Voxel-Space Ambient Occlusion

Rajeev Penmatsa[†] **and** ***Chris Wyman***

University of Iowa
Department of Computer Science
Technical Report UICS-12-01

[†] Department of Computer Science
14 MacLean Hall
The University of Iowa
Iowa City, IA 52242 USA

February 1, 2012

Abstract

Ambient occlusion adds important detail to a scene. This paper presents a multiresolution screen-space voxel based ambient occlusion technique, which improves G-buffer based techniques, avoiding artifacts such as haloing. Our technique is scalable to the level-of-detail desired by the user.

Because we assume voxels to be small objects, we compute ambient occlusion using an area rendering equation based formulation. As ambient occlusion changes smoothly across a scene, we use a stencil based screen-space multiresolution buffer to refine and render multiple resolution images. Curvature and depth discontinuity metrics to identify appropriate sampling rates.

1 Introduction

Ambient occlusion has become integral in most modern real-time renderings, due to the plausible, inexpensive detail it adds. Ambient occlusion provides a cheap approximation of global illumination from an ambient light source, a useful addition to many scenes.

Screen space ambient occlusion uses G-buffer information to compute shading, and cannot compensate for missing surfaces occluded by other objects. This introduces artifacts such as halos around objects. However, G-buffer based techniques enjoy popularity because of their relative simplicity.

Although depth-peeling can address this, choosing robust boundaries between depth layers becomes challenging in dynamic scenes. Recent work by McGuire [McG10] computes ambient occlusion by generating volumes around geometry, however geometry volume complexity increases with primitive count.

A coarse representation (voxelization in our case) of scene geometry can be a substitution for depth for ambient occlusion, and will not suffer from the artifacts associated with screen-space techniques. We use a single pass technique by Eisemann and Décorat [ED08] to voxelize screen-space geometry generating a coarse representation.This technique generates voxels quickly, however, the generated voxels are non-uniform, an issue we discuss in later sections.

2 Background

Several techniques have been developed to display ambient occlusion in real-time. Screen space ambient occlusion (SSAO) was developed by Mittring [Mit07], where the Z-buffer is sampled around a point to compute ambient occlusion by simple depth comparisons. This was a modification of unsharp mask by Luft et al [LCD06]. In a similar manner, Shanmugan and Arikan [SA07] computed ambient occlusion based on sphere approximations of geometry, both near and distant. Nearby occluders' ambient occlusion used the Z-buffer. Similarly, Hegeman et al [HPAD06] and Bunnell [Bun05] used spheres to represent approximate geometry.

Kontkanen and Laine [KL05] precomputed object occlusion contribution of geometry in a voxel grid and combined results during run-time. McGuire [McG10] and

Laine and Karras [LK10] compute ambient occlusion similar to shadow volumes by extruding the polygons and applying an occlusion function with the extrusions.

We use a screen space voxelization proposed by Eisemann and Décorat [ED08]. Reinbothe et al [RBA09] computed ambient occlusion by tracing rays in a voxel grid and filtering the occlusion result in screen space. Nichols et al [NPW10] use voxelization of screen-space geometry to render dynamic area lighting.

We use a modification of the ambient occlusion formulation by McGuire [McG10] in Ambient occlusion volumes, where we derive an area formulation of ambient occlusion from the area rendering equation formulation.

Our multiresolution technique depends on curvature discontinuities.To estimate curvature, we use the technique by Kim et al [KYYL08] to compute curvature in screen space by sampling two neighboring pixels.

We extend the stencil based multi-resolution technique by Nichols et al [NPW10] by using curvature and depth discontinuities, as opposed to normal and depth discontinuites.

3 Voxelization

Due to the low frequency nature of ambient occlusion, a coarse geometry representationl suffices for most computations. We use Eisemann's technique to generate voxels using a 128-bit integer buffer. The results is 128 bins between the near plane and the far plane, where each bin represents a boolean value, value of 1 indicating a presence of geometry, 0 the absence.

Voxels are generated within the viewing frustum resulting in non-uniform voxels. The voxels mimic the shape of the viewing frustum, because we are essentially splitting the frustum into smaller parts. Depending on the near plane and far plane, the voxels are usually 'stretched out' , i.e., voxels have more depth than width or height. Although this does not affect how the screen-space geometry is represented by the voxels, it affects how ambient occlusion is calculated.

We found a single 128-bit integer buffer is insufficient for high-quality ambient occlusion. Therefore, we use four 128-bit integer buffers over the depth range, resulting in 512 bins. The resulting increase in quality is significant, with only a small perfor-

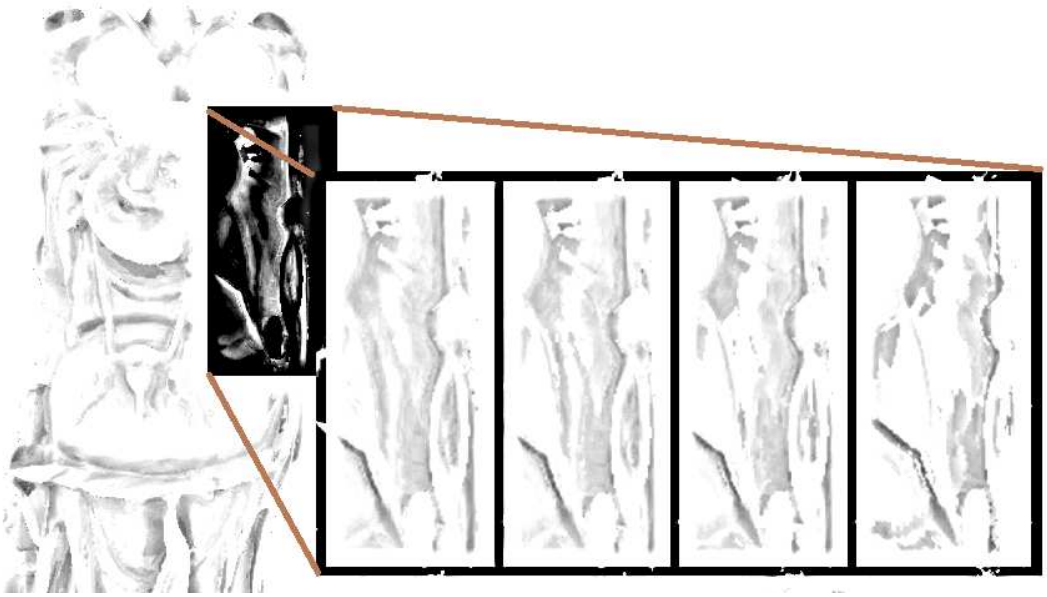

Figure 1: An example of voxel-space ambient occlusion rendered with 512, 384, 256 and 128 bins respectively.

mance hit.

The voxels are larger in the Z-direction than in the X and Y direction, forming a shape similar to the view frustum. Ambient occlusion over a hemisphere due to a voxel is computationally expensive as individual sides of voxels need projection on to the hemisphere to calculate the contribution. We derive a ambient occlusion equation from the area representation of the rendering equation in the following section.

4 Ambient Occlusion Computation

Ambient occlusion shades a point based on the obstruction of light from surrounding geometry. This section dervies an ambient occlusion formulation using the area based form of the rendering equation.

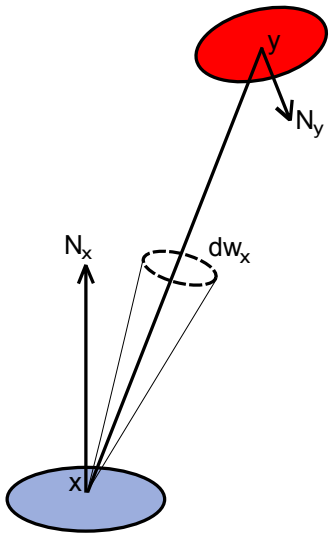

Figure 2: Image illustrating the area based rendering equation.

The area formulation of the rendering equation is given by:

$$L_o(x) = L_e(x) + \int_A f_r(x)\ L(y)\ V(x,\ y)\ \frac{(N_x \cdot \psi)\ (N_y \cdot -\psi)}{r_{xy}^2}\ dA_y \tag{1}$$

Where L_o is the light reflected from the point on the surface. And L_e is the emitted light. f_r represents the BRDF of the surface material, for incoming direction ψ and an outgoing direction. $L(y)$ is the incoming radiance in the direction ψ from a point y. $V(x,\ y)$ is the binary visibility between points x and y. $(N_x \cdot \psi)$ and $(N_y \cdot -\psi)$ represent the amount of radiance reaching point x depending on the angle made by direction ψ with the normals of point x and point y. r_{xy}^2 represents the square of the distance between the points x and y.

Similar to McGuire's work, we let the light arriving from nearby objects be L_n [McG10]. Light arriving from large distant objects (ambient light) be L_a. Relatively small, distance sources be L_s. Let the overall visibility of the object be V_o and the visibility of a point on the object be V_p. We can rewrite equation 1 as:

$$
\begin{aligned}
L_o(x \to \theta) = &\int_A f_r(x,\, \psi \to \theta)\; L_s\; V_o\; V_p\; G(x,\, y)\; dA_y \\
&+ \int_A f_r(x,\, \psi \to \theta)\; L_n\; (1 - V_o)\; V_p\; G(x,\, y)\; dA_y \\
&+ \int_A f_r(x,\, \psi \to \theta)\; L_a\; V_o\; V_p\; G(x,\, y)\; dA_y
\end{aligned} \tag{2}
$$

Where

$$
G(x, y) = \frac{(N_x.\psi)\; (N_y. - \psi)}{r_{xy}^2} \tag{3}
$$

Ambient light reaching a point due to large, distant light sources is:

$$
L = \int_A f_r(x,\, \psi \to \theta)\; L_a\; V_o\; V_p\; G(x,\, y)\; dA_y \tag{4}
$$

We can ignore the visibility of the object, because if a point on the object is visible, the object is also visible. Ambient occlusion can be formulated as the light reaching a point because of disocclusions of nearby objects. That is, light reaching because of objects that are 'invisible' in the vicinity of the point. Therefore, light reaching the point can be written as:

$$
AO = \int_A (1 - V_p)\; G(x,\, y)\; dA_y \tag{5}
$$

Which can be written as

$$
AO = \int_A (1 - V_p)\; \frac{(N_x \cdot \psi)\; (N_y \cdot -\psi)}{r_{xy}^2}\; dA_y \tag{6}
$$

Using monte carlo integration with N samples within the hemisphere, we have:

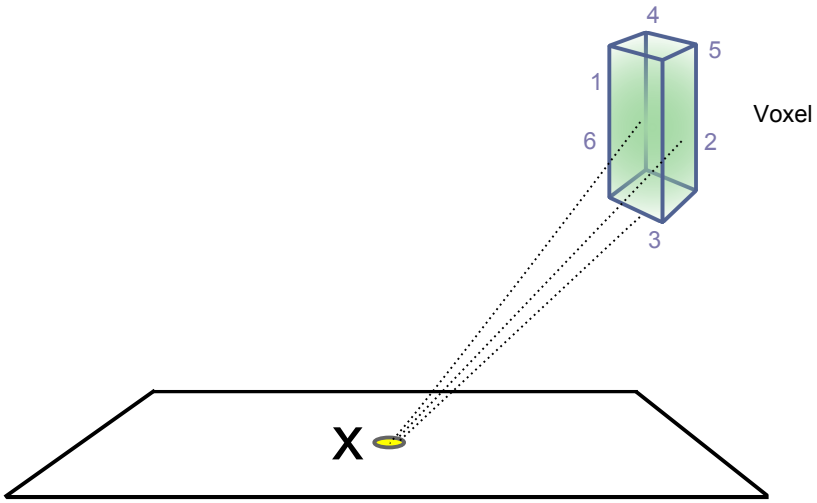

Figure 3: Contribution of a voxel to ambient occlusion at a point.

$$AO = \frac{1}{N} \sum_{i=0}^{N} (1 - V_p) \frac{(N_x \cdot \psi)\ (N_y \cdot -\psi)}{r_{xy}^2}\ dA_y \tag{7}$$

We rewrite this as

$$AO = 1 - \frac{1}{N} \sum_{i=0}^{N} V_p \frac{(N_x \cdot \psi)\ (N_y \cdot -\psi)}{r_{xy}^2}\ dA_y \tag{8}$$

We use the above equation to compute the occlusion due to voxels around a point, the details of which will be discussed in a following section.

5 Ambient Occlusion Contribution due to a Voxel

We assume each voxel has 6 surfaces that contribute to ambient occlusion. We compute the size of each voxel size by taking into account the world space size of each pixel.

The voxel's coordinate system is oriented such that the camera's *up* vector is co-linear with the y-direction of the voxel's coordinate system, and the camera's *right* vector is co-linear with the x-direction of the voxel's coordinate system. This also means that the *at* vector of the camera will be co-linear with the z-direction the voxel's coordinate system. The world size of front and back face of the voxel is obtained by inverse perspective transform. The areas of the sides oriented along the z-axis of the voxel's coordinate system can be computed by dividing the region between the near and the far plane into 512 equal bins.

We compute the contribution due to each of the sides of the voxel to ambient occlusion by using the area formulation of ambient occlusion (Equation 8).

6 Kernel Size and Multiple Steps along a Ray

To make our technique robust for rendering different scenes, we sample a user-defined screen-space radius around the pixel in world-space. We also generate a user-defined number of samples within a cosine-weighted hemisphere, allowing users to choose between performance and quality. These samples are mutliplied by the radius and projected back to screen-space, while nothing the depth at the sample. This lets us query the voxel buffer to check occupancy at the sample location. Sample locations are checked and presence of a voxel results in calculation of ambient occlusion for the voxel using the equation 8 described in section 4 and method described in section 5.

Similar to horizon occlusion mapping, each sample may be multiplied by multiple radii (user defined number of ray-steps) for more accurate ambient occlusion. If multiple-radii sampling is used, i.e., each cosine-weighted hemisphere sample is used multiple times (number of times is set by the user) at different radii, voxels closer to the point take higher precedence. This means that samples at a larger radii are ignored if a voxel is occupied at a smaller radii location for the same sample.

7 Multiresolution Ambient Occlusion

Crevices in geometry experience a higher degree of shading to occlusion than flat surfaces. Higher detail is required in such areas as ambient occlusion may vary significantly from point to point. We take advantage of this fact and use a stencil based

screen-space multi-resolution approach similar to Nichols and Wyman [NW09] and Nichols et al [NPW10], where we selectively render higher detail areas and lower detail areas with different precision. Ambient occlusion varies smoothly across a scene, allowing lower detail areas to be rendered at lower precision.

We compute the curvature and depth continuity and store them in a buffer. This is followed by setting stencil bits in parallel in a multiresolution buffer, based on a set threshold of depth and curvature metrics. The following sections explain the steps in more detail.

7.1 Curvature and Depth as Metrics

It can be noticed that the extent of ambient occlusion occurring within concave sections of geometry is higher than flat sections. Light entering such areas is blocked by surrounding geometry as high curvature is a result of neighboring geometry being higher or lower than our point of interest. We can observe this in the depth buffer as neighboring geometry having a higher or lower depth value. We use this observation to separate areas of high concave curvature from low concave curvature (convex curvature is ignored because contribution to ambient occlusion is same as flat regions). Areas with higher concave curvature are rendered at higher precision than flat or convex areas.

We compute a curvature buffer from screen space position and normals (G-buffer) using the technique by Kim et al.[KYYL08]. This technique requires sampling of three neighboring points (for position and normal) to compute curvature. Because ambient occlusion contribution due to convex and flat surfaces is the same, we modify the curvature buffer to ignore convex curvature. Perfectly flat and surfaces at right angles to each other (i.e., the dot product of one or more pairs of normals is zero) result in inaccuracies due to switching polarities of the dot product, which we rectify by clamping the curvature.

Edges indicate changes in world objects or surfaces, requiring higher detail. We sample depth values from the G-buffer and compute depth discontinuities similar to Nichols and Wyman [NW09] and Nichols et al.[NPW10]. We use the computed curvature buffer and depth buffer for stencil refinement, which is discussed in the following section.

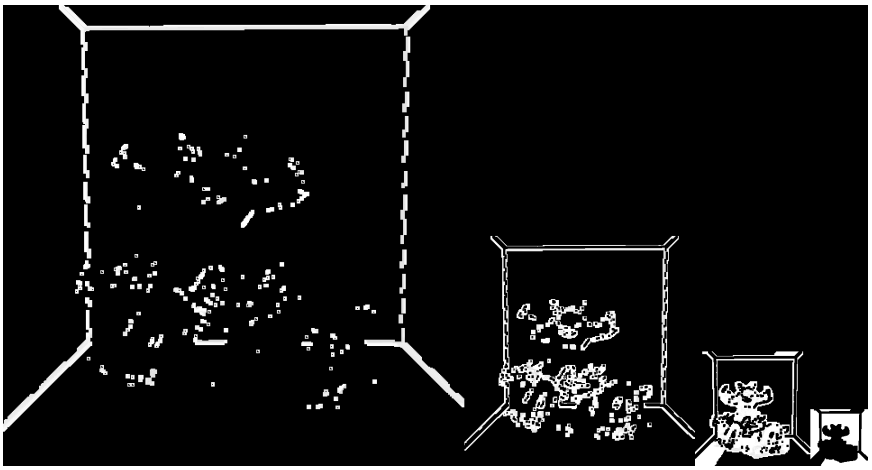

Figure 4: Multiresolution buffer we use for refinement. Multiple resolution mip-maps are stored in a single buffer.

7.2 Multiresolution Ambient Occlusion

We use a stencil based multi-resolution rendering approach similar to Nichols and Wyman[NW09] and Nichols et al [NPW10]. We start by constructing a mipmap structure in a multiresolution buffer. Stencil bits in the multiresolution buffer are set based on curvature and depth discontinuities at a given pixel based on user set curvature and depth discontinuity thresholds. For each layer in the mipmap structure, we check the curvature and depth discontinuity buffer to determine if the values fall within the computed threshold. If so, stencil bits are set in the layer.

Ambient occlusion is incrementally computed based on the level at set stencil bits. At each layer, a set stencil bit triggers ambient occlusion computation and the layers are rendered from coarsest to finest. Upsampling and interpolation of resulting shading is done using a bilinear interpolation similar to [NPW10]. We upsample from the coarsest to the finest level, linearly interpolating with neighbor texels, avoiding unset (in the stencil) neighbors for that level. The result of a coarse layer is also integrated into the next finer layer, producing a smooth transition between layers.

8 Results and Discussion

On a quad-core Intel Core i7 processor running at 2.8Ghz and a nVidia 470GTX video card, we have the following tabulated results. These results are for 36 samples per pixel, with 2 steps along every ray (72 samples in total) rendered at 1024 x 1024 resolution. We choose 2 steps along a ray to mimic ray marching. We also tested higher number of rays steps but found the resulting quality not significant enough to justify the performance decrease.

Step	512 bins	384 bins	256 bins	128 bins
Deferred Shading	2.9ms	2.9ms	2.8ms	2.8ms
Screen Voxelization	6.6ms	4.7ms	3.6ms	3.0ms
Curvature Mipmap	1.4ms	1.4ms	1.4ms	1.4ms
Mipmap refinement	1.1ms	1.1ms	1.1ms	1.1ms
VSAO	13.1ms	12.9ms	12.9ms	12.7ms
UpSample	0.5ms	0.5ms	0.5ms	0.5ms
Final Gather	0.4ms	0.4ms	0.4ms	0.4ms

We use nVidia OptiX to generate our ground truth image, with 36 samples per pixel. The samples are randomly selected in a cosine distributed hemisphere. Figure 4 compares the ground truth against our implementation.

The robustness of our technique lies in the ability to render with either quality or performance. With the ability to tweak the threshold levels for curvature and depth, and the number of samples and rays per sample, our technique can be adopted for both realtime and off-line applications. Our technique also maintains realtime performance even with a large number of samples.

Curvature and depth based multi-resolution rendering can also be adopted for usage with other techniques such as global illumination similar to Nichols et al.[NPW10].

9 Limitations

Our technique cannot be a substitute for ray-traced ambient occlusion, because voxelization represents an approximation of actual geometry. However, the number of

voxels per scene can be increased without a huge loss in performance, thus improving quality.

We use a voxelization technique that fails in cases where the geometry is non-watertight, which results in artifacts with our technique. This usually results in lack of shading in areas because of missing voxels.

Acknowledgments. The authors were supported by a grant from the Army Research Office (#W911NF-10-1-0338). Models come from the Stanford repository.

References

[BS09] L. Bavoil, M.Sainz *Multi-layer dual-resolution screen-space ambient occlusion.* In SIGGRAPH 2009

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

[ED08] E.Eisemann, X. Décorat *Single-pass gpu soldi voxelization for real-time applications* In Proc. Graphics Interface (2008), pp. 73-80

[HPAD06] K.Hegeman, S. Premože, M.Ashikhmin, G.Drettakis *Approximate ambient occlusion for trees*, In Proceedings of SI3D 2006 Nrew York, Ny, USA, 2006, ACM

[KL05] J. Kontkanen, S.Laine *Ambient occlusion fields*, In Proceedings of SI3D 2005, ACM Press, pp. 41-48

[KYYL08] KIM, Y., YU, J., YU, X., AND LEE, S. 2008. Line-art illustration of dynamic and specular surfaces. ACM Transactions on Graphics 27, 5, 156:1156:10.

[LCD06] T.Luft, C.Colditz, O.Deussen *Image Enhancement by Unsharp Masking the Depth Buffer*, ACM Transactions on Graphics, 2006

[LK10] S.Laine, T.Karras *Two methos for fast ray-casted ambient occlusion*, EGSR 2010

[McG10] M.McGuire *Ambient Occlusion Volumes*, In Proceedings of High Performance Graphics 2010 (Saarbrucken, Germany)

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

[MMAH07] M.Malmer, F.Malmer, U.Assarsson, N.Holzschuch *Fast precomputed ambient occlusion for proximity shadows*, journal of graphics, gpu, and game tools 12, 2(2007), 59-71

[NSW09] G.Nichols, J.Shopf, C.Wyman *Hierarchical image-space radiosity for interactive global illumination*, Computer Graphics Forum 28, 4 (2009), 11411149.

[NW09] G.Nichols, C.Wyman *Multiresolution splatting for indirect illumination*, In Proc. Symp. Interactive 3D Graphics and Games (2009), pp. 8390.

[NPW10] G.Nichols, R.Penmatsa, C.Wyman *Interactive, multiresolution image-space rendering for dynamic area lighting*, Eurographics Symposium on Rendering 29, 4.

[RBA09] C. Reinbothe and T.Boubekeur and M.Alexa, *Hybrid Ambient Occlusion*, EUROGRAPHICS 2009 Areas Papers, 2009

[SA07] P. Shanmugan, O.Arikan *Hardware accelerated ambient occlusion techniques on GPUs*, In Preceedings of SI3D 2007 New York, Ny, USA, 2000, ACM

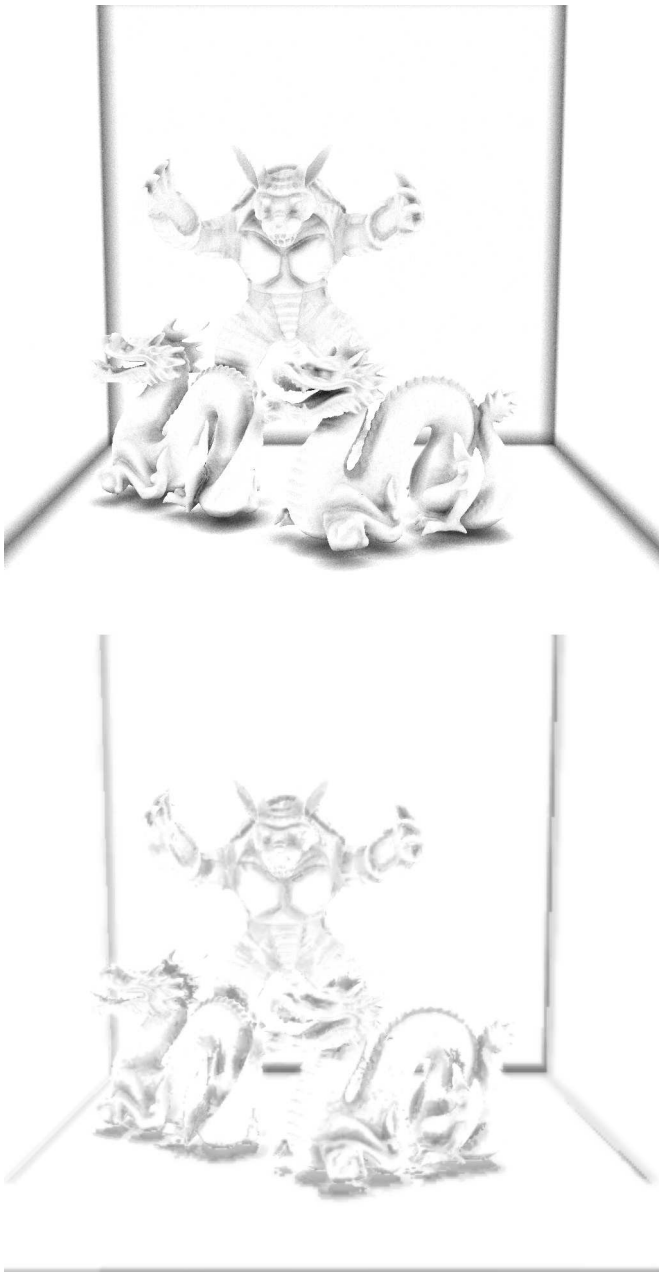

Figure 5: compares the ground truth against our implementation. Ground truth (Top) runs at 0.6fps and our our implementation runs at 40fps (Bottom) with the same number of samples.