ is the step function; $\varepsilon(x)$ is 1 if $x \geq 0$, and is 0 otherwise. This formulation requires generalized derivatives for classical ambient occlusion because its membership function is a step function, and the step function's derivative is a Dirac delta function (see Figure 3).

Substituting this integral into the ambient-occlusion formula gives

$$O(\vec{x}) = \frac{1}{\pi} \int_\Omega \int_0^R \frac{d\mu(r)}{dr} \varepsilon(D - r) \cos^+(\theta) dr d\omega. \qquad (3)$$

Now we consider a ray of $\vec{x} + \vec{\omega} r$, where shaded point $\vec{x}$ is the origin, $\vec{\omega}$ is the direction, and distance *r* is the ray parameter. Indicator $\varepsilon(D - r)$ is 1 if the intersection's distance *D* is larger than the current distance *r*, and is 0 otherwise; hence, this indicator shows whether an intersection has occurred. If we assume the ray intersects the surface

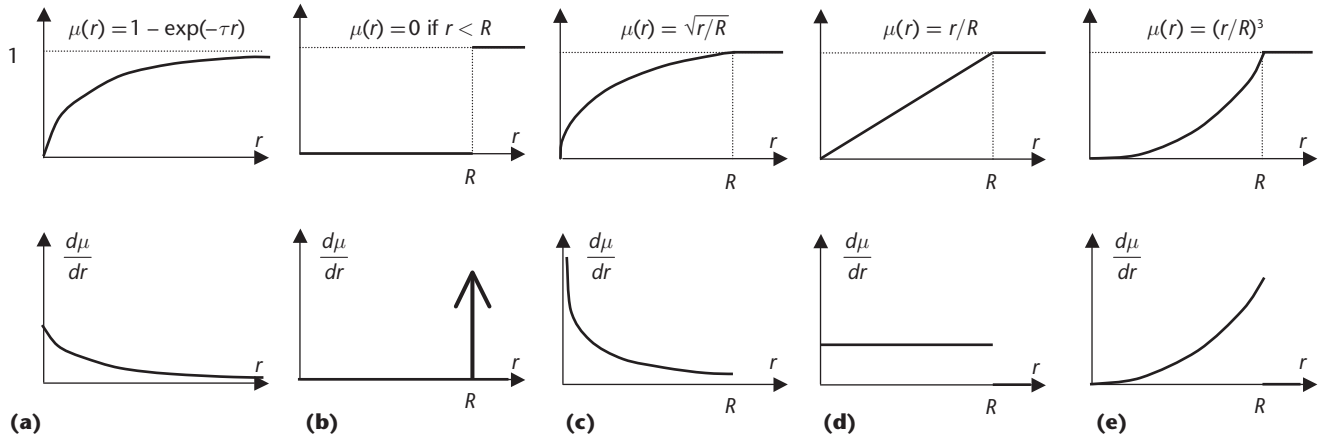


Figure 3. Example fuzzy membership functions and their derivatives: (a) exponential, (b) classical ambient occlusion, (c) localized square root, (d) linear, and (e) volumetric ambient occlusion. The exponential function can be given a physical interpretation, but this function requires global-visibility computations,[1,2] whereas the other functions need visibility checks only in a neighborhood of radius *R*. Classical ambient occlusion uses a nonfuzzy separation.[5] The localized square root is a good compromise between the global exponential and the nonfuzzy separation.[6]

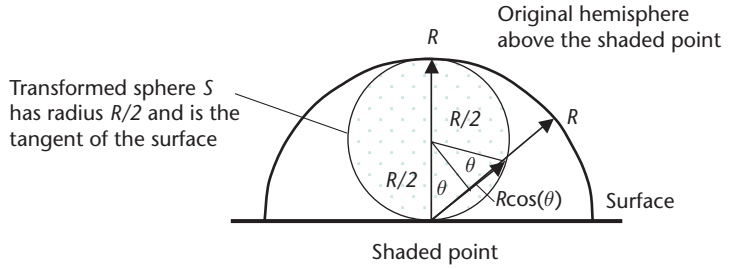


Figure 4. Transforming a hemisphere of radius *R* via shrinking distances by cos(θ) results in another sphere of radius *R/2*. The surface is the new sphere's tangent at shaded point $\vec{x}$.

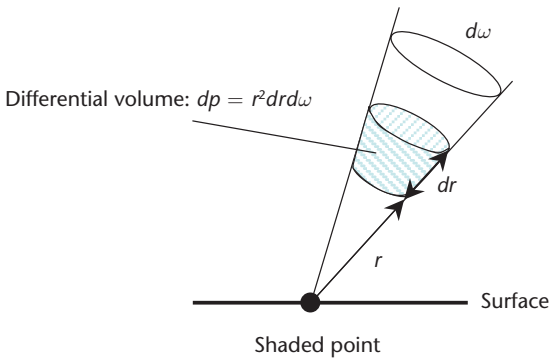


Figure 5. The differential volume swept when *r* changes by *dr* and direction $\vec{\omega}$ varies by *dω*. This relationship lets us replace the integral of directions and distance with a volumetric quadrature.

at most once in neighborhood *R*, then the condition that $\vec{x}+\vec{\omega}r$ is in the outer part ($I(\vec{x}+\vec{\omega}r)=1$) also shows that no intersection has occurred yet. So, as long as only one intersection is possible in *R*, $\varepsilon(D-r)$ and $I(\vec{x}+\vec{\omega}r)$ are equivalent. Replacing step function $\varepsilon(D-r)$ by indicator function *I* in Equation 3, we obtain

$$O(\vec{x})=\frac{1}{\pi}\int_{\Omega}\int_{0}^{R}\frac{d\mu(r)}{dr}I(\vec{x}+\vec{\omega}r)\cos^{+}(\theta)drd\omega\,. \qquad (4)$$

One possibility for estimating this integral is the Monte Carlo quadrature. According to *importance sampling*, we could use cosine-distributed sample directions $\vec{\omega}_i$ and distance samples r_i following density $d\mu(r)/dr$, obtained by transforming uniformly distributed samples ξ_i as $r_i = \mu^{-1}(\xi_i)$. In our case, the number of samples *n* is small, and we should generate the random samples only once, so we can hard-wire samples $(X_i, Y_i, Z_i)=\vec{\omega}_i n_i$ into the shader code computation:

$$O(\vec{x})\approx\frac{1}{n}\sum_{i=1}^{n}I\left(\vec{x}+X_i\vec{T}+Y_i\vec{B}+Z_i\vec{N}\right), \qquad (5)$$

where $\vec{T}$, $\vec{B}$, and $\vec{N}$ are the tangent, binormal, and normal vectors. We call this method the *containment-test-based* algorithm. It approximates the quadrature by averaging *n* binary numbers, such that the average would have a binomial distribution with a mean of $O(\vec{x})$ and a standard deviation of $\sqrt{O(\vec{x})(1-O(\vec{x}))/n}$. Ambient occlusion $O(\vec{x})$ is in [0, 1], and the standard deviation reaches its maximum $1/\sqrt{4n}$ when the ambient occlusion value is 1/2. Reducing the maximum standard deviation (that is, the error) to below 0.01 in a pixel requires 2,500 random samples—far too many for real applications.

So, we need a better estimate for the ambient-occlusion integral that requires fewer samples to achieve the same accuracy. We combine three techniques to reach this goal. First, and most important, we reformulate the integral to reduce the integrand's variation and to incorporate all available information into the quadrature. In particular, we use the distance to the separating surface as this additional information. Second, we replace statistically independent random samples with samples following the Poisson-disk distribution. Finally, we use interleaved sampling, exploiting the samples obtained from the neighboring pixels to improve the estimate in the current pixel.

Exploiting the Distance to the Separating Surface

As Equation 4 shows, the ambient occlusion is a double integral inside a hemisphere, where the integrand includes factor $\cos^{+}(\theta)$. So, directions enclosing a larger angle with the surface normal are less important. Rather than using multiplication, we can mimic the cosine factor's effect by reducing the integration domain's size proportionally to $\cos^{+}(\theta)$:

$$O(\vec{x})\approx\frac{1}{\pi}\int_{\Omega}\int_{0}^{R\cos^{+}(\theta)}\frac{d\mu(r)}{dr}I(\vec{x}+\vec{\omega})drd\omega\,.$$

This is equivalent to the original integral only if the integrand's remaining factors are constant; it can be accepted as an approximation in other cases.

Transforming a hemisphere by shrinking distances in directions enclosing angle θ with the surface normal by cos(θ) results in another sphere, which we denote as *S* (see Figure 4). This new sphere has radius *R*/2, and its center is at distance *R*/2 from shaded point $\vec{x}$ in the surface normal's direction. Whereas the shaded point was the center of the original hemisphere's base circle, the surface in this case is the new sphere's tangent at shaded point $\vec{x}$.

To replace integrals over directions and distances with a volumetric integral, we examine the volume swept when distance *r* changes by *dr* and when direction $\vec{\omega}$ varies by solid angle *dω* (see Figure 5). To do this, we sweep a differential volume $dp = r^2drd\omega$. Thus, we can express the ambient occlusion as a volumetric integral in *S* instead of a double integral of directions and distances:

$$O(\vec{x}) \approx \frac{1}{\pi} \int_{\vec{p} \in S} \frac{d\mu(r(\vec{p}))}{dr} \frac{1}{(r(\vec{p}))^2} I(\vec{p})\, dp,$$

where $r(\vec{p})$ is the distance of $\vec{p}$ from the shaded point.

If we set the fuzzy membership function such that $d\mu(r)/dr$ is proportional to r^2 (Figure 3e), the ambient-occlusion integral simplifies to the membership function's volumetric integral. This observation leads us to a new definition of a point's openness, which we call the *volumetric ambient occlusion*. We denote it as $V(\vec{x})$ to distinguish it from ambient occlusion $O(\vec{x})$. The volumetric ambient occlusion is the relative volume of the unoccluded part of tangent sphere S. Formally, we define this function as

$$V(\vec{x}) = \frac{\int_S I(\vec{p})\, dp}{|S|},$$

where $|S| = 4(R/2)^3\pi/3$ is the tangent sphere's volume and ensures that the volumetric ambient occlusion is in the range [0, 1]. Figure 6 compares the ambient-occlusion computation to ray-tracing-based methods, the containment-test-based algorithm, and the volumetric-ambient-conclusion computation.

When evaluating volumetric integral $\int_S I(\vec{p})dp$, we can exploit the distances to the separating surface. We compute the volume as a sum of pipes of equal cross-section areas, with each pipe's axis parallel to the z-axis (see Figure 7). The pipes are limited by either the separating surface or the tangent sphere's surface.

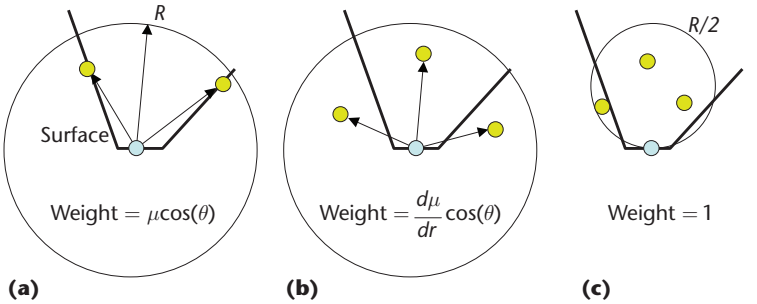


Figure 6. A comparison of (a) ray-tracing-based, (b) containment-test-based, and (c) volumetric ambient occlusion. These methods differ according to sample generation as well as weighting.

Denoting the tangent sphere's center as (x_c, y_c, z_c), we consider a disk of radius $R/2$ around this point, perpendicular to the z-axis. We sample n uniformly distributed points (x_i, y_i) in a unit-radius disk and transform them onto the disk with radius $R/2$—that is, perpendicular to z. A transformed point has coordinates $(x_iR/2, y_iR/2, z_c)$; we call it a *sample* in Figure 7. A line crossing the ith sample point and parallel to the z-axis enters the sphere at z_i^{entry}, exits it at z_i^{exit}, and intersects the surface at z_i^*, with $z_i^{entry} = z_c - \frac{R}{2}\sqrt{1 - x_i^2 - y_i^2}$ and $z_i^{exit} = z_c + \frac{R}{2}\sqrt{1 - x_i^2 - y_i^2}$. The points on this line belong to the outer region when their z-coordinates are less than z_i^*.

The traveled distance along the sphere's outer (unoccluded) part is $\Delta z_i = z_i^* - z_i^{entry}$ when

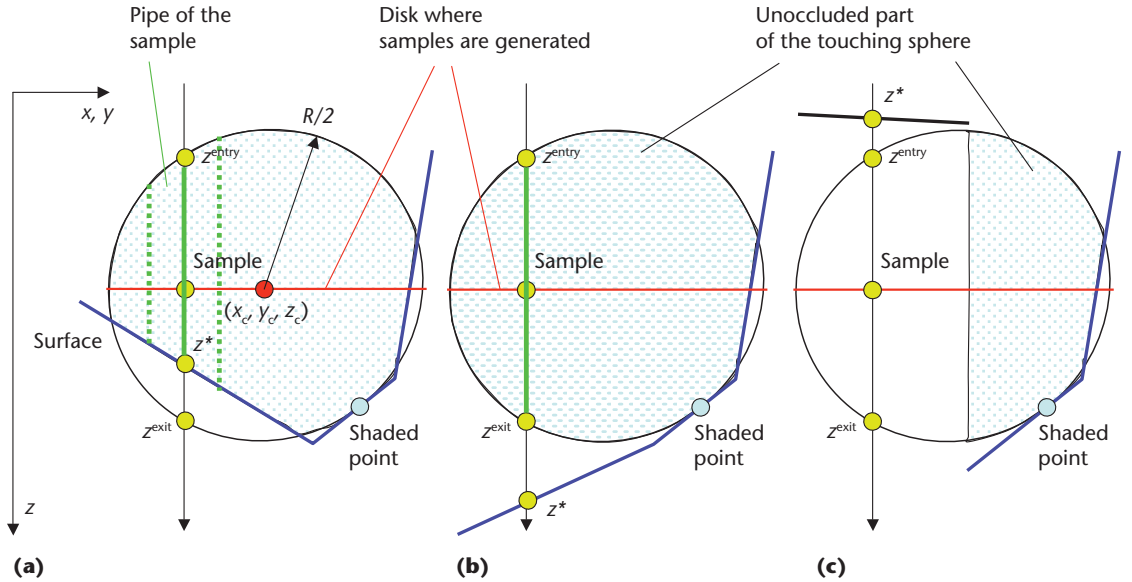


Figure 7. Computation of volumetric ambient occlusion. We approximate the volume of the tangent sphere's unoccluded part by uniformly sampling n points on the disk perpendicular to the z-axis. Each sample point represents the disk base area $(R/2)^2\pi/n$ of a pipe's cross-section inside the sphere's unoccluded part. The pipe's volume is the base area multiplied by the length Δz of the line segment in the tangent sphere's unoccluded part. To obtain this length, we identify three main cases: (a) when the line-surface intersection is in the sphere ($\Delta z = z^* - z^{entry}$), (b) when the surface is behind the sphere ($\Delta z = z^{exit} - z^{entry}$), and (c) when the surface is in front of the sphere ($\Delta z = 0$).

$z_i^{entry} \le z_i^* \le z_i^{exit}$ (see Figure 7a). The traveled distance is $\Delta z_i = z_i^{exit} - z_i^{entry}$ if the surface is behind the sphere (see Figure 7b). If $z_i^* < z_i^{entry}$, $\Delta z_i = 0$ because this part of the sphere is occluded (see Figure 7c).

If n sample points are uniformly distributed on the disk of radius $R/2$, a line is associated with the disk's $(R/2)^2\pi/n$ area. This leads to the following approximation of the volume of the tangent sphere's unoccluded part:

$$\int_S I(\vec{p})dp \approx \frac{R^2\pi}{4n}\sum_{i=1}^{n}\Delta z_i \,. \quad (6)$$

The fragment shader gets samples (x_i, y_i) as a constant array, and it estimates the point's volumetric ambient occlusion as

$$V(\vec{x}) = \frac{\int_S I(\vec{p})\,dp}{|S|} \approx \frac{3}{2Rn}\sum_{i=1}^{n}\Delta z_i = \frac{1}{F}\sum_{i=1}^{n}\Delta z_i \,. \quad (7)$$

Normalization constant F is the ratio of the tangent sphere's volume $|S| = 4(R/2)^3\pi/3$ and $R^2\pi/(4n)$, which we can compute analytically. However, rather than using the exact value of $|S|$, we can approximate it with the same samples we used to compute the unoccluded part's volume (Equation 6), resulting in this constant:

$$F = R\sum_{i=1}^{n}\sqrt{1 - x_i^2 - y_i^2} \,.$$

In this case, the approximations of the unoccluded part's volume and the tangent sphere's volume have a correlated error. So, computing their ratio reduces the error of the unoccluded part's volume, as in *weighted importance sampling*.[7]

Noise Reduction with Interleaved Sampling

The approach based on the Monte Carlo quadrature has some errors in each pixel, depending on the particular samples used. If we used different quasirandom numbers in neighboring pixels, dot noise would be present. Using the same quasirandom numbers in every pixel would make the error correlated and replace dot noise with stripes. Unfortunately, both stripes and pixel noise are quite disturbing. So, to reduce the error without taking an excessive number of samples, we apply *interleaved sampling*,[8] which uses different sets of samples in the pixels of a 4×4 pixel pattern, and we repeat the same sample structure periodically. We can obtain the 16 different sample sets from a single set by rotating the samples around the surface-normal by a random angle, α. We execute this rotation in the fragment shader, giving it 16 $[\cos(\alpha), \sin(\alpha)]$ pairs in addition to quasirandom samples (x_i, y_i). The errors in the pixels of a 4×4 pixel pattern are uncorrelated; we can successfully reduce them using a low-pass filter of the same size. Thus, interleaved sampling using a 4×4 pixel pattern multiplies the effective sample number by 16 but has only the added cost of a box-filtering with a 4×4 pixel window. 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 don't include the neighboring pixels in the averaging operation.

Results

We implemented the proposed methods in a DirectX, HLSL (High-Level Shader Language) environment, and we measured their performance on an Nvidia GeForce 8800 GTX GPU with 800×600 resolution.

Figure 8 compares Crytek's screen space ambient occlusion (SSAO) (http://en.wikipedia.org/wiki/Screen_Space_Ambient_Occlusion), containment-test-based ambient occlusion (Equation 5), volumetric ambient occlusion (Equation 7), and a reference obtained using Mental Ray's ray tracer. SSAO takes samples on the entire sphere and isn't based on the original ambient-occlusion formula. So, it assigns middle gray even for completely unoccluded surfaces, giving an unrealistically dark touch to the image. The containment-test and volumetric methods are faster than Crytek's SSAO.

For all three methods, the speed is sensitive to the neighborhood size. This sensitivity is due to the degraded cache efficiency of z-buffer accesses when the large R requires fetching distant samples. We can avoid this degradation by reducing the z-buffer resolution through an additional z-buffer filtering before the ambient-occlusion computation.

In terms of quality, volumetric ambient occlusion is the closest to the ray-traced reference. The quality degradation with respect to the ray-traced result stems mainly from the assumption that at most one intersection can occur in neighborhood R. This assumption limits the simultaneous consideration of close and distant occlusions, which reduces the level of detail in the final image. This is the price we must pay for real-time rendering.

Figure 9 evaluates the performance-quality tradeoff for containment-test-based and volumetric ambient occlusion and shows the effect of interleaved sampling. Because the volumetric approach evaluates part of the integration analytically, its results are smoother when computing only a few samples per pixel. The additional interleaved sampling helps eliminate sampling artifacts in all cases but slows down the computation and mildly blurs the image.

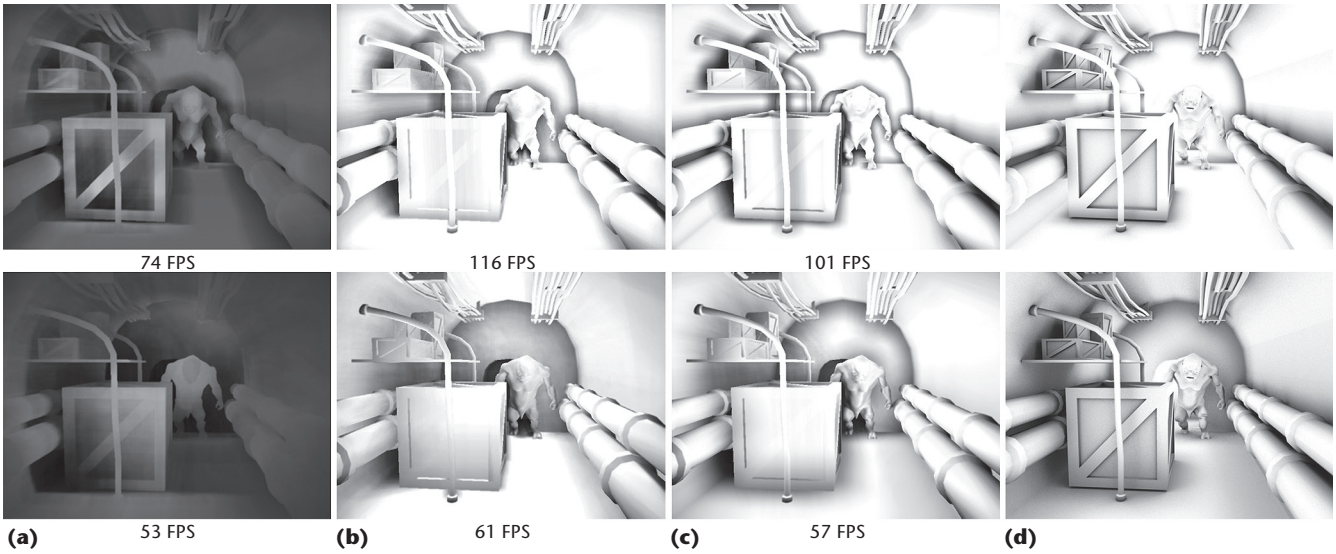


Figure 8. A comparison of (a) Crytek's screen space ambient occlusion (SSAO), (b) containment-test-based ambient occlusion, (c) volumetric ambient occlusion, and (d) a reference obtained with Mental Ray's ray tracer, for (upper row) a smaller ($R = 3$) and (lower row) a larger ($R = 9$) neighborhood. In all cases, we used 32 samples per pixel. We incorporated membership function $\mu = (r/R)^3$ in the containment test and ray-tracing methods—the same membership function implicitly used by volumetric ambient occlusion. Frame rates are in frames per second (FPS).

Figure 10 shows the power of weighted-importance sampling. We rendered the image in Figure 10a using the analytic formula (Equation 7), and in Figure 10b using weighted-importance sampling (WIS), both with only four samples per pixel. The WIS-rendered image is closer than the image rendered using the analytic formula is to the reference image rendered with 32 samples (Figure 10c).

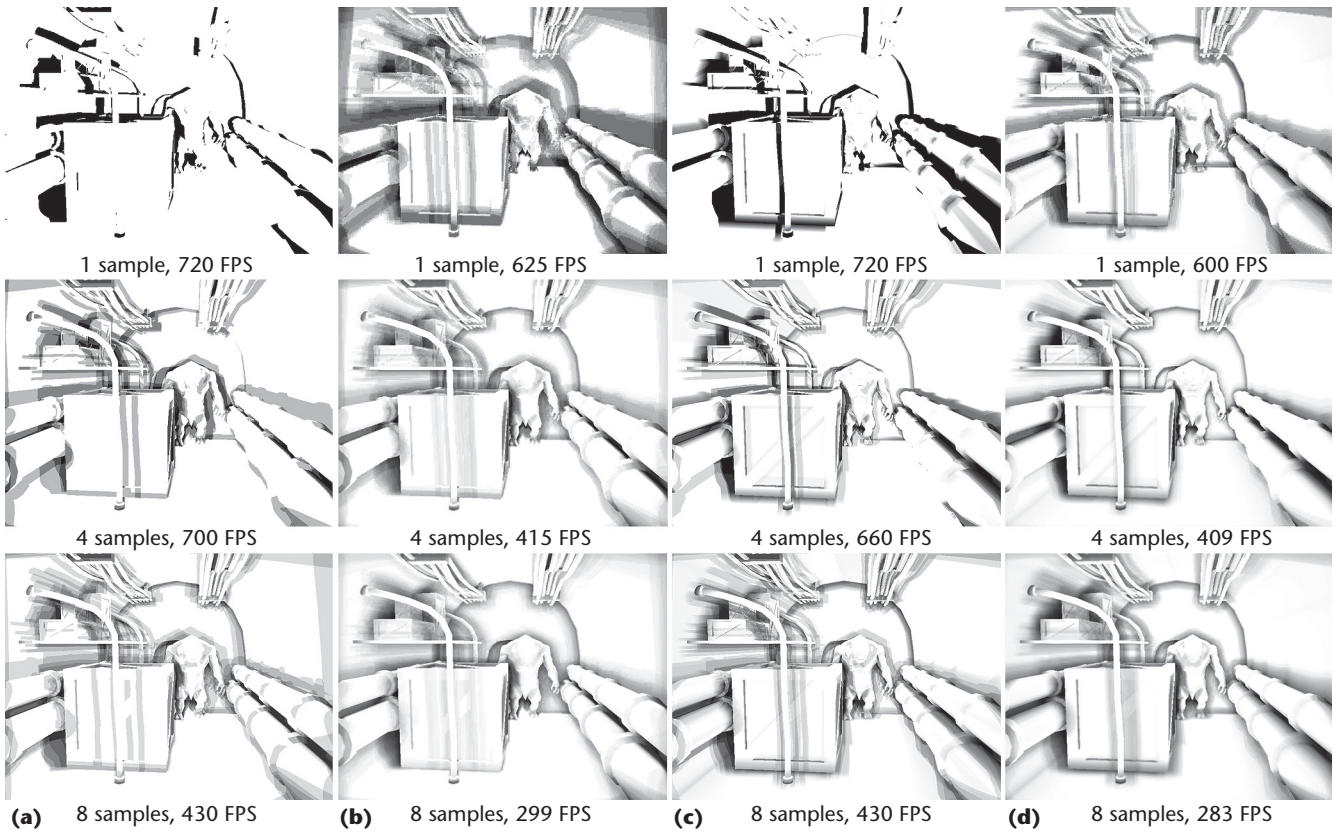
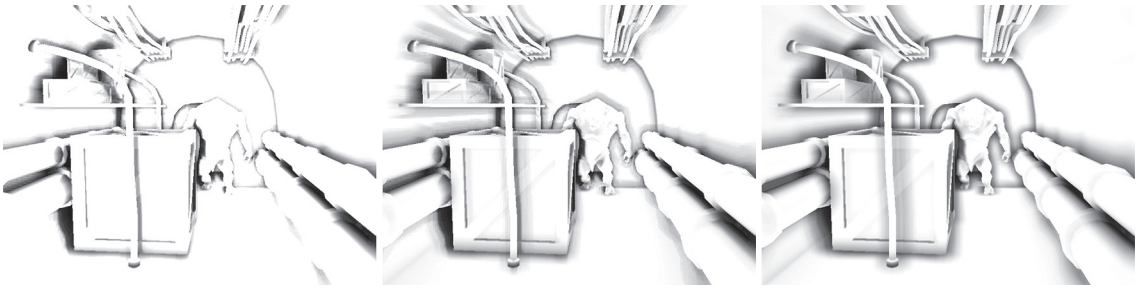


Figure 9. A comparison, using different numbers of samples per pixel, of (a) containment-test-based ambient occlusion, (b) containment-test-based ambient occlusion with interleaved sampling, (c) volumetric ambient occlusion, and (d) volumetric ambient occlusion with interleaved sampling.



(a) (b) (c)

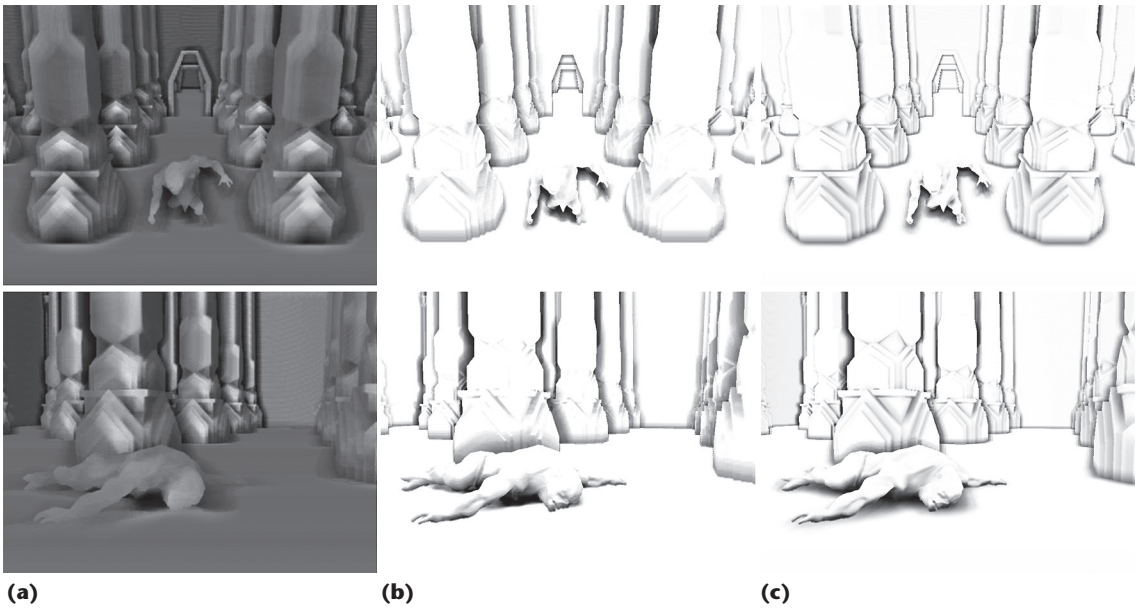
Figure 10. A comparison of images rendered using (a) the analytic formula (four samples), (b) weighted-importance sampling (WIS) (four samples), and (c) a reference image (32 samples). The image rendered using WIS is far closer to the reference image than is the image rendered using the analytic formula.

Figure 11 demonstrates the differences between Crytek's SSAO, classical ambient occlusion obtained with containment tests, and our new volumetric ambient occlusion. In all cases, we took only eight samples per pixel, and the frame rates were similar. Because volumetric ambient occlusion must integrate a lower-variation integrand, it provides smoother, better results than SSAO or the containment test.

In our implementation, the characteristic function can be a product of two characteristic functions: one separating the inner and outer parts according to the *z*-buffer, and the other handling displacement-mapped polygons (considering them as height fields defined in tangent space). The *z*-buffer-based classification handles other objects' occlusions, and the height-field-based classification handles self-shadowing. This enables obtaining the ambient-occlusion value for displacement-mapped surfaces even if the depth value isn't modified in the fragment shader, and it eliminates the *z*-buffer's accuracy problems.

Figure 12 compares images rendered with only the *z*-buffer-based characteristic function with images also including height field occlusions. It also shows the final image due to environment map lighting. We computed volumetric ambient occlusion with 12 samples per pixel. Our ambient-occlusion computation was only slightly more expensive than environment lighting yet significantly enhanced the details, at both the macro-structure-geometry and displacement-maps levels.

This article focused on a fast way to compute ambient occlusion. However, volumetric ambient occlusion can also serve as a new definition for a point's openness or accessibility. The method's key advantage over more traditional methods is that it can be evaluated more accurately with the same number of samples. This makes it more suitable for real-time systems that require low-noise results with only a few inexpensive samples. The method is also appropriate for dynamic geometry with dynamic height fields and displacement maps. In the future, we plan to extend our proposed ambient-occlusion computation method to estimate the indirect illumination coming from the neighborhood of the shaded point.



(a) (b) (c)

Figure 11. A comparison of (a) Crytek's screen space ambient occlusion (SSAO); (b) containment-test-based ambient occlusion using a step membership function, $\mu = \varepsilon(D - R)$; and (c) volumetric ambient occlusion. We used as few as eight samples per pixel. Both containment-test-based and volumetric ambient occlusion ran at approximately 250 FPS; Crytek's SSAO was slightly slower. The scene comprises 70,138 triangles.

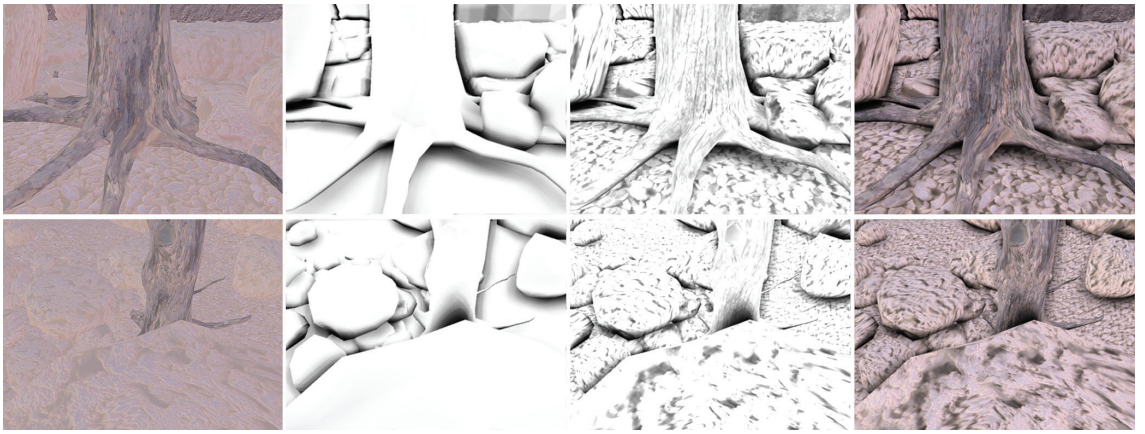


Figure 12. A tree with stones rendered by volumetric ambient occlusion, taking 12 samples per pixel: (a) environment lighting only (115 FPS), (b) *z*-buffer-based occlusions only (112 FPS), (c) *z*-buffer-based occlusions and height field occlusions (80 FPS), and (d) *z*-buffer-based occlusions and environment lighting (70 FPS). The scene comprises 109,784 triangles.

Acknowledgments

The Hungarian National Scientific Research Fund (OTKA T042735) and the Spanish government (TIN 2007-68066-C04-01) supported this research.

References

1. A. Iones et al., "Fast Realistic Lighting for Video Games," *IEEE Computer Graphics and Applications*, vol. 23, no. 3, 2003, pp. 54–64.
2. T. Annen et al., "Exponential Shadow Maps," *Proc. Graphics Interface* (GI 08), Canadian Information Processing Soc., 2008, pp. 155–161.
3. M. Mittring, "Finding Next Gen: CryEngine 2," *Siggraph 2007 Course Notes*, Course 28: Advanced Real-Time Rendering in 3D Graphics and Games, ACM Press, 2007, pp. 97–121.
4. M. Sainz, "Real-Time Depth Buffer Based Ambient Occlusion," *Proc. Games Developers Conf.* (GDC 08), GDC, 2008; http://developer.download.nvidia.com/presentations/2008/GDC/GDC08_Ambient_Occlusion.pdf.
5. M. Pharr and S. Green, "Ambient Occlusion," *GPU Gems*, R. Fernando, ed., Addison-Wesley, 2004, pp. 279–292.
6. A. Méndez, M. Sbert, and J. Catá, "Real-Time Obscurances with Color Bleeding," *Proc. 19th Spring Conf. Computer Graphics* (SCCG 03), ACM Press, 2003, pp. 171–176.
7. M. Powell and J. Swann, "Weighted Uniform Sampling—a Monte-Carlo Technique for Reducing Variance," *IMA J. Applied Mathematics*, vol. 2, no. 3, 1966, pp. 228–236.
8. A. Keller and W. Heidrich, "Interleaved Sampling," *Proc. 12th Eurographics Workshop Rendering Techniques*, Springer, 2001, pp. 269–276.

László Szirmay-Kalos *heads the Computer Graphics Group and the Department of Control Engineering and Information Technology at Budapest University of Technology and Economics. His research interests include Monte Carlo global-illumination algorithms, distributed high-performance visualization, and GPU implementation. Szirmay-Kalos has a PhD in computer graphics from the Hungarian Academy of Sciences. He is a Eurographics Association Fellow. Contact him at szirmay@iit.bme.hu.*

Tamás Umenhoffer *is an assistant professor in the Department of Control Engineering and Information Technology at Budapest University of Technology and Economics. His research focuses on games, realistic lighting effects, and medical visualization. Umenhoffer has an MS in computer science from the University of Pannonia. Contact him at umitomi@gmail.com.*

Balázs Tóth *is an assistant professor in the Department of Control Engineering and Information Technology at Budapest University of Technology and Economics. His research focuses on screen-space-rendering methods, including tone mapping, depth of field, deferred shading, and ambient occlusion, as well as the application of GPUs in medical image reconstruction. Tóth has an MS in computer science from the Budapest University of Technology and Economics. Contact him at tbalazs@sch.bme.hu.*

László Szécsi *is an assistant professor in the Department of Control Engineering and Information Technology at Budapest University of Technology and Economics. His research focuses on real-time Monte Carlo algorithms running in CPUs and GPUs. Szécsi has an MS in computer science from the Budapest University of Technology and Economics. Contact him at szecsi@iit.bme.hu.*

Mateu Sbert *is a professor of computer science at the University of Girona. His research interests include application of Monte Carlo and information theory techniques to computer graphics and image processing. Sbert has a PhD in computer science from the Technical University of Catalonia. Contact him at mateu@ima.udg.edu.*