High Performance Graphics (2010)
M. Doggett, S. Laine, and W. Hunt (Editors)

Ambient Occlusion Volumes

M. McGuire

NVIDIA and Williams College

Abstract

This paper introduces a new approximation algorithm for the near-field ambient occlusion problem. It combines known pieces in a new way to achieve substantially improved quality over fast methods and substantially improved performance compared to accurate methods. Intuitively, it computes the analog of a shadow volume for ambient light around each polygon, and then applies a tunable occlusion function within the region it encloses. The algorithm operates on dynamic triangle meshes and produces output that is comparable to ray traced occlusion for many scenes. The algorithm's performance on modern GPUs is largely independent of geometric complexity and is dominated by fill rate, as is the case with most deferred shading algorithms.

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. Artists have long recognized AO, and specifically seek to reproduce the real phenomena such as corner darkening and contact shadows. Ambient occlusion can also be quantified by specific terms in the integral equation for light transport (see section 2), which gives a basis for evaluating the error in AO approximation algorithms. As elaborated in that section, it is common practice to compute a specific variant called obscurance or near-field occlusion, in which the effective occlusion distance of each surface is limited to some small artist-selected distance, e.g., $\delta = 1$m for human-scale scenes. This is desirable because it allows a combination of local and global illumination, whether from a precomputed ambient or environment map term or truly dynamic, without double-counting sources; increases the performance of AO algorithms; and allows artistic control over the impact of occlusion.

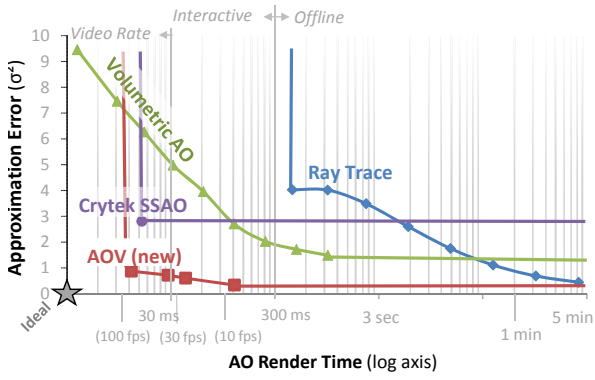
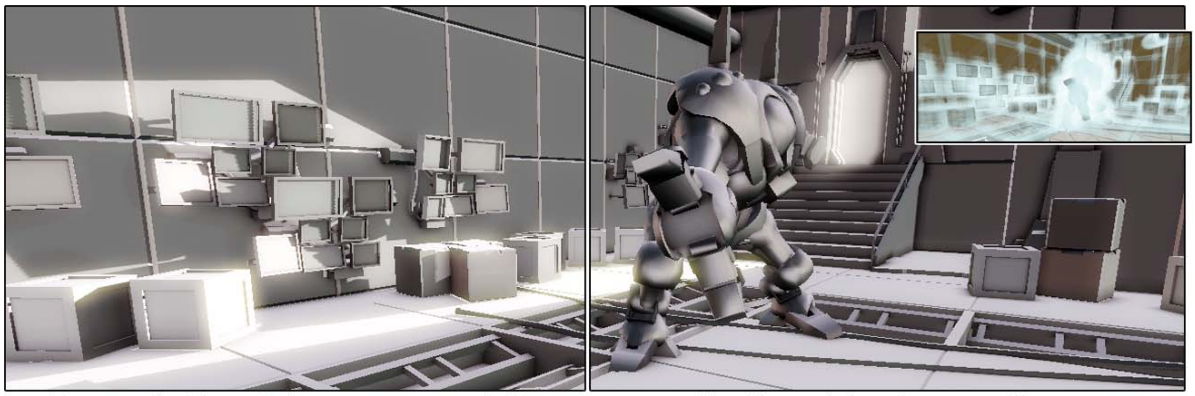


Figure 1: *Time vs. Error tradeoff for several algorithms on the Sponza scene at 1280×720. A 60-ms AOV render has comparable quality to a 5-min ray traced result.*

The primary contribution of this paper is an efficient new algorithm called **Ambient Occlusion Volume** rendering (AOV) for estimating the ambient occlusion based on an analytic solution to the occlusion integral. It is conceptually an extension of Kontkanen and Laine's precomputed ambient occlusion fields for objects [KL05] to individual, dynamic triangles in a mesh, using techniques borrowed from shadow volumes [Cro77]. AOV rendering is viewer independent and produces no noise or aliasing (beyond that of rasterization itself). Its main limitations are that it requires tessellating curved surfaces into meshes and it over-counts occlusion where thin objects are stacked.

Previous ambient occlusion algorithms tend to be fast or good, but not both simultaneously. AOV aims to achieve both reasonable quality and reasonable performance (figure 1) rather than being too geared toward either dimension. It maintains image quality near that of ray tracing, but provides a suitably efficient approximation to enable interaction in modeling and visualization environments.

DOI: 10.2312/EGGH/HPG10/047-056



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

Figure 2: *For this 1.4M-triangle scene at 1280×720 resolution, Ambient Occlusion Volume results have quality comparable to ray tracing 1200 visibility samples per pixel but run in real time: a 4000× speedup. Inset: occlusion volume wireframes.*

2. Ambient Occlusion Problem Statement

This section formalizes ambient occlusion in the context of physically-based rendering. The formal definition motivates our later choice of ray tracing as a "correct" solution for this term when comparing algorithms. The integral equation for exitant radiance at position $\vec{x}$ in direction $\hat{\omega}_o$ is

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

where function f describes how incident light scatters at a surface and L_e represents emitted light. By convention, all vectors point away from $\vec{x}$.

Ambient illumination is the light incident on an imaginary sphere of radius δ about $\vec{x}$ due to objects outside that sphere. It can be computed for every frame time and point with a global illumination algorithm, as is common for offline rendering. Alternatively, a temporally and spatially invariant approximation can be precomputed, as is more common in real-time rendering. Precomputed illumination is often encoded as one of: a constant, a function on the sphere represented in a spherical harmonic basis, or a cube map. Because light superimposes, ambient illumination can also be separated into dynamic and precomputed terms.

Let $L_a(\vec{x},\hat{\omega}_i) = L_i(\vec{x}+\hat{\omega}_i\delta,\hat{\omega}_i)$ represent the ambient illumination. Observe that it may not actually reach $\vec{x}$ if there is an occluder inside the imaginary sphere. Let visibility function $\mathcal{V}(\hat{\omega}_i) = 1$ if there is unoccluded line of sight between $\vec{x}$ and $\vec{x}+\hat{\omega}_i\delta$ and 0 otherwise. A common approximation of the ambient contribution to eq. 1 is (restricting to the positive hemisphere about $\hat{n}$ and omitting function arguments to reveal the mathematical structure):

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

This is only an approximation because multiplication and integration do not commute, except for constants. That is, this approximation is only exact when distant light L_a is independent of direction and f is Lambertian (which explains why Phong's ambient term is a constant function of the diffuse reflectivity only). However, eq. 2 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 2 can be precomputed; for real-time rendering the result is typically encoded in a MIP-mapped cube map [SI05] for negligible run-time cost. This is how I lit the scene in figure 2. Note the repeated $(\hat{\omega}_i \cdot \hat{n})$ factor. This is necessary on both sides to diminish the off-normal light and visibility. The $1/\pi$ compensates for the repeated term and normalizes the right factor in eq. 2, which 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**:

$$\mathrm{AO} = \frac{1}{\pi}\int_{\mathbf{S}^2_+}(1-\mathcal{V})\cdot(\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i = 1 - \frac{1}{\pi}\int_{\mathbf{S}^2_+}\mathcal{V}\cdot(\hat{\omega}_i \cdot \hat{n})\, d\hat{\omega}_i \quad (3)$$

A hard cutoff at δ would reveal the transition between 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 $1-\mathcal{V}(\vec{x},\vec{y})$ with fractional **obscurance** [ZIK98] $(1-\mathcal{V}(\vec{x},\vec{y}))g(\vec{x},\vec{y})$, where **falloff function** g is smooth, monotonic, and is 0 for $||\vec{x}-\vec{y}|| \geq \delta$ and 1 at $\vec{x} = \vec{y}$.

Arikan et al. [AFO05] nicely formalized this decomposition into near vs. far illumination; this is now common in rendering and the remainder of this paper takes that division as given. I also assume that limiting occlusion falloff to δ is a desirable aspect and not a limitation. This is supported by the fact that far-field occlusion is handled in the far-field simulation (beyond the scope of this work) which produces the ambient illumination, and because any enclosed indoor scene would undesirably be *completely* occluded, and therefore have zero ambient illumination, were the far-field occlusion considered in the near field [SA07].

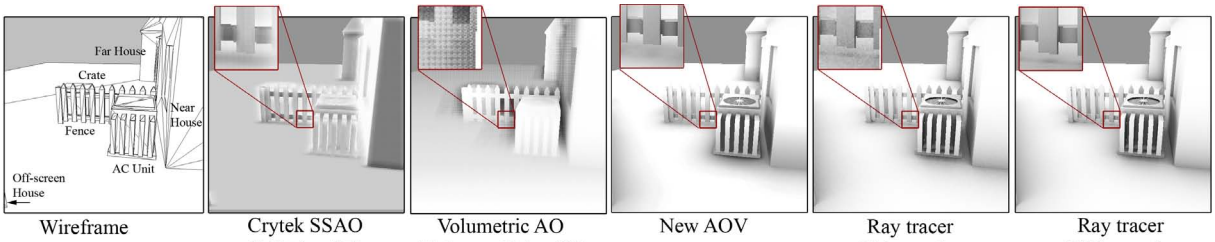


Figure 3: *This "Suburb" stress-test scene contains close proximity between surfaces, varying depth discontinuities, large off-screen occluders, and steep screen-space slopes. Various algorithms exhibit aliasing, noise, and over-occlusion compared to the far right ray traced reference.*

3. Related Work

The basic idea of the AOV algorithm is that an **ambient occlusion volume** is the analog for ambient light of Crow's **shadow volume** [Cro77] for a point source. Zhukov et al. [ZIK98] introduced ambient occlusion and obscurance in the context of the radiosity algorithm. They derived an analytic approximation to the form factor between points on differential patches and apply this to occlusion. I apply a similar approach to whole polygons that is closer to the analytic polygon form factors introduced by Baum et al. [BRW89].

Many previous analytic algorithms approximate occluders as spheres [Bun05, HPAD06, RWS*06, SA07, SGNS07]. AOV follows other work [BRW89, AFO05, SBSO09] in directly solving for a mesh's analytic occlusion, but is the first to do so in real time due to algorithmic improvements.

It is common to compute AO results at low spatial [Mit07, Kaj09, BS09, FM08] or angular [BS09, RBA09] resolution and then joint-bilateral upsample to full resolution. The intuition behind this is that AO is smooth across a plane, and therefore often smooth in screen space as well. Stochastic sampling AO methods produce substantial noise, which they rely on upsampling smooth. AOV's analytic solution is already noise-free and upsampling introduces error, so I consider upsampling an optional step and apply it only where specifically denoted in the results section.

Bunnell [Bun05] 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. Hoberock and Jia [HJ07] extend Bunnell's algorithm to deferred shading per-pixel computations and a disk hierarchy with true polygon occluders at the leaves following Baum et al.'s form-factor computation. Both methods require multiple passes to converge and tend to over-estimate occlusion because the disks are larger than the actual scene polygons they approximate. Shopf et al.'s [SBSO09] method is the starting point for our own. They extend Hoberock and Jia's analytic deferred-shading method to a single-pass screen space method by rasterizing occlusion bounding cubes. This enables real-time performance but introduces double-occlusion. They demonstrate a result on spheres and derive the quadrilaterals case. Beginning with their ideas, we extend the algorithm with tight dynamic bounding volumes, partial coverage, and bilateral upsampling; resolve the occlusion over-estimate; and then provide detailed analysis for the polygon case.

Reinbothe et al.'s Hybrid AO [RBA09] traces rays against a voxelized scene and then corrects high-frequency features with a less accurate SSAO pass. Sloan et al.'s image-based global illumination method [SGNS07] generates accurate ambient occlusion and indirect illumination in real-time for small scenes using virtual light probes. Their method uses spheres as proxy occluders and accumulates the illumination in spherical harmonic coefficients. A recent method by Laine and Karras [LK10] extends AOV with bit masks that prevent over counting (at the cost of quantization), tighter bounding volumes, and acceleration via level of detail.

Another branch of the literature pursues the phenomenological characteristics rather than the physics of AO. Hegeman et al. [HPAD06] 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 [LCD06] 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. The Crytek SSAO [Mit07, Kaj09] algorithm adapted unsharp masking for games by sparsely sampling visibility rays against the depth buffer and filtering the result. Subsequent techniques improved SSAO quality at varying performance by: adding distant occluders [SA07], directional occlusion and indirect illumination [RGS09], better filtering and sampling [SA07, FM08, BS09], and better obscurance [SKUT*09]. Evans [Eva06] precomputed voxel signed-distance fields around static meshes by rasterization and then estimated occlusion by convexity. Similarly, Kontkanen and Laine [KL05] and Malmer et al. [MMAH07] directly precomputed AO on a voxel grid and composed results at run time.

4. Analytic Polygon Occlusion

Let X be an infinitesimal patch of a smooth manifold. Without loss of generality, let the centroid of X be at the origin with normal $\hat{n}$. Let P be a polygon with vertices $\{\vec{p}_0, ..., \vec{p}_{k-1}\}$ that lie entirely within the positive half plane $\vec{p} \cdot \hat{n} \geq 0$. The occlusion by P of ambient illumination directed to X from the sphere at infinity is equal to the form factor that describes the diffuse radiative transfer between P and X,

$$\mathrm{AO}_P(\hat{n}) = \frac{1}{2\pi} \sum_{i=0}^{k-1} \cos^{-1}\left(\frac{\vec{p}_i \cdot \vec{p}_j}{||\vec{p}_i||\,||\vec{p}_j||}\right) \hat{n} \cdot \frac{\vec{p}_i \times \vec{p}_j}{||\vec{p}_i \times \vec{p}_j||} \quad (4)$$

where $j = (i+1) \bmod k$. This was first introduced to graphics by Baum et al. [BRW89] in the context of the radiosity algorithm. I implement it with 1 arccosine, 15 multiply-add, and 2 reciprocal square root operations per edge.

5. Ambient Occlusion Volume Algorithm

I now extend the analytic solution in equation 4 for occlusion of one infinitesimal patch by one polygon to an approximation algorithm for the ambient occlusion of all visible points by a set of polygons, using OpenGL terminology.

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

1. Initialize an **accessibility buffer** to 1 at each pixel
2. Disable depth write, enable depth test, and enable depth clamp (`GL_depth_clamp`) to prevent near-plane clipping
3. (**Vertex Shader:**) Transform all scene vertices as if rendering visible surfaces, e.g., apply skinning and modelview transformations
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 $g = g_P(\vec{x})$; discard the fragment if $g \leq 0$
      b. Let P' be P clipped [HJ07] to the positive half space of the tangent plane at $\vec{x}$
      c. Decrement the accessibility at the projection of $\vec{x}$ by $g \cdot \mathrm{AO}_{P'}(\hat{n})$ via saturating subtractive blending
5. Shading: Modulate the ambient illumination L_a (from eq. 2) 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.

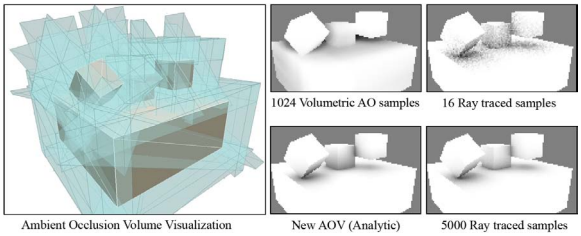


Figure 4: *Left: Ambient occlusion volume visualization. Right Four: Accessibility buffers computed by ray tracing, Volumetric AO, and our new Ambient Occlusion Volumes for a simple scene. The AOV result is indistinguishable from the converged ray traced result for this scene. No matter how many samples are used, Volumetric AO (like other screen space methods) cannot converge to the correct result.*

I found that the R channel of an 8-bit RGB texture had sufficient precision to implement the accessibility buffer and that higher precision had minimal visual impact on most scenes but significantly degraded performance.

A list of quadrilaterals is a good representation for $\{B\}$, however, OpenGL and DirectX can only output triangle strips from a geometry shader. Under current APIs one must either convert the quadrilateral list to triangle strips or construct all B in a separate pass over the scene geometry. Three implementation choices for the latter alternative are an OpenGL transform feedback loop, an OpenCL or CUDA program, and a set of CPU vertex and geometry shaders. Approximately half of the faces in $\{B\}$ are backfaces, which the rasterizer automatically culls. I found no performance advantage in doing so explicitly during face generation.

I implemented both a GPU geometry shader that outputs triangle strips, which is well-suited to dynamic geometry, and a CPU geometry shader that outputs quadrilaterals, which is well-suited to precomputing the volumes for static geometry. For scenes with volumes covering many pixels, precomputation gave up to 20% speedup in our tests (figure 10), although in some cases the bandwidth impact of storing large precomputed streams may actually decrease rendering performance (e.g., the Trees scene). As is the case for shadow volumes, static and dynamic AOV geometry correctly occlude each other–this is an optimization, not an approximation.

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 $\delta \to 0$. For efficiency, I chose:

$$g(\vec{x}) = \bar{\alpha} \prod_{i=0}^{k} \max\left(0, \min\left(1, (\vec{x} - \vec{p}_{i \bmod k}) \cdot \hat{m}_i / \delta + 1\right)\right), \quad (5)$$

where $\bar{\alpha} = 1$ for solid surfaces, $\hat{m}_{i<k}$ are the (inward facing) normals to the edges of P shown in figure 5, and $\hat{m}_k$ is the negative normal to P:

$$\hat{m}_k = \mathrm{S}((\vec{p}_2 - \vec{p}_0) \times (\vec{p}_1 - \vec{p}_0)) \quad (6)$$

$$\hat{m}_{0 \le i < k} = \mathrm{S}\left((\vec{p}_{(i+1) \bmod k} - \vec{p}_i) \times \vec{m}_k\right) \quad (7)$$

as shown in figure 5.

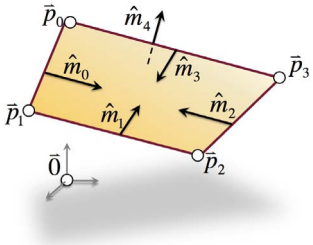


Figure 5: *A polygon P that faces $\vec{0}$, its $k = 4$ vertices $\{\vec{p}_0, ..., \vec{p}_3\}$, inward edge normals $\{\hat{m}_0, ..., \hat{m}_3\}$, and negative face normal $\hat{m}_4$. I depict a quadrilateral occluder in this section to emphasize that the algorithm is defined on convex polygon meshes even though current GPUs are largely restricted to triangle meshes.*

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, B_k be the plane above P, B_{k+1} be the plane that contains P, and $\hat{m}_i$ be the normal to B_i. For a maximum obscurance distance δ, let B_i: $\vec{x} \mid (\vec{x} - \vec{p}_i) \cdot \hat{m}_i = \delta$, where $\vec{p}_k = \vec{p}_0$ and $\vec{p}_{k+1} = \vec{p}_0 + \delta \hat{m}_{k+1}$.

Because I derived occlusion for a point at the origin, g will always be evaluated at $\vec{x} = 0$ but the $\vec{p}_i$ will change. The falloff function from equation 5 thus simplifies to

$$g(\vec{0}) = \bar{\alpha} \prod_{i=0}^{k} \max(0, \min(1, 1 - \vec{p}_{i \bmod k} \cdot \hat{m}_i / \delta)). \quad (8)$$

AOV uses 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 k 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:

$$\vec{e}_i \cdot -\hat{m}_i = \delta \quad (9)$$

$$\vec{e}_i \cdot -\hat{m}_{(i-1+k) \bmod k} = \delta \quad (10)$$

$$\vec{e}_i \cdot \hat{m}_k = 0 \quad (11)$$

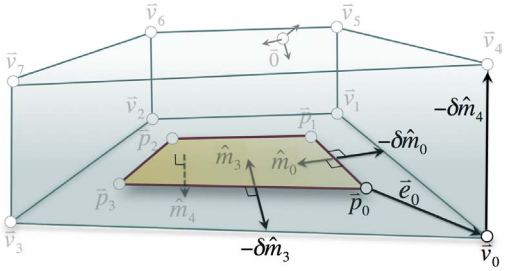


Figure 6: *Diagram of the ambient occlusion volume V cast by a polygon P. The volume's base is formed by extending each original polygon vertex $\vec{p}_i$ along a vector $\vec{e}_i$ derived from the normals $\hat{m}$ of the adjacent edges and maximum obscurance distance δ.*

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

$$\vec{e}_{0 \le i < k} = \begin{bmatrix} \hat{m}_i \\ \hat{m}_{(i-1+k) \bmod k} \\ \hat{m}_k \end{bmatrix}^{-1} \begin{bmatrix} -\delta \\ -\delta \\ 0 \end{bmatrix} \quad (12)$$

$$\vec{e}_{k \le i < 2k} = \vec{e}_{i-k} - \delta \hat{m}_k \quad (13)$$

Sliver polygons create long volumes yet little occlusion, so as a practical measure, 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. I 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.

It is tempting to consider extending the method to use the full α-channel of a texture map to achieve positional variation in occlusion from a single polygon. I have not found an efficient way of doing this without tessellating the polygon. The source of the problem is that the analytic solution assumes opacity is constant over the integration domain, and making it piecewise constant would require multiple integrals...which is equivalent to tessellating the polygon.

5.4. Compensation Map

Overlapping occlusion volumes due to adjacent polygons on the same surface lead to correct results under the AOV algorithm. This is because of the choice of analytic solution and point-based falloff function. This is an important benefit of the algorithm. It means that results are extremely robust to tessellation and level of detail changes.

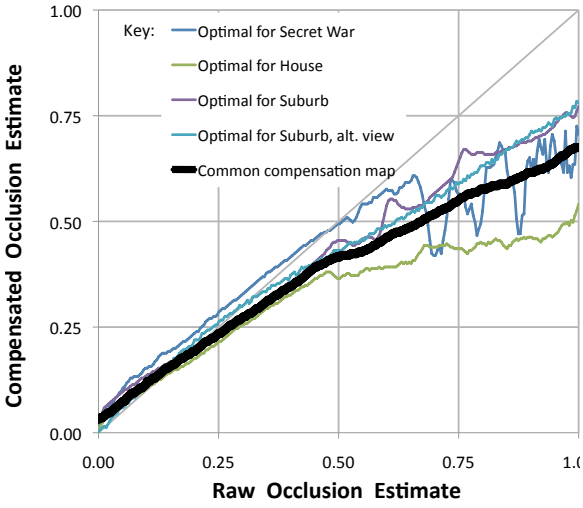


Figure 7: *Thick line: The compensation map curve that mitigates error from artificial multiple occlusion. Thin lines: Over-fit curves for a few representative scenes.*

However, where objects thinner than δ are in proximity closer than δ, they can create artificial multiple occlusion. This is because the occlusion volume of one object passes entirely through another object. This produces overdarkening of the adjacent faces on the objects, which can be observed in figure 8 (left). It is similar to artifacts of shadows passing through walls when casting shadow volumes or rendering shadow maps from only a subset of the scene.

To mitigate this effect, I extend the AOV algorithm to nonlinearly rescale the computed AO of highly occluded regions according to the thick line plotted in figure 7, which can be encoded as a 1D luminance texture on a GPU. The performance impact of the additional texture fetch is below the threshold of measurement, and automatically clamps the input AO to [0, 1]. I created the curve by fitting AOV results from multiple viewpoints from the Suburb and House scenes to ray traced results (I then applied it to all scenes). An intuitive explanation for the right end of the curve is that the corner where three walls meet at a right angle has about 75% occlusion, so that is roughly the maximum occlusion one would expect to encounter in most man-made scenes.

Note that unlike the rest of the algorithm, the compensation map has no physical basis. It is an ad hoc solution to allow rendering of scenes that violate the δ-thickness assumption. The AO for such scenes is strictly over-estimated, and the thin curves demonstrate that it is typically worse in highly occluded areas. Thus, some roughly-parabolic curve is a reasonable correction in many cases.

5.5. Optional Sparse Sampling

Any AO method can be accelerated by reconstruction from sparse samples. AO is an integral over $d\vec{x}$ and $d\hat{\omega}$. If the sparsely sampled dimension is angular ($d\hat{\omega}$) [Mit07, BS09, FM08], the reconstruction filter masks noise. If that dimension is spatial ($d\vec{x}$) [BS09, RBA09], the filter masks aliasing. In both cases, undersampling is a source of error, particularly where geometry changes rapidly in screen space.

The AOV algorithm can trade accuracy for performance by sampling accessibility at low resolution and then reconstructing the full-screen result with a cross-bilateral filter. I chose to reconstruct with a gaussian cross bilateral filter [ED04, PSA*04] that falls off with surface normal discrepancy and 3D distance between the desired and available samples, and implement it as separate 1D horizontal and vertical passes.

Undersampling is a source of error when geometry changes rapidly in screen space. Because the primary benefit of the AOV algorithm is image quality, I did not use sparse sampling in results, except where explicitly noted. In those cases, reconstruction passes account for 1.1-1.3 ms of the total render time at 1280×720. This is independent of the resolution reduction.

6. Results

For all algorithm comparisons, I directly processed indexed triangle meshes. All algorithms were given identical depth and normal 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 a GeForce GT 280 GPU. All images are at 1280×720 ("HD 720p"), and the falloff distance was $\delta = 1.5$m. For figure 2, both sides of the image were lit on the GPU with one shadow-mapped spot light and a cube environment map modulated by (1 - AO), following eq. 2.

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.

I denote sparse sampling for all algorithms considered as the product of amortized samples taken per pixel and reconstruction filter size, which gives the total amortized samples affecting a result pixel. Thus, $1/3^2 \cdot 5^2$ denotes one visibility sample per pixel in 3×3-downsampled buffer, followed by a 5×5 upsampling kernel.

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 I measure the others. The second 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 [Mit07, Kaj09], 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 and Kajalin's published results. The third column is Volumetric AO [SKUT*09], 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. (These artifacts may not be visible in the printed paper; see the inset details in figure 3 or manually zoom in on any result in the electronic version of the paper.)

Note that Crytek SSAO uses no falloff, Volumetric AO uses cubic falloff, and our falloff is the product of many factors. Yet I measure error results against linear falloff in a ray tracer, which a nonlinear algorithm could not possibly match. I 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 AOV does) compromises quality. I compensate by choosing $\delta_{AOV} = 0.6\delta_{ray}$ and $\delta_{volumetric} = 1.1\delta_{ray}$, which minimized their error on test scenes. Note that all of these algorithms enjoy a significant performance benefit from the limited occlusion radius. Screen space methods require $O(\delta^2)$ filter taps for convergence and experience performance cliffs from cache misses when δ grows too large. The ray tracer can terminate tracing at distance δ rather than propagating through the entire tree. AOV consumes $O(\delta^2)$ fill rate.

The first row of figure 11 is the "Sponza" benchmark model. The AOV result is comparable to the ray traced one. 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 (omitting 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 *in the ray tracer* where occlusion rays miss the fence because they are parallel to it within floating point precision.

"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 algorithm 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 2) was extracted from the "Secret War" level of Xbox 360 game *Marvel Ultimate Alliance 2*. 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 one can drive the time as low as 4 ms while maintaining low error.

6.2. Compensation Map

Figure 8 shows the suburb scene as rendered by the AOV algorithm. Recall that this scene was constructed to be difficult to render. AOV correctly produces the soft shadows on the side of the house, the soft shadow from the house outside the viewport on the left, and exhibits no noise at any location. The primary artifact is that, compared to the ray traced result, the AOV result is too dark between the fins of the air conditioning (AC) unit and where the AC unit casts a soft shadow on the nearby wall. This overdarkening arises from overlapping occlusion volumes from the thin fin features. The shadow underneath the fan is also less distinct. That is a separate artifact that arises because the fan grille is modeled as a single α-masked polygon, so AOV sees only a translucent quad instead of detailed grille blades. Modeling the central disk as a separate polygon would correct this. After compensation the local intensity image matches the ray traced reference, largely correcting the overdarkening. Note that some detail has been lost at the bottom of the air conditioner fins where the occlusion saturated. Also note the AC unit's shadow on the wall of the house, which now has correct intensity, and that throughout the image, medium tone regions have been preserved.

6.3. Quantitative Results

6.3.1. Estimation Error

I quantify error with a perceptually-motivated variance metric. Informally, this is

$$\mathrm{E}[\mathrm{AO}] = \sigma^2 = \mathrm{logMSE}[1 - \mathrm{AO}] + \mathrm{logMSE}[\vec{\nabla}(1 - \mathrm{AO})].$$

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

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

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

$$\log^\dagger x = \mathrm{sign}(x) \cdot \log(|x| + 1) \qquad (14)$$

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

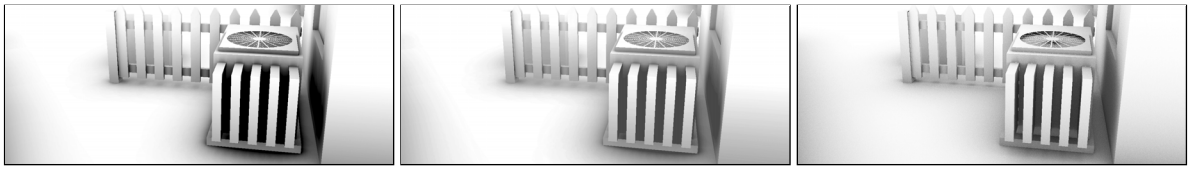


Figure 8: *Left: AOV result before compensation. Middle: AOV result after compensation map. Right: Ray traced 256 spp.*

Method, samples \ Scene		Belgium *Architecture* 687,054 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^4)	Time (ms)	Error (σ^4)	Time (ms)	Error (σ^4)	Time (ms)	Error (σ^4)	Time (ms)	Error (σ^4)	Time (ms)	Error (σ^4)	Time (ms)	Error (σ^4)
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**	0.59	**137.3**	0.46	**25.9**	0.27	**100.3**	2.03	**31.2**	**0.43**	**110.1**	0.28	**31.7**	0.65
	$1/3^2 \cdot 5^2$	41.6	1.03	20.5	0.51	7.2	0.69	38.0	2.40	21.3	0.73	32.2	0.61	5.2	0.68
	$1/5^2 \cdot 5^2$	28.1	1.55	8.9	0.72	5.6	0.78	28.7	2.74	18.9	0.88	21.8	0.72	2.8	0.74
	$1/15^2 \cdot 5^2$	11.9	2.28	2.9	1.10	2.9	0.90	12.8	3.14	4.6	1.16	10.2	0.88	1.4	0.85
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	31.0	4.74	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	$16 \cdot 4^2$	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 I report the AO render time in milliseconds and a measure of perceptual error (as 8-bit variance; see eqn. 14). Darker color coding is better (i.e., lower numbers). The new AOV algorithm balances quality and performance, so its rows are heavily shaded.*

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 samples per pixel and the filter kernel size for them. For this test, I assumed static scene geometry and pre-computed the occlusion volumes for AOV and bounding volume hierarchy for the ray tracer. See figure 10 for a comparison of best- (fully static) and worst- (fully dynamic) case render times for AOV on these scenes.

To make trends visible in figure 9, the low (i.e., good) time and error values are colored dark. The data is arranged so that rows near the top are high quality and ones near the bottom are fast. The gray-outlined rows for Ray Trace with 1941 samples and full-quality AOV demonstrate the success of AOV for the target application: upgrading offline ray tracing to interactive AOV rendering in a modeling program or similar application, while retaining the character of the illumination. Pushing the algorithm farther, with sparse screen-space sampling AOV performance is competitive with screen space methods, although sparse sampling degrades image quality.

6.3.2. Execution Time

In practice, I observe that performance of the AOV algorithm is primarily governed by the amount of occlusion volume overdraw ("fill rate") as measured by `GL_OCCLUSION_QUERY` and largely independent of scene polygon count, as shown in figure 10. This holds even for fully dynamic scenes in which every volume is computed on the GPU every frame. Note the outlier of the dynamic render time for the Belgium scene, which has neither high geometric complexity nor fill consumption. I attribute this to small screen-space triangles in that scene, which either starve the geometry processor of resources or cause it to block on the rasterizer. The average volume in that scene covers only two pixels, so although little fill rate is consumed, there is significant edge processing per pixel rendered. In the starvation scenario, rasterization is inefficient because most threads in a pixel shader warp are idle because they lie outside the triangle. Those threads are still devoted to the pixel processor and therefore starve the geometry processor of resources. In the blocking scenario, the rasterizer's input queue blocks because of the overhead of triangle setup for tiny triangles.

Figure 1 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 and never

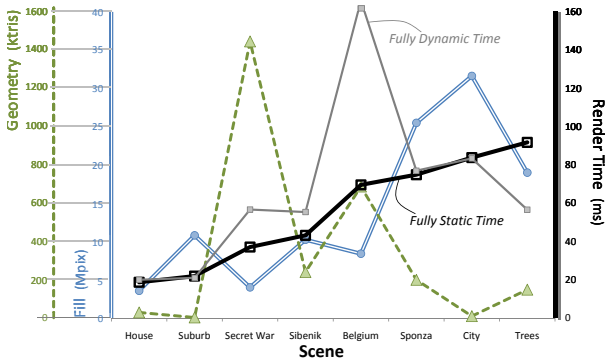


Figure 10: *Scene geometry, AOV fill consumption, and AOV render time for various scenes. The thick dark line denotes pre-processed static volumes, the thin gray line represents a fully dynamic scene with volumes computed in a GPU geometry shader. This is a bar graph rendered with lines to emphasize correlations. Render time is strongly correlated with fill and largely independent of scene complexity.*

matches AOV quality. The Crytek algorithm is not intended for variable sample counts, but I extend its quality level line to show where it intersects the other algorithms. Volumetric is well-suited to the game domain because it can achieve very high performance. AOV's substantially higher quality is preferable for non-game applications where ray tracing would be considered.

Acknowledgements

Thanks to Chris Wassum and Vicarious Visions for the Secret War and the Android character, Max McGuire (Unknown Worlds) for helping with other models, Marko Dabrovic for Sponza and Sibenik, Tom Garrity (Williams), Corey Taylor (EA), and Joakim Carlsson and Patrik Sjölin (Chalmers) for their advice and corrections. Thanks to Pete Shirley for his substantial help editing the final draft of the paper.

References

[AFO05] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Fast and detailed approximate global illumination by irradiance decomposition. *ACM Trans. Graph. 24*, 3 (2005), 1108–1114.

[BRW89] BAUM D. R., RUSHMEIER H. E., WINGET J. M.: Improving radiosity solutions through the use of analytically determined form-factors. In *Proceedings of SIGGRAPH '89* (New York, NY, USA, 1989), ACM, pp. 325–334.

[BS09] BAVOIL L., SAINZ M.: Multi-layer dual-resolution screen-space ambient occlusion. In *SIGGRAPH 2009: Talks* (New York, NY, USA, 2009), ACM, pp. 1–1.

[Bun05] BUNNELL M.: *Dynamic ambient occlusion and indirect lighting*. Addison-Wesley Professional, 2005, pp. 223–233.

[Cro77] CROW F. C.: Shadow algorithms for computer graphics. In *Proceedings of SIGGRAPH '77* (New York, NY, USA, 1977), ACM, pp. 242–248.

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Trans. Graph. 23*, 3 (2004).

[Eva06] EVANS A.: Fast approximations for global illumination on dynamic scenes. In *ACM SIGGRAPH 2006 Courses* (New York, NY, USA, 2006), ACM, pp. 153–171.

[FM08] FILION D., MCNAUGHTON R.: Starcraft II effects & techniques. In *Advances in real-time rendering in 3D graphics and games course notes*, Tatarchuk N., (Ed.). August 2008.

[HJ07] HOBEROCK J., JIA Y.: *High-Quality Ambient Occlusion*. Addison-Wesley Professional, 2007, ch. 12.

[HPAD06] HEGEMAN K., PREMOŽE S., ASHIKHMIN M., DRETTAKIS G.: Approximate ambient occlusion for trees. In *Proceedings of SI3D 2006* (New York, NY, USA, 2006), ACM.

[Kaj09] KAJALIN V.: *ShaderX*^{7}. Charles River Media, March 2009, ch. Screen Space Ambient Occlusion, pp. 413–424.

[KL05] KONTKANEN J., LAINE S.: Ambient occlusion fields. In *Proceedings of SI3D 2005* (2005), ACM Press, pp. 41–48.

[LCD06] LUFT T., COLDITZ C., DEUSSEN O.: Image enhancement by unsharp masking the depth buffer. *ACM Transactions on Graphics 25*, 3 (jul 2006), 1206–1213.

[LK10] LAINE S., KARRAS T.: Two methods for fast ray-casted ambient occlusion. In *EGSR 2010* (June 2010).

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, pp. 97–121.

[MMAH07] MALMER M., MALMER F., ASSARSSON U., HOLZSCHUCH N.: Fast precomputed ambient occlusion for proximity shadows. *journal of graphics, gpu, and game tools 12*, 2 (2007), 59–71.

[PSA*04] PETSCHNIGG G., SZELISKI R., AGRAWALA M., COHEN M., HOPPE H., TOYAMA K.: Digital photography with flash and no-flash image pairs. In *ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 664–672.

[RBA09] REINBOTHE C., BOUBEKEUR T., ALEXA M.: Hybrid ambient occlusion. *EUROGRAPHICS 2009Areaspapers*(2009).

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *Proceedings of SI3D 2009* (New York, NY, USA, 2009), ACM.

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 977–986.

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of SI3D 2007* (New York, NY, USA, 2007), ACM, pp. 73–80.

[SBSO09] SHOPF J., BARCZAK J., SCHEUERMANN T., OAT C.: deferred occlusion from analytic surfaces. In *ShaderX*^{7}, Engel W., (Ed.). 2009, pp. 445–454.

[SGNS07] SLOAN P.-P., GOVINDARAJU N. K., NOWROUZEZAHRAI D., SNYDER J.: Image-based proxy accumulation for real-time soft global illumination. In *Proceedings of Pacific Graphics 2007* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 97–105.

[SI05] SCHEUERMANN T., ISIDORO J.: Cubemap filtering with cubemapgen, 2005. GDC Talk.

[SKUT*09] SZIRMAY-KALOS L., UMENHOFFER T., TŰTH B., SZŐCSI L., CASASAYAS M.: Volumetric ambient occlusion. *IEEE Computer Graphics and Applications* (2009).

[ZIK98] ZHUKOV S., INOES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques '98* (1998), Drettakis G., Max N., (Eds.), Eurographics, Springer-Verlag Wien New York, pp. 45–56.

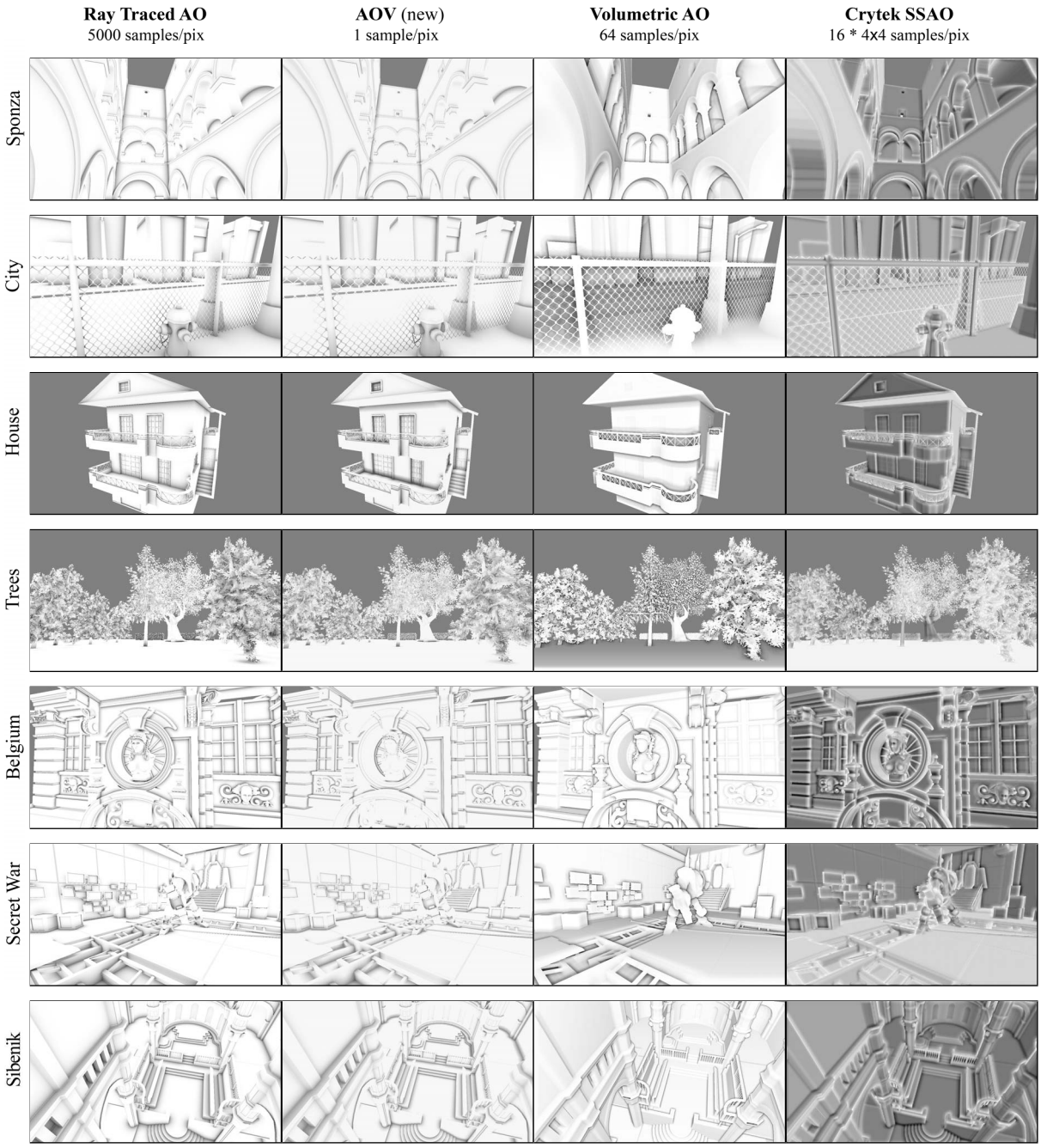


Figure 11: *Selected qualitative results for several scenes and algorithms (at 1000 dpi for zooming in the electronic version of this paper). I treat the left-most column as the reference solution.*