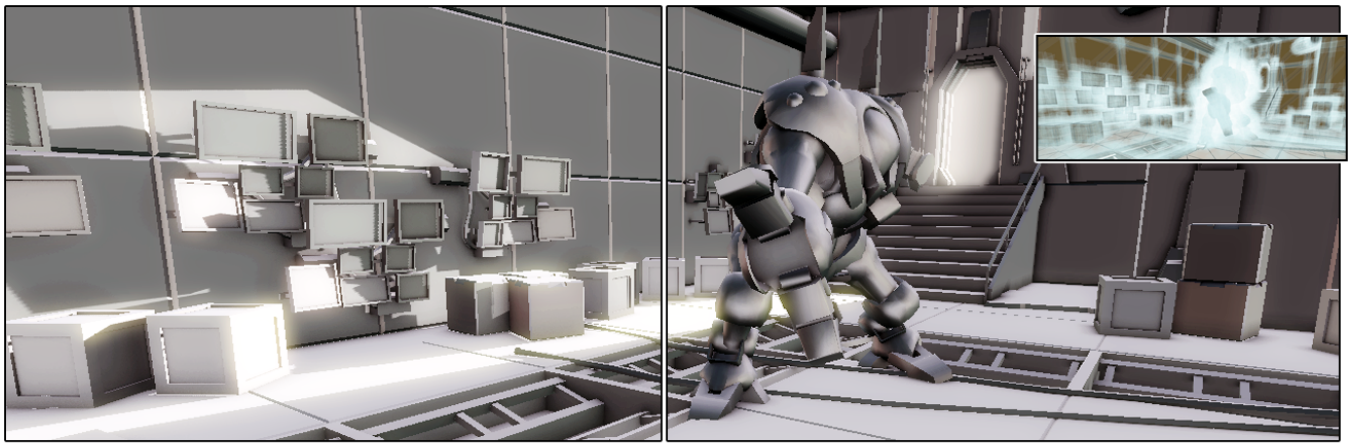
Ambient Occlusion Volumes

Morgan McGuire*
Williams College



(a) Ray traced ambient occlusion: 124,309 ms = 2.1 min / frame (b) Ambient occlusion volumes: 31 ms / frame

Figure 1: For this 1.4M-triangle scene at 1280×720 resolution, the new Ambient Occlusion Volume algorithm produces quality comparable to ray tracing 1200 visibility samples per pixel but executes in real time: a **4000× speedup**. The inset shows the occlusion volume wireframe.

Abstract

This paper introduces a new approximation algorithm for the ambient occlusion problem. It begins with an analytic solution for the cosine-weighted solid angle a polygon, which is proportional to occlusion. It then describes ambient occlusion volumes, which are like shadow volumes for ambient light. The new algorithm evaluates the analytic solution at visible points within these volumes.

The new algorithm's output approaches the quality of ray traced results, but it is thousands of times faster and operates on dynamic polygon sets. This enables interaction with high quality ambient occlusion rendering for the first time. Because the algorithm builds on an analytic solution, results are free of the noise and blurring observed in numerical sampling methods. An optional step trades this quality for greater performance by reconstructing sparsely sampled occlusion with a cross-bilateral filter. The sources of error in the new approximation are over-occlusion in areas containing very close, parallel objects, and potential aliasing when the optional sparse sampling is employed.

The new algorithm can execute efficiently on a GPU within a deferred shading pipeline. It takes as input only the original mesh and geometry buffers. A geometry shader produces the occlusion volumes and a pixel shader applies the occlusion algorithm. We demonstrate high-quality results at interactive rates for an OpenGL implementation, including motivating cases such as architectural models with fine details and video game assets rendered at 1280×720 pixels (HD 720p).

Keywords: ambient occlusion, projected solid angle, shadow volume

Technical Report CSTR-200901
December 6, 2009
Williams College Computer Science Department
Williamstown, MA 01267

*e-mail: morgan@cs.williams.edu

1 Introduction

Ambient illumination is an approximation to the light reflected from the sky and other objects in a scene. Ambient occlusion (AO) is the darkening that occurs when this illumination is locally blocked by another object or a nearby fold in a surface. Both ambient illumination and occlusion are qualitatively important to perception of shape and depth in the real world and can be quantified by specific terms in the integral equation for light transport (see section 2).

The problem of computing an explicit AO factor has been studied for over a decade, with significant recent advances. Many AO algorithms cluster at the extremes of the performance-vs.-quality curve. For example, screen-space GPU methods are extremely fast but coarse approximations, and ray tracing is very accurate but converges too slowly for interaction. Games on today's console GPUs are limited by graphics performance, so they require the fastest approximations. A few live-action film effects and civil engineering daylight simulations require the highest accuracy available. For all other applications, extremes are poor tradeoffs. This includes modeling, CAD, and visualization; games on *tomorrow's* hardware; and most film rendering. For those applications, an algorithm that both is fast and has high quality results is preferred over maximizing one property at the expense of the other, especially this if enables interaction with previously offline "preview" rendering.

The primary contribution of this paper is an efficient new algorithm called Ambient Occlusion Volume rendering (AOV) for estimating the ambient occlusion based on analytic solution to the occlusion integral. It provides both qualitative and quantitative analysis against three of the best alternative algorithms.

The new AOV algorithm is viewer independent, produces no noise or aliasing (beyond that of rasterization itself), correctly includes the "cosine" $\hat{\omega} \cdot \hat{n}$ factor, requires no preprocessing, and operates directly on meshes. It offers quality approaching that of offline ray tracing, yet at interactive rates. An optional subsampling and reconstruction step smoothly trades accuracy for performance suitable for immersive real-time applications like games. We demonstrate render times from 10-200 ms per frame, depending on scene complexity and sampling rate choice. The drawbacks of

our algorithm are that it requires tessellating curved surfaces into meshes, it double-counts occlusion where thin objects are stacked, and that the run time is linear in visible scene complexity. Because it uses a z-buffer, AOV only shades one visible surface per pixel, although like other deferred-rendering algorithms it can be extended to support translucency via recent multisample methods [Kircher and Lawrance 2009; Bavoil et al. 2007].

2 Ambient Occlusion Problem Statement

This section formalizes ambient occlusion and derives its physical basis. The rendering integral equation for exitant radiance at point $\vec{x}$ in direction $\hat{\omega}_o$ is:

$$L_o(\vec{x},\hat{\omega}_o) = L_e(\vec{x},\hat{\omega}_o) + \int_{\mathbf{S}^2} L_i(\vec{x},\hat{\omega}_i) f(\vec{x},\hat{\omega}_i,\hat{\omega}_o)(\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i \quad (1)$$

An image is the solution to incident radiance L_i at points on the image plane plane and directions toward the aperture. Incident light can be separated into terms due to *nearby* objects (L_n); large, distant areas "*ambient*" (L_a); and relatively small, distant "*sources*" (L_s):

$$L_i(\vec{x},\hat{\omega}_i) = L_n(\vec{x},\hat{\omega}_i)\,(1-\mathscr{V}(\vec{x},\vec{y})) + (L_a(\vec{y},\hat{\omega}_i) + L_s(\vec{y},\hat{\omega}_i))\,\mathscr{V}(\vec{x},\vec{y}),$$

where $\vec{y} = \vec{x} + \hat{\omega}_i\delta$ is a point on the sphere about $\vec{x}$ at distance δ. This is the sphere that separates "near" from "distant". Let visibility function $\mathscr{V}(\vec{x},\vec{y}) = 1$ if there is an unobstructed line of sight between $\vec{x}$ and $\vec{y}$ and 0 otherwise. The rendering equation is then (with implicit arguments and only considering the hemisphere $\mathbf{S}^2_+$ about $\hat{n}$, for an opaque surface):

$$\begin{aligned} L_o = {} & L_e && \text{"emitted"} && (2)\\ & + \int_{\mathbf{S}^2_+} L_s \cdot f \cdot \mathscr{V} \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i && \text{"direct"} && (3)\\ & + \int_{\mathbf{S}^2_+} L_n \cdot f \cdot (1-\mathscr{V}) \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i && \text{"indirect"} && (4)\\ & + \int_{\mathbf{S}^2_+} L_a \cdot f \cdot \mathscr{V} \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i. && \text{"ambient"} && (5) \end{aligned}$$

A common approximation is to then factor the ambient term into

$$\int_{\mathbf{S}^2_+} L_a \cdot f \cdot \mathscr{V} \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i \approx \quad (6)$$

$$\left[\pi \int_{\mathbf{S}^2_+} L_a \cdot f\, d\hat{\omega}_i\right] \cdot \left[\frac{1}{\pi}\int_{\mathbf{S}^2_+} \mathscr{V} \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i\right]. \quad (7)$$

This is only an approximation because multiplication and integration do not commute, except for constants. That is, unless distant light L_a is independent of direction and f is Lambertian (which explains Phong's ambient term). However, this approximation is reasonable if both functions are relatively smooth over most of the sphere, which is the case for a typical sky-and-ground model and Lambertian-plus-glossy BRDF. In this case, the left bracketed factor on line 7 can be precomputed; for real-time rendering the result is typically encoded in a MIP-mapped cube map [Scheuermann and Isidoro 2005] for negligible run-time cost. This is how we lit the scene in figure 1. The right factor in eq. 7 is a scalar between 0 and 1 indicating the fractional **accessibility** of a point.

Because objects typically have explicit representations and empty space is implicit in most scene models, accessibility is often expressed in terms of **ambient occlusion**:

$$AO = \frac{1}{\pi}\int_{\mathbf{S}^2_+} (1-\mathscr{V}) \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i = 1 - \frac{1}{\pi}\int_{\mathbf{S}^2_+} \mathscr{V} \cdot (\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i \quad (8)$$

A hard cutoff at distance δ would reveal the distinction between different methods used for computing visibility at different scales (e.g., ambient occlusion vs. shadow map and no area occlusion), so it is common to replace binary occlusion $1-\mathscr{V}(\vec{x},\vec{y})$ with fractional **obscurance** [Zhukov et al. 1998] $(1-\mathscr{V}(\vec{x},\vec{y}))g(\vec{x},\vec{y})$, where **falloff function** g is smooth, monotonic, and is 0 for $||\vec{x}-\vec{y}|| \geq \delta$ and 1 at $\vec{x} = \vec{y}$.

3 Related Work

3.1 Shadow Volumes

The basic idea of the AOV algorithm is that an ambient occlusion volume is to area or indirect light as Crow's shadow volumes [1977] are to point sources. Like shadow volumes, AOV rendering benefits from high fill-rate, hierarchical and early-out depth tests, geometry shaders, and depth clamping features in GPUs.AOVs are much smaller than shadow volumes (figure 5), so they avoid some shadow volume drawbacks: our near-plane clipping issues are easy to resolve, we spend comparatively little fill-rate rasterizing over empty space, and no stencil buffer is required because the point-in-volume test is explicit in a pixel shader.

3.2 Common Ideas

This subsection relates our work to key ideas common in previous work. See chapter 9 of Akenine-Möller et al.'s text [2008] for a thorough survey of AO work.

A common approximation to the the cosine-weighted solid angle of a sphere (or disk) enables estimating AO for scenes modeled by bounding spheres [Bunnell 2005; Hegeman et al. 2006; Ren et al. 2006; Shanmugam and Arikan 2007]. We contribute an exact analytic solution for polygons so that analytic methods can directly operate on meshes.

AO is low frequency across a plane, so any AO method can be accelerated by reconstruction from sparse samples. The cross bilateral filter [Eisemann and Durand 2004; Petschnigg et al. 2004] with the gaussian kernel that falls off with surface normals and the depth buffer discrepancy is the common reconstruction filter. AO is an integral over $d\vec{x}$ and $d\hat{\omega}$. If the sparsely sampled dimension is angular ($d\hat{\omega}$) [Mittring 2007; Bavoil and Sainz 2009; Filion and McNaughton 2008], the reconstruction filter masks noise. If that dimension is spatial ($d\vec{x}$) [Bavoil and Sainz 2009; Reinbothe et al. 2009], the filter masks aliasing. In both cases, undersampling is a source of error, particularly where geometry changes rapidly in screen space.

We apply sparse spatial sampling in our results where explicitly noted in our results. We denote sparse sampling as the product of amortized samples taken per pixel and reconstruction filter size. Thus $1/9 * 5\times5$ denotes one sample/pixel at a 1/3 scale buffer followed by a 5×5 reconstruction kernel and $16/4 * 5\times5$ denotes 16 samples/pixel for a 1/2 scale buffer followed by a 5×5 filter.

Almost any AO algorithm can simultaneously compute diffuse interreflection at minimal incremental cost [Zhukov et al. 1998; Bunnell 2005; Evans 2006; Ritschel et al. 2009]. We focus exclusively on pure AO (eq. 8) in this paper.

3.3 Physically-Based Methods

Physically-based methods generally target off-line rendering applications such as "beauty render" stills, architectural visualizations, and cinema-quality animation. They produce accurate results but most require minutes to render a frame.

Monte-carlo ray tracing is the gold standard for ambient occlusion. Any global illumination algorithm also captures the effect, although ray tracing is preferred because, e.g., photon mapping and radiosity tend to overly smooth AO and Metropolis light transport and path tracing produce significant noise.

Zhukov et al. [1998] introduced ambient occlusion and obscurance in the context of the radiosity algorithm. They derived an

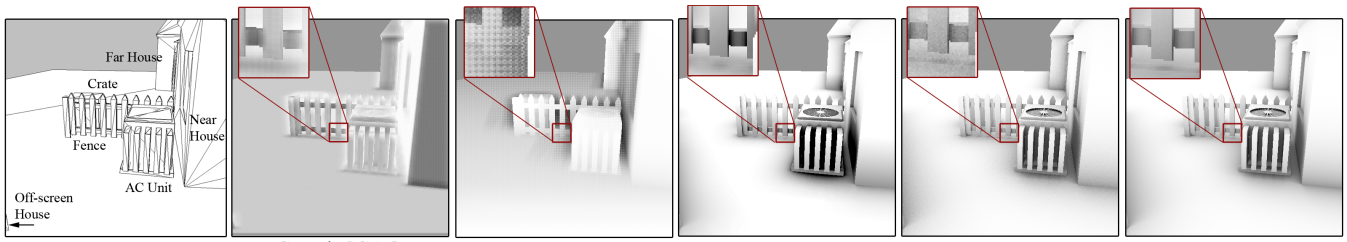


Figure 2: This "Suburb" stress-test scene contains close proximity between surfaces, varying depth discontinuities, large off-screen occluders, and steep screen-space slopes. Various algorithms exhibit aliasing, noise, and over-occlusion compared to the far right ray traced reference.

analytic approximation to the form factor between points on differential patches and apply this to occlusion. We apply this approach to whole polygons.

Bunnell [2005] introduced a purely geometric method. It requires preprocessing the scene into a set of disks with bent normals. His algorithm computes approximate analytic occlusion between the disks. It is designed to operate as a general purpose computation on a GPU, and variations on it have been used in film production. To resolve double-counting, it iterates over the disks multiple times and decreases the AO contribution of a disk by its own occlusion on the previous pass.

Reinbothe et al.'s Hybrid AO [2009] traces rays against a voxelized scene and then corrects high-frequency features with a less accurate SSAO pass. Performance and accuracy can be traded by altering the emphasis and constants of these two passes, similar to our AOV algorithm.Sloan et al.'s image-based global illumination method [2007] generates accurate ambient occlusion and indirect illumination in real-time for small scenes using virtual light probes.

Baum et al. [1989] are the first we are aware of to combine correct analytic form factors with numerical integration for global illumination. We show that their analytic form factor expression is also equivalent to the ambient occlusion between two surfaces and apply it to the AO problem. Their numerical sampling was over polygon patches, which gives blocky results. By incorporating a deferred-shading frame work and computing the bounding geometry for a falloff function, we can directly sample occlusion at each pixel.

3.4 Phenomenological Methods

An artist might describe ambient occlusion as: nearby objects darken each other ("contact shadows"), these shadows darkening and grow distinct with proximity ("contact hardening"), and concave regions of an object are dark. Most real-time AO methods trade accuracy for significant performance gains by directly simulating these phenomena instead of light transport.

Hegeman et al. [2006] recognized that AO is essential to the rendering of foliage, which is now a standard test (see figure 11 row 4). They coarsely approximated trees with bounding spheres and grass with occlusion gradients. Luft et al.'s seminal unsharp masking paper [2006] introduced the screen space ambient occlusion (SSAO) approach: they treat the depth buffer as a heightfield and identify concave regions by filtering. They are careful to point out that this has only passing resemblance to actual ambient occlusion, however it remarkably improves the perception of depth and they demonstrate applications in visualization.

SSAO methods are an excellent approach for immersive applications like games requiring very fast and scene-invariant run time per frame. The Crytek SSAO [Mittring 2007] algorithm adapted unsharp masking for games by sparsely sampling visibility rays against the depth buffer and filtering the result.The Crytek algorithm produces artifacts like over-brightening of concave regions, noise, aliasing, and halos around objects (figure 2), yet is so efficient that it quickly became the standard method for real-time AO in games. Subsequent techniques improved SSAO quality at varying performance by: adding distant occluders [Shanmugam and Arikan 2007], directional occlusion and indirect illumination [Ritschel et al. 2009], better filtering and sampling [Shanmugam and Arikan 2007; Filion and McNaughton 2008; Bavoil and Sainz 2009], and better obscurance [Szirmay-Kalos et al. 2009]. Viewer dependence and limited sampling range remains inherent problems with all SSAO methods. We show running comparisons between AOV and **Volumetric AO** [Szirmay-Kalos et al. 2009], which is the fastest, most recent, and most physically-correct screen space method.

Evans [2006] precomputed voxel signed-distance fields around static meshes by rasterization and then estimated occlusion by convexity. He also applied this technique to indirect illumination. Kontkanen and Laine [2005] extended this to a phyiscally-based method for AO only by precomputing the actual occlusion on a voxel grid. Both methods are restricted to rigid objects, trade space for time, and produce double-counting artifacts. However, they give results comparable to ray tracing and are very fast. The benefits of these methods inspired us extend their *voxel* ambient occlusion volume idea to *geometric* ambient occlusion volumes, which are fully dynamic and require no storage beyond the original mesh.

4 Analytic Solution to Polygon Occlusion

Ambient occlusion arises in the rendering equation from the integral of a cosine factor over a solid angle. We transform this integral into a series sum on polygon vertices and then derive an efficient, analytic solution for the series terms. The key observation is that the cosine-weighted solid angle (a.k.a. the "projected solid angle") of a polygon is equal to the area of its double projection: onto the unit sphere and then onto the tangent plane. This is also equivalent to the radiative transfer between a point and a polygon, i.e., the form factor from the radiosity framework. An analytic solution to that has long been known from heat transfer literature and was introduced to graphics by Baum et al. [1989] as:

$$\mathrm{AO}_P = \frac{1}{2\pi}\sum_{i=0}^{k-1} \cos^{-1}\left(\frac{\vec{p}_i \cdot \vec{p}_j}{||\vec{p}_i||\,||\vec{p}_j||}\right)\hat{n}\cdot\frac{\vec{p}_i \times \vec{p}_j}{||\vec{p}_i \times \vec{p}_j||} \qquad (9)$$

where $j = (i+1) \mod k$.

Our derivation differs slightly from the classical one in that it reduces the problem from one on 3D vectors to an expression on 2D vectors within the tangent plane of the occluded surface. Those 2D vectors are readily available after the necessary clipping to the tangent plane, which can reduce the overall operation count in the implementation.

4.1 Notation

Without loss of generality, assume that the point being occluded is at the origin $\vec{0}$. It is on some surface with normal $\hat{n}$, which defines a local tangent plane to the surface (figure 3). For any point $\vec{x} \neq \vec{0}$ in the positive half space of the tangent plane, let $\mathrm{S}(\vec{x}) = \vec{x}/||\vec{x}||$ denote its projection onto the unit sphere $\mathbf{S}^2$, and let $\mathrm{T}(\vec{x}) = \vec{x} - \hat{n}(\hat{n} \cdot \vec{x})$ denote its projection onto the tangent plane. Let the definition of these operators be overloaded to operate on sets of points as well.

A **spherical polygon** has vertices in $\mathbf{S}^2$ and its edges are great circle arcs. The spherical polyhedron defined by ordered vertex list $(\hat{p}_0, ..., \hat{p}_{k-1})$ subtends the same solid angle about $\vec{0}$ as any polygon $(\vec{v}_0, ..., \vec{v}_{k-1})$ in R^3 for which $\mathrm{S}(\vec{v}_i) = \hat{p}_i,\ \forall\ 0 \leq i < k$. Our final result holds for arbitrary (e.g., disconnected) point sets on a plane with polygonal boundaries, although to simplify equation 11 we give the derivation only for simple polygons.

4.2 Ambient Occlusion: AO

In the context of the rendering equation, the **ambient occlusion** of the unit hemisphere $\mathbf{S}^2_+$ by a planar polygon P whose vertices $(\vec{p}_0, ..., \vec{p}_{k-1})$ are above the tangent plane is

$$\mathrm{AO}_P = \frac{1}{\pi} \int_{\mathbf{S}^2_+} (1 - \mathscr{V}_P(\hat{\omega}))(\hat{\omega} \cdot \hat{n})\, d\hat{\omega}, \tag{10}$$

where **visibility function** $\mathscr{V}_P(\hat{\omega}) = 0$ if a ray in direction $\hat{\omega}$ from the origin intersects the polygon and $\mathscr{V}_P(\hat{\omega}) = 1$ otherwise. Note that AO_P is the cosine-weighted solid angle of P divided by $\int_{\mathbf{S}^2_+} (\hat{\omega} \cdot \hat{n})\, d\hat{\omega} = \pi$.

We change the integration domain to eliminate the visibility function and cosine weighting:

$$\mathrm{AO}_P \quad = \quad \frac{1}{\pi} \int_{\mathrm{S}(P)} (\hat{\omega} \cdot \hat{n})\, d\hat{\omega} \quad = \quad \frac{1}{\pi} \int_{\mathrm{T}(\mathrm{S}(P))} 1\, d\vec{x}. \tag{11}$$

The integrand is now trivial, but the integration domain has become more complex. Domain $\mathrm{T}(\mathrm{S}(P))$ is a planar shape bounded by set of elliptic arcs about the origin, as shown in figure 4. It is not a polygon. The area of a planar *polygon* is the sum of the signed areas of the triangles formed by an arbitrary common point in the plane and the polygon edges. We can generalize this to a sum of signed areas for the individual arcs in $\mathrm{T}(\mathrm{S}(P))$, thus

$$\mathrm{AO}_P = \frac{1}{\pi} \sum_{i=0}^{k-1} \mathscr{A}\left(\mathrm{S}(\vec{p}_i), \mathrm{S}(\vec{p}_{(i+1) \bmod k})\right), \tag{12}$$

where $\mathscr{A}(\hat{a}, \hat{b})$ is the signed area of the projection of the disk segment $(\vec{0}, \hat{a}, \hat{b})$ into an elliptic segment on the tangent plane.

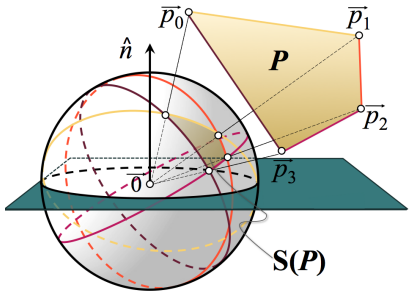


Figure 3: Spherical polygon $\mathrm{S}(P)$ is the projection of Cartesian polygon P onto the unit sphere. Both subtend the same solid angle and thus produce the same occlusion.

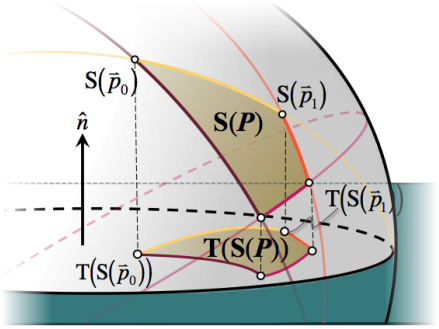


Figure 4: $\mathrm{T}(\mathrm{S}(P))$ is the projection of $\mathrm{S}(P)$ onto the plane through the origin with normal $\hat{n}$. Its area is equal to the cosine-weighted solid angle of P, which is equal to π times the ambient occlusion from P.

4.3 Area of an Elliptic Segment: $\mathscr{A}$

The intersection of a plane $L : \vec{x} \mid \vec{x} \cdot \hat{m} = 0$ and $\mathbf{S}^2$ is a great circle $C : \hat{x}(\theta) = \hat{s}\cos\theta + \hat{t}\sin\theta$ where $\hat{s}$ and $\hat{t}$ are arbitrary orthogonal unit vectors for which $\hat{s} \times \hat{t} = \hat{m}$. Circle C projects to ellipse

$$\mathrm{T}(C) = E : \vec{x}(\theta) = \mathrm{T}(\hat{s})\cos\theta + \mathrm{T}(\hat{t})\sin\theta \tag{13}$$

in the tangent plane. Note that in general, $\mathrm{T}(\hat{s})$ and $\mathrm{T}(\hat{t})$ do not have unit length and are not the semi-axes of the ellipse.

Ellipse E has radius 1 along its major semi-axis and radius $|\hat{n} \cdot \hat{m}|$ along its minor axis. It is related to the unit circle by a non-uniform scale compression along the minor axis. That transformation preserves relative areas. Therefore, the relative area of an elliptic segment is the same as the relative area of a unit-circle segment when the ellipse is that circle compressed along the minor axis to $|\hat{n} \cdot \hat{m}|$. The signed area of the elliptic segment from

$$\mathrm{T}(\hat{a}) \quad = \quad \mathrm{T}(\hat{s})\cos\theta_a + \mathrm{T}(\hat{t})\sin\theta_a \qquad \text{to} \tag{14}$$

$$\mathrm{T}(\hat{b}) \quad = \quad \mathrm{T}(\hat{s})\cos\theta_b + \mathrm{T}(\hat{t})\sin\theta_b \tag{15}$$

is therefore

$$\mathscr{A}(\hat{a}, \hat{b}) = \frac{1}{2}\,[\theta_b - \theta_a]\,[\hat{n} \cdot \hat{m}]. \tag{16}$$

Choosing

$$\hat{s} \quad = \quad \hat{a}, \tag{17}$$

$$\hat{t} \quad = \quad \mathrm{S}\left(\hat{b} - \hat{a}(\hat{a} \cdot \hat{b})\right) \tag{18}$$

gives $\theta_a = 0$, $\hat{m} = \mathrm{S}\left(\hat{a} \times \hat{b}\right)$, and $\theta_b =$ measure of the angle between $\hat{a}$ and $\hat{b}$. Both $\hat{a}$ and $\hat{b}$ have unit length, so by the law of cosines,

$$\mathscr{A}(\hat{a}, \hat{b}) = -\frac{1}{2}\left[\cos^{-1}\left(1 - \frac{1}{2}||\hat{a} - \hat{b}||^2\right)\right]\left[\hat{n} \cdot \mathrm{S}\left(\hat{a} \times \hat{b}\right)\right] \tag{19}$$

We note two important properties of this relation. First, the area is signed. When $\mathscr{A}$ is positive, the elliptic arc winds counter-clockwise about $\hat{n}$.Second, projected area correctly falls to zero when $\vec{0}$, $\mathrm{T}(\hat{a})$, and $\mathrm{T}\left(\hat{b}\right)$ are collinear.

5 Ambient Occlusion Volume Algorithm

We now extend the analytic solution in equation 12 for occlusion of one point by one polygon to an approximation algorithm for the ambient occlusion of all visible points by a set of polygons. To guide implementers, we describe it in the context of OpenGL.

The algorithm takes typical deferred rendering inputs: a set of polygons, a camera, a depth buffer, and a normal buffer.

1. Initialize a screen-space **accessibility buffer** to 1 at each pixel
2. Disable depth write, enable depth test, and enable depth clamp (`GL_ARB_depth_clamp`) to prevent near-plane clipping
3. (**Vertex Shader:**) Transform all scene vertices as if rendering visible geometry, e.g., skinning and modelview transform
4. (**Geometry Shader:**) For each polygon P in the scene:
   i. Let the ambient occlusion volume V be the region over which obscurance falloff function $g_P > 0$
   ii. Construct a series of polygons $\{B\}$ that bound V
   iii. If the camera is inside V, replace $\{B\}$ with a full-screen rectangle at the near clipping plane.
   iv. (**Pixel Shader:**) For each visible point $\vec{x} \in V$ conservatively identified by rasterizing $\{B\}$:
      a. Let P' be P clipped to the positive half space of the tangent plane at $\vec{x}$
      b. Decrement the accessibility buffer at $\vec{x}$ by $g_{P'}(\vec{x}) \cdot \mathrm{AO}_{P'}(\vec{x})$ via subtractive, saturating alpha blending

Modulate the ambient illumination term (equation 5) by the accessibility buffer during a subsequent forward or deferred shading pass, as if it were a shadow map or stencil buffer for ambient illumination. A list of quadrilaterals is a good representation for $\{B\}$, however, GLSL v1.50.09 can only output triangle strips from a geometry shader. Under current APIs one must therefore either convert the quadrilateral list to a triangle strip with some degenerate elements; or construct all B in a separate pass over the scene geometry. Three implementation choices for that pass are OpenGL transform feedback, an OpenCL or CUDA program, and CPU vertex and geometry shaders. For simplicity we used the latter.

5.1 Falloff Function g

Consider a convex polygon P with vertices $(\vec{p}_0, ..., \vec{p}_{k-1})$, no three of which are collinear. The falloff function should be monotonic in distance from P and map distances $0 \to 1$ and $\delta \to 0$. We choose:

$$g(\vec{x}) = \bar{\alpha} \prod_{i=0}^{k} \max\left(0, \min\left(1, (\vec{x} - \vec{p}_{i \bmod k}) \cdot \hat{m}_i / \delta + 1\right)\right), \qquad (20)$$

where $\bar{\alpha} = 1$ for solid surfaces, $\hat{m}_{i<k}$ are the (inward facing) normals to the edges of P shown in figure 6, and $\hat{m}_k$ is the negative normal to P:

$$\hat{m}_k = \mathrm{S}((\vec{p}_2 - \vec{p}_0) \times (\vec{p}_1 - \vec{p}_0)) \qquad (21)$$

$$\hat{m}_{0 \le i < k} = \mathrm{S}\Big((\vec{p}_{(i+1) \bmod k} - \vec{p}_i) \times \vec{m}_k\Big) \qquad (22)$$

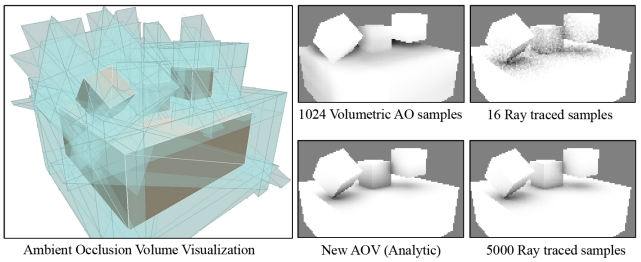


Figure 5: Ambient occlusion volume visualization and accessibility (=1-AO) buffers computed by three algorithms for a simple scene.

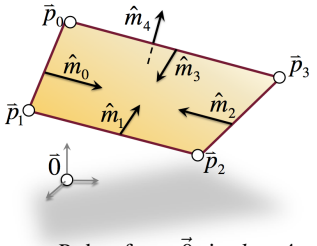


Figure 6: A polygon P that faces $\vec{0}$, its $k = 4$ vertices $\vec{p}_0, ..., \vec{p}_3$, inward edge normals $\hat{m}_0, ..., \hat{m}_3$, and negative face normal $\hat{m}_4$.

5.2 Bounding Volume $\{B\}$

The ambient occlusion volume of P is bounded by $k+2$ planes. Let planes $B_0 ... B_{k-1}$ correspond to the polygon edges, plane B_k be the one above P, plane B_{k+1} contain P, and $\hat{m}_i$ be the normal to B_i. For a maximum obscurance distance δ,

$$B_i : \ \vec{x} \mid (\vec{x} - \vec{p}_i) \cdot \hat{m}_i = \delta \qquad (23)$$

where $\vec{p}_k = \vec{p}_0$ and $\vec{p}_{k+1} = \vec{p}_0 + \delta \hat{m}_{k+1}$.

Note that we ignore the plane B_{k+2} containing the polygon because we want no falloff near the polygon itself. Because the AOV algorithm is defined on a point at the origin, g will always be evaluated at $\vec{x} = 0$ but the $\vec{p}_i$ will change. Falloff equation 20 thus simplifies to:

$$g(\vec{0}) = \bar{\alpha} \prod_{i=0}^{k} \max\left(0, \min\left(1, 1 - \vec{p}_{i \bmod k} \cdot \vec{m}_i / \delta\right)\right). \qquad (24)$$

We use rasterization to efficiently find all visible points within an ambient occlusion volume. Let the volume bounded by these planes be defined by a polyhedron V with vertices $(\vec{v}_0, ... \vec{v}_{2k-1})$, where the first k vertices are in the plane of P and the second three are displaced along the plane normal from them.

Let **extension vectors** $\vec{e}_i = \vec{v}_i - \vec{p}_{i \bmod k}$ be the displacements of polygon vertices to volume vertices. Because the bounding planes of the volume are offset along the inward facing edge normals $\hat{m}_i$ by the maximum obscurance distance and are in the plane of P, $\vec{e}_{i<k}$ is constrained by:

$$\vec{e}_i \cdot -\hat{m}_i = \delta \qquad (25)$$

$$\vec{e}_i \cdot -\hat{m}_{(i+1) \bmod k} = \delta \qquad (26)$$

$$\vec{e}_i \cdot \hat{m}_k = 0 \qquad (27)$$

Extension vector $\vec{e}_i$ is therefore given by the solution:

$$\vec{e}_{0 \le i < k} = \begin{bmatrix} \hat{m}_i \\ \hat{m}_{(i+1) \bmod k} \\ \hat{m}_k \end{bmatrix}^{-1} \begin{bmatrix} -\delta \\ -\delta \\ 0 \end{bmatrix} \qquad (28)$$

$$\vec{e}_{k \le i < 2k} = \vec{e}_{i-k} - \delta \hat{m}_k \qquad (29)$$

Sliver polygons create long volumes yet little occlusion, so as a practical measure we clamp all $||\vec{e}_i||$ to at most 2δ.

5.3 Masked Polygons and $\bar{\alpha}$

Artists often model planar surfaces with complex contours, such as foliage or a fence, as α-masked polygons. Because the underlying geometry does not match the actual occlusion properties of such a surface, its ambient occlusion volume will result in an over-estimate of occlusion. We therefore weigh the occlusion due to a surface by its average α value, $\bar{\alpha}$, which can be determined efficiently in the geometry shader by a texture fetch from a low MIP level. As with regular α blending and testing, this computation need only be performed for surfaces that are tagged as having a mask.

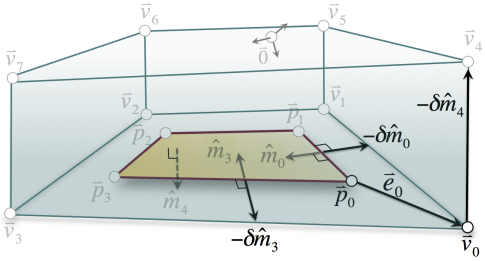


Figure 7: Visualization of the ambient occlusion volume V cast by a polygon P. The volume's base is formed by extending each original polygon vertex $\vec{p}_i$ along a vector $\vec{e}_i$ derived from the normals $\hat{m}$ of the adjacent edges and maximum obscurance distance δ.

6 Results

For all algorithm comparisons we processed indexed triangle meshes; quad meshes render about 35% faster under AOV but are less common in GPU rendering. All algorithms were given identical G-buffers as input and computed the AO factor only. The ray tracer used a bounding volume hierarchy and ran on 8 cores of an Intel Dual-Quad Core2 processor. GPU algorithms executed on an NVIDIA GeForce 280 GT GPU. All images are at 1280×720 unless noted.For figure 1, both sides of the image were lit on the GPU with one shadow-mapped spot light and a cube environment map the AO computation.

As a rough metric for implementation difficulty, our full-resolution ambient occlusion volume implementation added 229 C++ and GLSL statements and one draw call to a deferred renderer.

6.1 Qualitative Results

Figure 11 shows selected results from multiple algorithms and scenes. The PDF version of this paper contains full resolution images that can be zoomed to see pixel-level detail. The left-most column shows the reference ray traced result against which we measure the others. The 2nd column is our AOV algorithm rendered without sparse sampling; it overdarkens multiply occluded areas but is generally faithful to the reference result.

The right-most column of results are by the Crytek SSAO algorithm [Mittring 2007], which is a common baseline in AO literature and a defacto standard among game developers because of its performance. The incorrect gray flat surfaces and white halos are consistent with Mittring's published results. The third column is Volumetric AO [Szirmay-Kalos et al. 2009], which is both the most recent and most efficient published AO algorithm.The stippling and black halos in the Volumetric results are consistent with images from their original paper.

Note that Crytek SSAO uses no falloff, Volumetric AO uses cubic falloff, and our falloff is the product of many factors. Yet we measure error results against linear falloff in a ray tracer, which a nonlinear algorithm could not possibly match. We still believe this is a good metric. The algorithms with non-linear falloff functions use them because any other falloff would be less efficient in those algorithms. Artists seem satisfied with linear falloff in MentalRay and other popular renderers, so varying from that for efficiency (as we do) compromises quality. We compensate by choosing $\delta_{AOV} = 0.6\delta_{ray\ trace}$ and $\delta_{volumetric} = 1.1\delta_{ray\ trace}$, which minimized their error on test scenes.

The first row of figure 11 is the standard "Sponza" benchmark model. The AOV result is comparable to the ray traced reference. Note the black halos on columns in the Volumetric result.

"City" demonstrates occlusion created and received by an α-masked surface. The chain link fence contains only two textured triangles (excepting the posts). The Volumetric AO's black halos around the fence are a drawback of that method. The AOV algorithm creates a single occlusion volume for the entire fence, but with the correct $\bar{\alpha}$ value it is close to the reference. The white line under the fence in the ray traced result is an artifact where occlusion rays miss the fence because they are nearly parallel to it.

"House" is a simple architectural model. It shows that the primary artifact of AOV is overdarkening. Note that Volumetric AO misses the windows and stairs because the depth discrepancy is small. "Trees" contains many α-masked polygons with high depth complexity. All algorithms perform well, although Volumetric AO self-occludes the ground plane and has excessive stippling.

"Belgium" is a highly-tessellated architectural detail. This is a challenging case for our algorithms because it generates occlusion polygons of only a few pixels in area, which are inefficient on a GPU. AOV is able to reproduce the details at different scales, including the braids on the central figure, window mullions, and masonry gaps. The Volumetric algorithm's result is also very good and is substantially faster because it is independent of scene complexity.

The scene in row 5 (shown lit in figure 1) was extracted from the "Secret War" level of Xbox 360 game *Marvel Ultimate Alliance 2*, where it appeared without AO. Secret War produces 1.5M occlusion volumes, but only a quarter pass the frustum and depth test and AOV renders it in 31 ms. With sparse sampling we can drive the time as low as 4 ms while maintaining low error.

6.2 Quantitative Results

We quantify error with perceptually-motivated variance metric,

$$\text{“E[AO]} = \sigma^2 = \text{logMSE}[1-\text{AO}] + \text{logMSE}[\vec{\nabla}(1-\text{AO})]\text{.”} \quad (30)$$

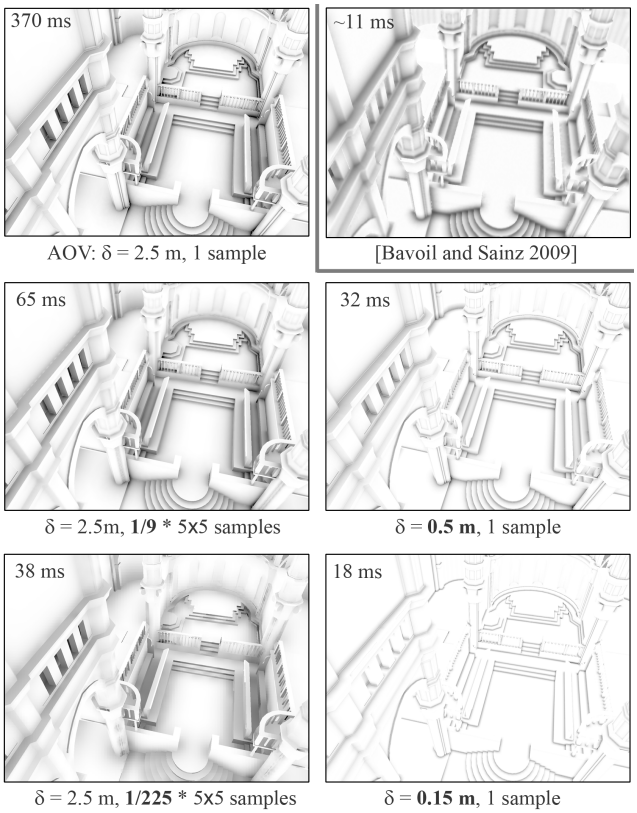


Figure 8: Ambient occlusion for Sibenik cathedral computed by AOV at 1600×1200 with varying δ and sparse sampling. Upper right: Horizon AO result at a sampling rate of 16/4 * 15×15.

Method, samples \ Scene		Belgium *Architecture* 687054 tris		City *Alpha* 9,624 tris		House *Architecture* 28,866 tris		Trees *Foliage* 148,101 tris		Secret War *Video Game* 1,445,620 tris		Sponza *Architecture* 199,362 tris		Suburb *Worst Case* 2,688 tris	
		Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)	Time (ms)	Error (σ^2)
Ray Trace	5000	490803.0	0.00	603202.0	0.00	283226.0	0.00	691334.0	0.00	556228.0	0.00	642571.0	0.00	401222.0	0.00
	1941	190687.0	0.35	234515.0	0.65	109923.0	0.23	270322.0	0.28	216142.0	0.89	249890.0	0.45	155504.0	1.05
	292	28779.9	1.02	35372.4	1.47	16574.2	0.60	40803.0	0.80	32564.8	2.07	37627.1	1.12	23439.2	2.19
	1	156.4	6.83	167.9	6.06	87.7	2.88	195.0	6.06	161.1	9.38	180.9	3.50	130.7	8.98
AOV (new)	1	77.7	0.72	137.3	0.46	25.9	0.51	100.3	1.59	31.18	0.43221	110.1	0.34	31.7	1.53
	1/9 * 5x5	40.3	1.13	19.4	0.52	6.2	0.69	37.0	1.74	20.19	0.72643	31.1	0.61	4.2	1.48
	1/25 * 5x5	27.1	1.55	7.9	0.72	4.5	0.78	27.6	2.10	17.9	0.87595	20.8	0.72	1.8	1.40
	1/225 * 5x5	10.9	2.28	1.9	1.10	2.0	0.90	11.7	4.01	3.45	1.16066	9.1	0.88	0.4	1.08
Volumetric	1024	895.4	2.19	1035.3	3.96	473.7	1.20	742.8	4.32	954.8	1.62	967.9	1.49	1050.1	1.34
	256	224.3	2.50	259.3	4.75	119.0	1.47	186.8	4.98	252.1	2.55	242.9	2.03	265.2	2.28
	32	29.3	4.11	33.6	7.03	15.6	2.42	24.4	6.35	30.9765	4.7377	238.6	2.39	34.4	4.29
	1	3.1	6.65	3.2	12.89	1.7	4.58	2.4	12.26	3.1	9.94	3.1	8.93	3.2	11.36
Crytek	16 * 4x4	15.6	4.34	15.6	3.82	12.8	1.68	14.3	2.98	15.6	2.85	15.7	2.81	15.5	2.76

Color Key: Fast 10ms 33ms 100ms > 200ms ... Slow

Exact Inaccurate

Figure 9: Representative results for selected AO approximation algorithms at 1280×720 on varying scenes and sampling rates. For each trial we report the AO render time in milliseconds and a measure of perceptual error (as 8-bit variance; see eqn. 31). Darker color coding is better (i.e., lower numbers). The new AOV algorithm balances quality and performance, so its rows are heavily shaded.

Formally, let the error in $T = 1 - \text{AO}$; $0 < T_{x,y} < 255$, an 8-bit "test" accessibility approximation, be the the mean squared error (i.e., variance) across the log-mean and log-gradient, compared to a ray traced "reference" R computed from 5000 samples per pixel:

$$\text{E}[T] = \text{MSE}[\log^\dagger T] + \frac{1}{2}\left(\text{MSE}\left[\log^\dagger \frac{\partial T}{\partial x}\right] + \text{MSE}\left[\log^\dagger \frac{\partial T}{\partial y}\right]\right) \tag{31}$$

where $\log^\dagger$ preserves signs and avoids the singularity at $\log 0$:

$$\log^\dagger T = \text{sign}(T) \cdot \log(|T| + 1) \tag{32}$$

Reflected ambient radiance is linear in accessibility, so like the human visual system, this metric tracks both to radiance and changes in radiance with decreasing marginal sensitivity.

Figure 9 reports render time and error for multiple trials varying the sampling parameter for each algorithm. Crytek and sparse-AOV both use post-filtering, so the table lists both the amortized of samples per pixel and the filter kernel size for them.Low, i.e., good, time and error values are colored dark in the table to make trends visible. The data is arranged so that rows near the top are for the highest quality results and ones near the bottom are the fastest. AOV provides a mixture of quality and performance that is close to ray tracing while maintaining interactive rates. With sparse sampling its performance is competitive with screen space methods. For these times we assumed that the geometry was static and pre-computed the occlusion volumes, which is a common shadow volume optimization. Computing fully dynamic volumes in a geometry shader adds about 25% overhead to the performance numbers, largely because it reduces the number of computation units available for pixel shading. Note that static and dynamic volumes can be mixed (again, as with shadow volumes), and that static geometry will correctly occlude dynamic objects and vice versa.

In practice, we observe that performance of the AOV algorithm is primarily gated by overdraw ("fill rate"). Thus the results are only weakly dependent on polygon count. For example, the high-polygon Secret War scene renders three times faster than the Trees scene, even though the trees scene contains one tenth as much geometry. There is a lot of empty space in the outdoor Trees scene and massive overdraw from the overlapping occlusion volumes of individual branches. For the indoor, industrial Secret War scene there is little overlap between volumes and many simply fail the early-out depth test because they are behind walls.

Figure 10 demonstrates the tradeoff between time and error and the convergence rate of different algorithms for a single scene. At 15×15 subsampling, AOV renders Sponza at 100 fps with quality comparable to a 1 minute per frame ray traced rendering; both have the same mean, but the ray tracer is very noisy and AOV misses the high frequencies. Moving towards full-resolution decreases the AOV frame rate but recovers the high frequencies. Volumetric AO at comparable performance to AOV has high error. It improves rapidly, but asymptotically converges to an incorrect result an never matches AOV quality. The Crytek algorithm is not intended for variable sample counts, but we extend its quality level line to show where it intersects the other algorithms.

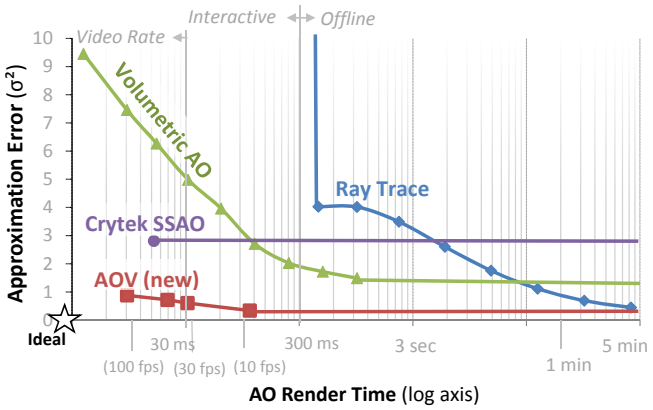


Figure 10: Time vs. Error tradeoff for several algorithms on the Sponza scene at 1280×720. Closer to the lower-left is better. After 60 ms, AOV quality is comparable to a 5 min ray traced result.

7 Discussion

Previous screen space methods remain a good choice for current generation games because their render times are both short and scene-independent. Our AOV algorithm is a new alternative for applications where quality and performance are both important, and it can trade between them by sparse sampling.

Our analytic cosine-weighted solid angle solution is applicable

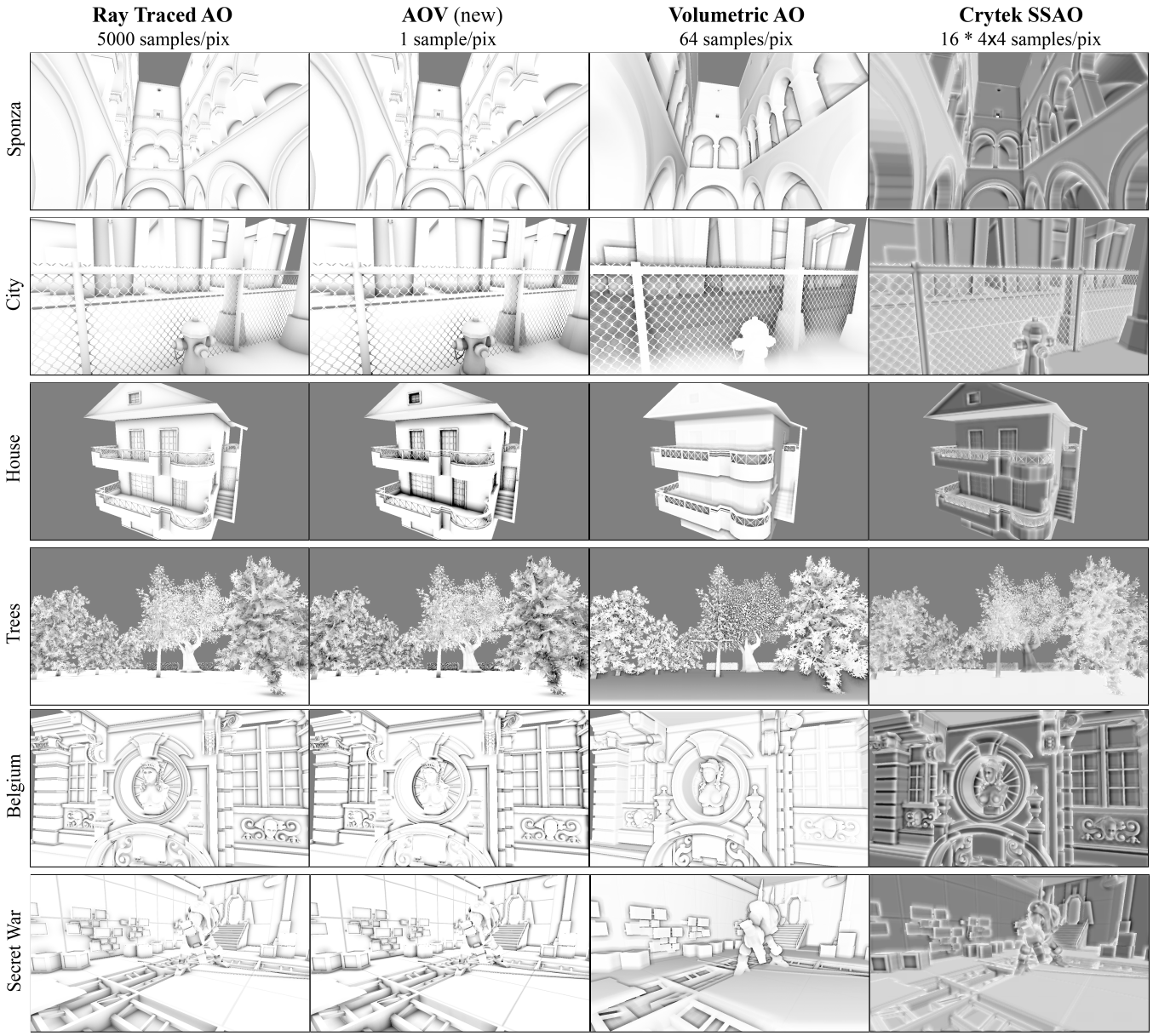


Figure 11: Selected qualitative results for several scenes and algorithms. We treat the left-most column as the reference solution.

beyond ambient occlusion. This integral arises in any problem where flux is incident on a surface. It appears in most illumination computations in computer graphics, and also outside computer science in biology, physics, astronomy, and engineering problems.

Acknowledgements

Thanks to NVIDIA for donating the GPUs used in these experiments, Chris Wassum and Vicarious Visions for modeling and granting permission to use Secret War and the Android character, Max McGuire (Unknown Worlds) for helping with the other models, Marko Dabrovic for modeling Sponza and Sibenik, and Tom Garrity for discussing the projected solid angle geometry.

References

AKENINE-MÖLLER, T., HAINES, E., AND HOFFMAN, N. 2008. *Real-Time Rendering 3rd Edition*. A. K. Peters, Ltd., Natick, MA, USA.

BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. 1989. Improving radiosity solutions through the use of analytically determined form-factors. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 325–334.

BAVOIL, L., AND SAINZ, M. 2009. Multi-layer dual-resolution screen-space ambient occlusion. In *SIGGRAPH 2009: Talks*, ACM, New York, NY, USA, 1–1.

BAVOIL, L., CALLAHAN, S. P., LEFOHN, A., COMBA, JO A. L. D., AND SILVA, C. T. 2007. Multi-fragment effects on the gpu using the k-buffer. In *Proceedings of SI3D 2007*, ACM, New York, NY, USA, 97–104.

BUNNELL, M. 2005. *Dynamic ambient occlusion and indirect lighting*. Addison-Wesley Professional, 223–233.

CROW, F. C. 1977. Shadow algorithms for computer graphics. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 242–248.

EISEMANN, E., AND DURAND, F. 2004. Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3, 673–678.

EVANS, A. 2006. Fast approximations for global illumination on dynamic scenes. In *ACM SIGGRAPH 2006 Courses*, ACM, New York, NY, USA, 153–171.

FILION, D., AND MCNAUGHTON, R. 2008. Starcraft II effects & techniques. In *Advances in real-time rendering in 3D graphics and games course notes*, N. Tatarchuk, Ed. August.

HEGEMAN, K., PREMOŽE, S., ASHIKHMIN, M., AND DRETTAKIS, G. 2006. Approximate ambient occlusion for trees. In *Proceedings of SI3D 2006*, ACM, New York, NY, USA, 87–92.

KIRCHER, S., AND LAWRANCE, A. 2009. Inferred lighting: fast dynamic lighting and shadows for opaque and translucent objects. In *Proc. of the 2009 Symposium on Video Games*, ACM, New York, NY, USA, 39–45.

KONTKANEN, J., AND LAINE, S. 2005. Ambient occlusion fields. In *Proceedings of SI3D 2005*, ACM Press, 41–48.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics 25*, 3 (jul), 1206–1213.

MITTRING, M. 2007. Finding next gen: Cryengine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, ACM, New York, NY, USA, 97–121.

PETSCHNIGG, G., SZELISKI, R., AGRAWALA, M., COHEN, M., HOPPE, H., AND TOYAMA, K. 2004. Digital photography with flash and no-flash image pairs. In *ACM SIGGRAPH 2004 Papers*, ACM, New York, NY, USA, 664–672.

REINBOTHE, C., BOUBEKEUR, T., AND ALEXA, M. 2009. Hybrid ambient occlusion. *EUROGRAPHICS 2009 Areas Papers*, 1–8.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 977–986.

RITSCHEL, T., GROSCH, T., AND SEIDEL, H.-P. 2009. Approximating dynamic global illumination in image space. In *Proceedings of SI3D 2009*, ACM, New York, NY, USA, 75–82.

SCHEUERMANN, T., AND ISIDORO, J., 2005. Cubemap filtering with cubemapgen. GDC Talk.

SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of SI3D 2007*, ACM, New York, NY, USA, 73–80.

SLOAN, P.-P., GOVINDARAJU, N. K., NOWROUZEZAHRAI, D., AND SNYDER, J. 2007. Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of Pacific Graphics 2007*, IEEE Computer Society, Washington, DC, USA, 97–105.

SZIRMAY-KALOS, L., UMENHOFFER, T., TTH, B., SZCSI, L., AND CASASAYAS, M. 2009. Volumetric ambient occlusion. *IEEE Computer Graphics and Applications*. Preprint—to appear.

ZHUKOV, S., INOES, A., AND KRONIN, G. 1998. An ambient light illumination model. In *Rendering Techniques '98*, Springer-Verlag Wien New York, G. Drettakis and N. Max, Eds., Eurographics, 45–56.