Eurographics Symposium on Rendering (2022)
A. Ghosh and L.-Y. Wei (Editors)

Stenciled Volumetric Ambient Occlusion

Felix Brüll René Kern Thorsten Grosch

TU Clausthal, Germany

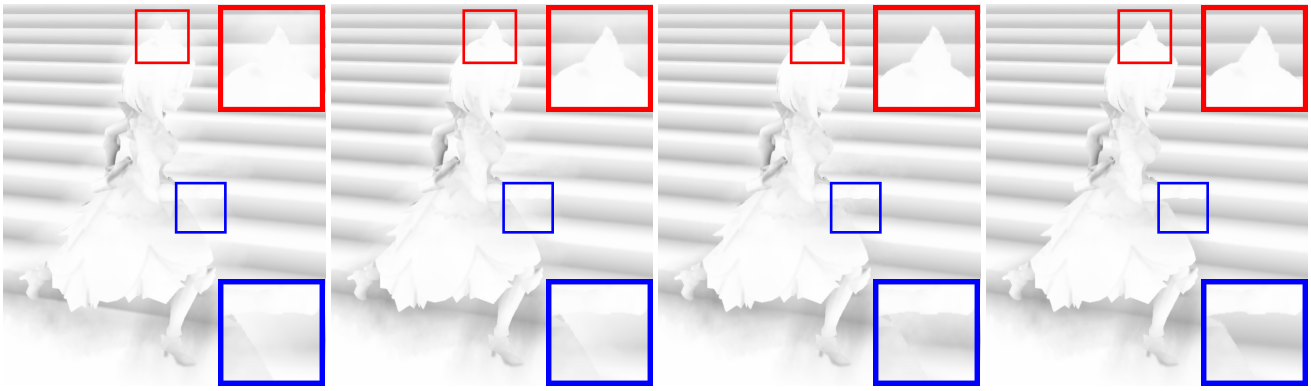


Figure 1: *The original VAO algorithm compared with our improved methods. Character model is Noelle from Genshin Impact [She21].*

Abstract

Screen-space Ambient Occlusion (AO) is commonly used in games to calculate the exposure of each pixel to ambient lighting with the help of a depth buffer. Due to its screen-space nature, it suffers from several artifacts if depth information is missing. Stochastic-Depth AO [VSE21] was introduced to minimize the probability of missing depth information, however, rendering a full stochastic depth map can be very expensive. We introduce a novel rendering pipeline for AO that divides the AO pass into two phases, which allows us to create a stochastic depth map for only a subset of pixels, in order to decrease the rendering time drastically. We also introduce a variant that replaces the stochastic depth map with ray tracing that has competitive performance. Our AO variants are based on Volumetric AO [LS10], which produces similar effects compared to the commonly used horizon-based AO [BSD08], but requires less texture samples to produce good results.

CCS Concepts

• ***Computing methodologies*** → ***Rasterization; Ray tracing;*** *Visibility;*

1. Introduction

Ambient Occlusion (AO) is an established technique to improve our perception of 3D scenes by darkening corners and creases. Screen-space techniques allow displaying AO within a few milliseconds, even for completely dynamic scenes and high pixel resolutions. However, missing depth information behind the first hit often leads to halo artifacts around depth discontinuities. Both hardware-supported ray tracing and a stochastic depth map can overcome this problem, but both are too expensive to compute for all pixels in the image. We therefore propose a *Stenciled Volumetric Ambient Occlusion* that quickly detects the small subset of pixels which require further depth information and selectively trace rays or compute stochastic depth values for them. The accompanying video and Fig. 1 show an example with a moving character that has a bright halo around itself, which can be removed with our techniques with only a small overhead.

To simplify notation, we define the AO value inversely as:

$$AO = \begin{cases} \mathbf{1}, \text{ if fully } \textbf{visible} \\ \mathbf{0}, \text{ if completely } \textbf{occluded} \end{cases}$$

This way, all AO images are consistent with the equations.

DOI: 10.2312/sr.20221153

2. Related Work

Ambient Occlusion was invented by Zhukov et al. [ZIK98] and used for production rendering by Landis [Lan02]. For dynamic scenes, Bunnell [Bun05] used disks to approximate AO. Ambient occlusion fields [KL05] can be used to precompute AO for rigid transformations in a grid structure. Completely dynamic scenes are possible with screen space methods [Mit07; FM08; SA07], where occlusion is approximated from nearby pixel depth values. This was further extended to directional occlusion and indirect bounces in screen space [RGS09]. To overcome the missing depth information, hybrid AO [RBA09] computes occlusion from a voxel representation. Jimenez et. al. [JWPJ16] uses a spherical harmonics representation along with spatio-temporal sampling to achieve high quality real-time AO on console hardware. Ray tracing can be used to compute occlusion in object space for extremely large, mostly static scenes using spatial hashing [Gau20]. Recent work also uses neural networks to compute AO directly from a geometry buffer without shaders [NAM*17]. In the following section we explain the AO methods that are most relevant for our work.

2.1. Horizon Based AO (HBAO)

Horizon Based AO (HBAO) [BSD08] samples the depth buffer 4 times in a single direction to find a horizon angle under which all directions are occluded. This is repeated for 8 directions in screen space resulting in 32 samples per pixel. Since the sample directions are randomized, the result is very noisy and needs to be blurred by a depth aware bilateral filter [ED04].

NVIDIA's HBAO+ [TP16] has the following improvements:

- It uses a simpler AO approximation, similar to [MML12].
- It uses interleaved rendering to minimize cache thrashing [BJ13].
- It can take a second depth buffer as input, to reduce artifacts.

Multiple approaches exists to obtain the second depth buffer:

- **Depth Peeling** [Eve01; BS09] can obtain the next layer of depth, which requires a second geometry pass. However, the next layer of depth can be insufficient to prevent all artifacts that are shown in Fig. 1. Note that the character model wears multiple layers of cloth and the second layer of depth will often be another layer of cloth, instead of the staircase.
- A **minimum separation distance** [MMNL16] for the second layer can alleviate the aforementioned problem.
- A pixel-synchronized **k-buffer** [Sal13] can capture up to k layers in a single geometry pass.
- The authors of HBAO+ introduce a fast approach to prevent the artifacts in Fig. 1. The first depth buffer is rendered with all geometry but exluding the character models. The depth buffer is then copied and all character models are rendered on top for the second depth buffer. This way, two depth buffers can be obtained with a single pass over all geometry.

Stochastic-Depth AO [VSE21] was introduced to capture multiple randomized depth layers in a multisample texture which they call stochastic depth map. A single layer of a stochastic depth map is shown in Fig. 2. Since they use a multisample texture, it is possible to have 1, 2, 4, 8 or 16 depth samples per pixel. The stochastic depth map is created in a second geometry pass, that outputs a random coverage mask to stochastically discard some depth values.

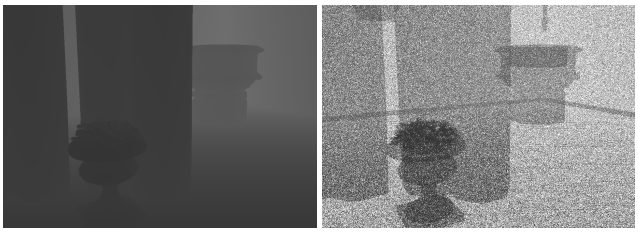


Figure 2: *Depth map (left) vs. stochastic depth map (right)*

2.2. Volumetric AO (VAO)

The first screen space AO [Mit07] by Crytek is closely related to volumetric AO: Point samples inside a sphere are used to estimate the unoccupied volume. Volumetric Ambient Occlusion [SKUT*09; BPB17] and Volumetric Obscurance [LS10] refined this idea by taking line samples instead of point samples to estimate the volume (see Sec. 3.2). Statistical Volumetric Obscurance [HSEE15] further refines the idea of area samples that were mentioned in [LS10] by using summed area tables to evaluate the integral with a single sample.

3. Our AO Algorithms

Our goal is to combine a fast screen space AO with ray tracing for ambiguous depth samples. We choose VAO as the underlying AO algorithm, because it can work with as few as 8 samples, whereas HBAO needs 32 samples per pixel. This is important because the number of rays will depend on the number of samples per pixel.

Our contributions are:

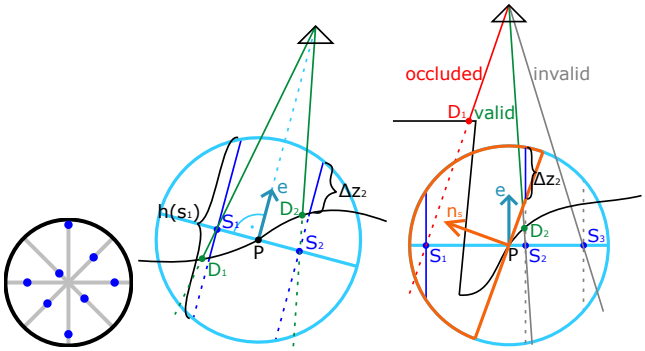
- An **optimized sample generation** for 8 samples per pixel that can capture small and large scale details (Sec. 3.1).
- A **modified thickness model** that allows us to cull depth samples more aggressively (Sec. 3.3).
- **Ray Traced VAO**: A variation of VAO that utilizes ray tracing for ambiguous depth samples (Sec. 3.6) and an optimized two phase pipeline (Sec. 3.8).
- **Stochastic Depth VAO**: A variation of VAO that makes use of stochastic depth maps (Sec. 3.7) and an optimized two phase pipeline that creates the map for selected pixels only (Sec. 3.9).

In this chapter we will explain the baseline VAO algorithm first, which will be similar to Volumetric Obscurance [LS10]. Next, we explain how to use ray tracing or a stochastic depth map to improve the visual quality of VAO. Finally, we explain our optimized stenciled approaches.

The full source code is available on `https://github.com/kopaka1822/Falcor/tree/ambient_occlusion`.

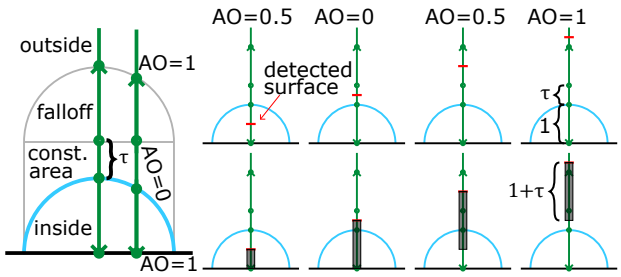
3.1. Sample Generation

We decided to generate $N = 8$ samples per pixel because we intend to store one bit visibility information per sample and eight samples fit nicely into a single byte. The goal is to generate samples on a unit disc to evaluate the occupied volume of a unit sphere. Unfortunately, sample generation via poisson disc sampling [SKUT*09;



(a) *Samples* **(b)** *Sphere Sampling* **(c)** *Hemisphere Sampling*

Figure 3: *VAO (hemi-)sphere sampling. The samples are distributed on the tangent plane perpendicular to the eye vector e. The latter can produce ray samples, that lie entirely below the hemisphere (S_3).*



(a) *Definition* **(b)** *Interpretation*

Figure 4: *(a) shows the different areas of our halo prevention model: Inside the sphere, AO will gradually decrease when a surface gets closer to the sphere boundary. In the constant area, AO will remain zero. In front of the constant area, AO will gradually increase back to 1. The size of the falloff area is identical to the size of the area inside. The size of the constant area is determined by the user-defined thickness τ. (b) shows the interpretation of this definition for a single depth sample (top): The AO value corresponds to the part of the column which is inside the hemisphere (bottom).*

LS10] produces an unfavorable distribution for $N = 8$ where 7/8 samples are placed with similar radii around the center of the disc.

Therefore, we use a quasi-Monte Carlo method to evaluate the occupied volume. First let us take a look on how to analytically compute the volume V of a unit sphere by integrating the height of a sample in 2D polar coordinates $h(r,\varphi)$ over a unit disc:

$$V = \int_0^1 \int_0^{2\pi} h(r,\varphi)\, r d\varphi\, dr = \frac{4}{3}\pi \tag{1}$$

$$h(r,\varphi) = 2\sqrt{1-r^2} \tag{2}$$

Later, we replace $h(r,\varphi)$ with the actual unoccupied height based on the depth samples. We reformulate this for Monte Carlo integration:

$$V \approx \frac{1}{N}\sum_{i=1}^{N} \frac{h(r_i,\varphi_i)}{pdf(r_i,\varphi_i)} \tag{3}$$

The optimal probability density *pdf* would be proportional to h. Based on this, we generate $N = 8$ samples on the unit disc after using inversion sampling for the radius (see appendix Sec. 7.1):

$$\varphi_i = 2\pi\frac{i}{8} + \xi \tag{4}$$

$$r_i = \sqrt{1-\psi_i^{2/3}} \tag{5}$$

We use the base 2 Van der Corput sequence [van35] from 8 to 15 which gives us 8 uniformly distributed random numbers for ψ_i: $\frac{1}{16}, \frac{9}{16}, \frac{5}{16}, \frac{13}{16}, \frac{3}{16}, \frac{11}{16}, \frac{7}{16}, \frac{15}{16}$. ξ is a random variable that is choosen for each pixel in a 4x4 pixel block to trade banding artifacts for noise. This results in the blue sampling points depicted in Fig. 3a.

3.2. Sample Evaluation

For simplicity we only describe how to sample a unit sphere. Fig. 3b depicts how we obtain line samples: We orient the sampling disc perpendicular to the eye-vector e centered around the world-space position P. After placing a sample s_i on the sampling disc, we project its world space position S_i to screen space, to obtain a depth sample from the depth buffer. Next, we calculate the world space position of the depth sample D_i. With this we approximate the distance between the sphere entry point and the depth sample:

$$\Delta z_i = \frac{h(s_i)}{2} - dot(D_i - P, e) \tag{6}$$

The normalized Monte Carlo integral for AO is then:

$$AO = \frac{1}{8}\sum_{i=1}^{8} a_i, \quad a_i = \frac{\Delta z_i}{h(s_i)} \tag{7}$$

a_i is the AO value of a single sample and AO is the solution of the Monte Carlo integral. We would like to note, that the calculation of Δz_i is an approximation of the true value for an orthographic camera.

In our work we use hemisphere sampling as in [LS10]: We only calculate Δz_i inside the hemisphere which is defined by the surface normal n_s (see Fig. 3c). We multiply AO by two, because half of the sphere is ignored. Note, that this can result in invalid samples that lie completely below the hemisphere. For these we set $a_i = 0$ instead of evaluating the depth.

3.3. Halo Prevention

So far, we did not handle occluded rays, as shown in Fig. 3c, which would result in negative Δz_i values. Assuming full occlusion for occluded rays will result in black halos around objects [SKUT*09; LS10]. Ignoring surfaces in front of the hemisphere has noticeable artifacts: When a surface moves into the hemisphere the AO value will jump from 1 to 0 instantly. To avoid this, we use a linear falloff function as in [HSEE15] that smoothly fades out the ambient value over distance. We modify this by introducing a *constant area* before we use the linear falloff, as shown in Fig. 4. This mimics the thickness model from [LS10] and also speeds up the rendering process later: we only consider rays as **occluded**, when a surface appears

$\tau = 0$ $\tau = 0.5$ $\tau = 1.0$

Figure 5: *Increasing the thickness τ leads to stronger darkening and can emphasize small, thin objects, like the leaves.*

before the constant area. Rays with a surface inside the constant area will be considered **valid** as well.

The height of the constant area depends on the unit sphere height of the sample position $h(s)$ and a thickness value τ:

$$\text{const. area} = 1 + \tau - \frac{h(s)}{2}. \tag{8}$$

We set $\tau \in [0,1]$ manually for each scene, the impact of the thickness is visualized in Fig. 5.

3.4. Postprocessing

Since we randomize each sample angle φ_i in a 4x4 pixel block, our AO is noisy. This noise can be removed with a bilateral filter of radius 4. We use the same separated depth aware bilateral blur filter as HBAO+ [TP16].

3.5. Volumetric Ambient Occlusion (VAO)

The baseline VAO method is executed in a single full screen pass that takes a linearized depth buffer and face normals as input. Eight samples are generated as explained in Sec. 3.1 and a_i is computed based on Eq. 7 if a depth sample is inside the hemisphere. If a depth sample is below the hemisphere, we set $a_i = 1$, otherwise, we use the halo model from Sec. 3.3. Finally, we filter the result as discussed in Sec. 3.4.

3.6. Ray Traced VAO (RT-VAO)

Every time the baseline VAO produces an **occluded** ray (Fig. 3c) for a single sample, a better partial AO value a_i could be obtained by using ray tracing after the point of occlusion: Based on our halo model from from Sec. 3.3, the AO value is smaller than 1 if we have intersections between the start of the falloff area and the end of our sampling hemisphere. Therefore, we only need to look for intersections in this area. We compute the AO value a_{ij} for all surfaces j inside this area and set the final AO value to $a_i = \min\{a_{i1}, \cdots, a_{ij}\}$. This guarantees smooth changes of the AO value as intersections enter and leave the halo area. Fig. 6 shows a simple example for a pixel with 4 depth values.

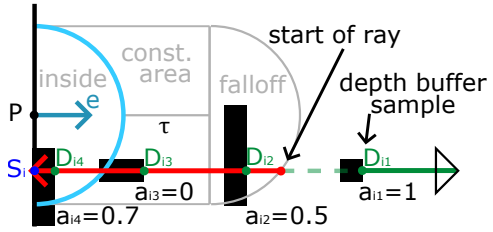
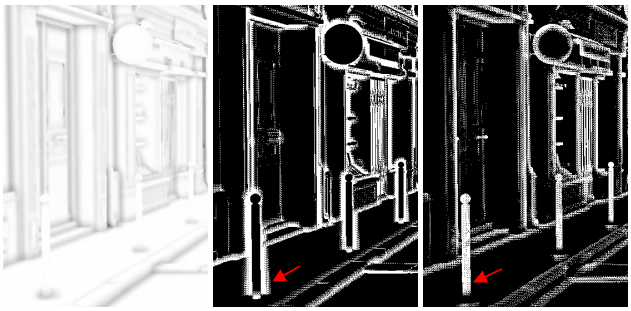


Figure 6: *Multiple depth samples D_{i1} to D_{i4} and their corresponding AO values a_{i1} to a_{i4} of sample position S_i. The initial depth sample D_{i1} is in front of the falloff area and its corresponding AO value is $a_{i1} = 1$. Since this sample is also in front of the constant area, it will be classified as **occluded** (too far away from S_i). With ray tracing we can obtain the **valid** intersections D_{i2} to D_{i4} and compute their AO values a_{i2} to a_{i4}. The final AO value is then set to: $a_i = \min(a_{i1}, a_{i2}, a_{i3}, a_{i4}) = a_{i3}$.*



(a) *Our VAO* **(b)** *Bitmask* **(c)** *SD Mask*

Figure 7: *Visualization of our VAO (a) in the bistro scene [Lum17]. Notice the pole on the lower left: In the bitmask (b), the pixels around the pole are marked because their depth values are behind the pole and their sample disks are partially occluded by the pole. In the stochastic depth mask (c), the pole itself is marked, because additional depth samples are required for this area.*

We implemented this in a single full screen pass by using DirectX ray queries (**RQ-VAO**) and also in a dedicated ray tracing pass with *TraceRay()* (**RT-VAO**).

3.7. Stochastic Depth VAO (SD-VAO)

Since ray tracing hardware is not available for all consumers, we worked on an alternate approach that uses a stochastic depth map [VSE21]. This achieves similar results in comparable time.

Similar to the previous approach, we evaluate all samples of a stochastic depth map if and only if a sample is considered occluded. Again, we compute the AO value a_{ij} for all stochastic depth samples j and set: $a_i = min\{a_{i1}, \cdots, a_{ij}\}$.

3.8. Ray Traced Stenciled VAO (RT-SVAO)

We observed that the ray query version RQ-VAO performed considerably worse than the dedicated ray tracing pass RT-VAO. We concluded that this is due to the high thread divergence that occurs,

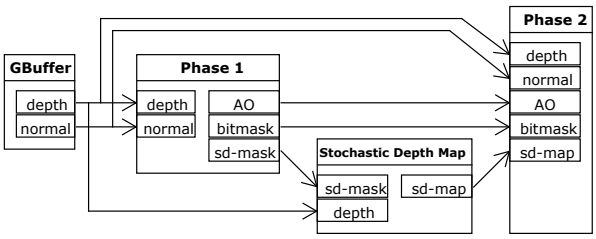


Figure 8: *The pipeline of our SD-SVAO (without bilateral-filter). Each module has the required inputs on the left side and the resulting outputs on the right.*

	time	total (inc. single)
single depth	0.34	0.34
depth peeling	0.37	0.71
k-buffer (k=2)	3.51	3.51
SD2	0.68	1.02
SD4	0.75	1.09
SD8	1.2	1.54
Stenciled-SD2	0.39	0.73
Stenciled-SD4	0.42	0.76
Stenciled-SD8	0.55	0.89

Table 1: *Rendering time in ms of various depth buffer techniques in the bistro [Lum17].*

when only few samples need to be raytraced. In this case, the shader needs to wait for all workgroup threads to finish ray tracing, before it can continue when using ray queries.

The dedicated ray tracing pipeline already works around this problem by regrouping threads on the fly. However, we improved the performance even further by making use of domain knowledge. Motivated by [LKA13] we split our AO kernel in two phases to minimize thread divergence:

In the **first phase** we only evaluate the depth buffer in a full screen pass. In addition to the AO value, we output an 8-bit bitmask for each pixel: Whenever we encounter an occluded sample, we set its contribution to zero and set the i-th bit in our bitmask. An example of the resulting bitmask is shown in Fig. 7b.

In the **second phase** we only do ray tracing for the pixels, where at least one bit of the bitmask is set. These are the pixels, where at least one sample was occluded.

We implemented the second phase with ray queries (**RQ-SVAO**) and with the dedicated ray tracing pass (**RT-SVAO**).

3.9. Stochastic Depth Stenciled VAO (SD-SVAO)

Since the generation of a stochastic depth map is expensive, we decided to create a stochastic depth map only for selected pixels. For this, we adjusted the previously explained two phase pipeline as follows (see Fig. 8):

In the **first phase** we only evaluate the depth buffer. We output the same AO values and bitmask as before. Additionally, we use a write-only random access texture to mark pixels where additional depth values are needed due to occluded samples. We refer to this texture as stochastic depth mask, which is shown in Fig. 7c: Whenever we encounter an occluded sample, we additionally write a 1 into the stochastic depth mask at the screen space position of our sample S_i (which differs from our own screen space position).

Next we create a **Stochastic Depth Map** only for pixels where the stochastic depth mask is nonzero. For this we copy the values from the stochastic depth mask into the stencil buffer of our stochastic depth map (D24_S8 format) and enable the early stencil test for rendering to save bandwidth.

In the full screen pass of the **second phase** we use the bitmask to mask out inactive invocations. Here, we evaluate the samples from the generated stochastic depth as in Sec. 3.7.

4. Results

For a fair timing comparison we implemented all techniques from the previous section in Falcor [KCK*22]. All renderings were done on the NVIDIA RTX 2080 Ti. We also implemented HBAO+ [TP16] from section Sec. 2.1 with interleaved rendering.

Additionally we use **HBAO+SD**, which is the implementation of stochastic depth AO [VSE21]. However, as opposed to the original paper, we implemented interleaved rendering for the primary depth map to achieve a more competitive performance.

In Sec. 4.1 we will give an overview about the cost of obtaining multiple depth buffers in various ways. In Sec. 4.2 we will look at the rendering performance of our algorithms. In Sec. 4.3 we will demonstrate the visual benefits between the techniques.

4.1. Secondary Depth Buffer

Tab. 1 shows that depth peeling [BS09] a single layer of depth is only slightly more expensive than a normal depth rendering pass. Creating a stochastic depth map with two layers (SD2) is twice as expensive. Notably, our stenciled versions of the stochastic depth map are only slightly more expensive than an actual depth peeling pass. In this work we use a stochastic depth map with 4 samples per pixel (SD4) and D24_S8 format.

The pixel synchronized k-buffer [Sal13] performs considerably worse than all other techniques, even though it requires only a single pass over the geometry. We suspect that the serialization of the pixel shader due to the pixel synchronization is the reason.

4.2. AO Performance

We consider AO to be part of a bigger rendering pipeline and therefore expect the linearized depth buffer and face normals to be available without additional cost. We include the times for the acquisition of the stochastic depth map, the AO computation itself and the time for postprocessing blur in our benchmarks.

Fig. 9 shows the used scenes with their respective bitmasks.

	SD map	AO	Total	Speedup
VAO		0.22	0.44	
SD-VAO	0.35	0.29	0.86	
SD-SVAO	0.18	0.23+0.1	0.73	1.17
RT-VAO		0.9	1.12	
RQ-SVAO		0.22+0.48	0.92	1.22
HBAO+		0.38	0.6	
HBAO+SD	0.35	1.0	1.57	

Times in ms for Crytek Sponza [McG17]. Blur is 0.22 ms.

VAO		0.22	0.44	
SD-VAO	0.85	0.31	1.38	
SD-SVAO	0.76	0.24+0.13	1.35	1.02
RT-VAO		1.36	1.58	
RQ-SVAO		0.24+0.7	1.16	1.36
HBAO+		0.38	0.6	
HBAO+SD	0.35	1.19	2.26	

Times in ms for Zero Day [Win19]. Blur is 0.22 ms.

VAO		0.21	0.46	
SD-VAO	1.74	0.32	2.31	
SD-SVAO	1.45	0.21+0.15	2.06	1.12
RT-VAO		3.74	3.99	
RT-SVAO		0.23+3.26	3.74	1.07
HBAO+		0.39	0.64	
HBAO+SD	1.74	1.39	3.38	

Times in ms for Emerald Sqr. [NHB17]. Blur is 0.25 ms.

VAO		0.22	0.44	
SD-VAO	1.83	0.24	2.29	
SD-SVAO	1.58	0.23+0.10	2.13	1.08
RT-VAO		2.08	2.30	
RT-SVAO		0.21+1.93	2.36	0.97
HBAO+		0.31	0.53	
HBAO+SD	1.83	0.77	2.82	

Times in ms for Hairy Desert. Blur is 0.22 ms.

Table 2: *Rendering times for 1920x1080 with 64 pixel guard band. The AO times for SVAO are split into first phase + second phase.*

	SD map	AO	Total	Speedup
VAO		0.21	0.45	
SD-VAO	0.83	0.29	1.36	
SD-SVAO	0.43	0.21+0.10	0.98	1.39
RT-VAO		1.18	1.42	
RT-SVAO		0.21+0.81	1.26	1.13
HBAO+		0.3	0.64	
HBAO+SD	0.83	1.0	2.17	

1920x1080 with 64 pixel guard band. Blur is 0.24.

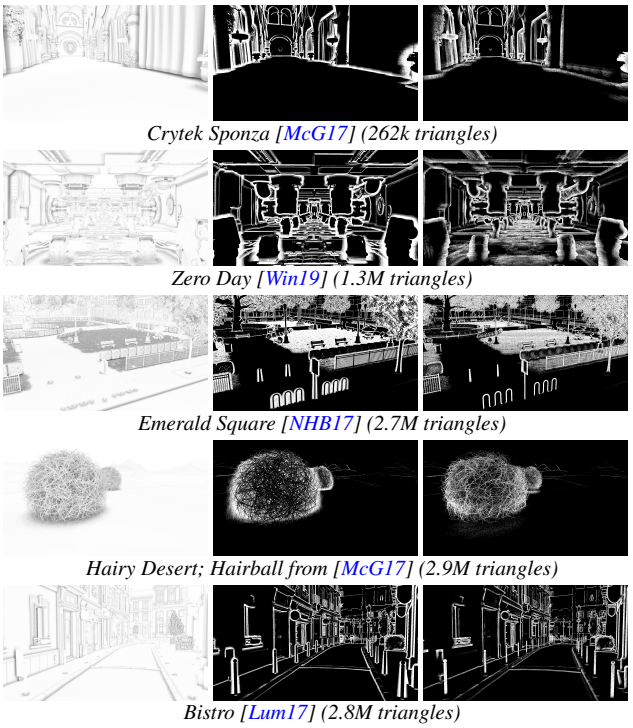
VAO		0.35	0.69	
SD-VAO	1.30	0.45	2.09	
SD-SVAO	0.56	0.40+0.14	1.44	1.45
RT-VAO		1.75	2.09	
RQ-SVAO		0.38+1.18	1.90	1.10

2560x1440 with 96 pixel guard band. Blur is 0.34.

VAO		0.64	1.38	
SD-VAO	2.50	0.88	4.12	
SD-SVAO	1.02	0.70+0.26	2.72	1.51
RT-VAO		3.08	3.82	
RQ-SVAO		0.64+2.06	3.44	1.11

3840x2160 with 128 pixel guard band. Blur is 0.74.

Table 3: *Times for different resolutions in the bistro [Lum17]*



Crytek Sponza [McG17] (262k triangles)

Zero Day [Win19] (1.3M triangles)

Emerald Square [NHB17] (2.7M triangles)

Hairy Desert; Hairball from [McG17] (2.9M triangles)

Bistro [Lum17] (2.8M triangles)

Figure 9: *Overview of used scenes. From left to right: Ray Traced VAO, Bitmask, Stochastic Depth Mask.*

Tab. 2 includes the rendering times for HD resolution and Tab. 3 depicts the rendering times for resolutions from HD to 4k.

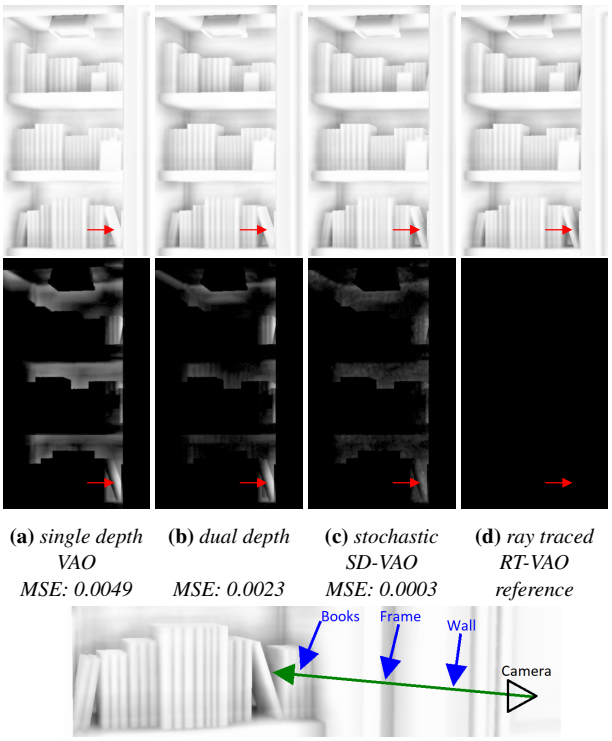
In all cases we were able to reduce the rendering time for the stochastic depth map. Furthermore, our SD-SVAO is always faster than the non-stenciled SD-VAO and the performance improvement ranges from 2% to 51%. As expected, scenes with a lot of self occlusion were more expensive to render.

It is also noticeable that VAO is a better fit for stochastic depth maps in general when compared to the runtimes of HBAO, because it uses less texture samples and therefore requires less bandwidth.

Our Ray Traced SVAO was also almost always faster than the non-stenciled counterpart. Only the stress test scene of the hairball proved to be difficult. In the remaining scenes, the performance improvements range from 7% to 36%.

4.3. Image Quality

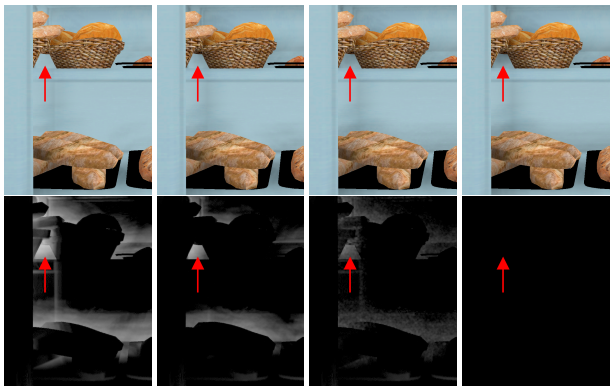
Fig. 1 already showed an example where the baseline VAO has noticeable halos near depth discontinuities. These artifacts become less recognizable after supplying a second depth buffer (dual depth), but are still visible. Our RT-VAO completely removes the bright halos, while our SD-VAO shows significant improvements over the simple dual depth variant. In Fig. 10 and Fig. 11 we present two more examples. Generally, the stochastic depth map is not a perfect replacement for ray tracing, but it handles the difficult cases much better than a single depth buffer. This is also confirmed by the Mean Squared Error (MSE) below the two figures. The dual depth



(a) *single depth VAO* MSE: 0.0049 **(b)** *dual depth* MSE: 0.0023 **(c)** *stochastic SD-VAO* MSE: 0.0003 **(d)** *ray traced RT-VAO reference*

(e) *different view angle with occluders (blue)*

Figure 10: *VAO renderings for a bookstore window in the bistro. The top row shows VAO renderings, the middle row shows difference images to RT-VAO. The bottom row shows the scene from a different angle: single and dual depth VAO can not properly compute the area marked by the red arrow, because nearby samples are blocked by up to three different occluders marked in blue.*



(a) *single depth VAO* MSE: 0.0057 **(b)** *dual depth* MSE: 0.0014 **(c)** *stochastic SD-VAO* MSE: 0.0003 **(d)** *ray traced RT-VAO reference*

Figure 11: *Colored VAO renderings for a shop window of a bakery store in the bistro. The top row shows VAO renderings, the bottom row shows differences to RT-VAO. The red arrow marks an area that is difficult for single and dual depth VAO due to the amount of nearby occluders: The window frame, bread baskets and pastries.*

variant handles cases with up to two occluders perfectly, however, as soon as a third occluder is introduced one can observe white halos or missing shadows. This effect is especially noticeable during camera motion. Stochastic depth maps are better at handling such cases since the relevant depth layer is generally captured in at least some of the stochastic depth samples.

In the end, RT-VAO and SD-VAO reduce halos to a point where they are no longer noticeable after adding diffuse shading as shown in the supplemental video. However, both techniques require additional computations and should only be used if the time budget of the specific application allows it, which is already the case for some of the presented scenes. Also note that RT-VAO and RT-SVAO produces identical visual results just like SD-VAO and SD-SVAO do.

4.4. VAO vs. HBAO

In Fig. 12 and Fig. 13 we show renderings of our SD-VAO and Vermeers HBAO+SD [VSE21]. Both techniques result in different images, however, both manage to darken corners and creases in their own way. With VAO the corners are sometimes darker, and with HBAO the falloff is generally larger and appears softer. After adding the diffuse color, the differences become even less noticeable. Please note, that our optimized SD-SVAO renders the scene more than twice as fast as HBAO+SD (see Tab. 3).

5. Limitations and Future Work

Compared to RT-VAO, SD-VAO partially underestimated the occlusion since sometimes depth samples are missing. The authors [VSE21] proposed a compensation for HBAO+ which increases the contribution of stochastic samples to alleviate this artifacts. Unfortunately, we could not come up with a similar countermeasure for VAO that did not introduce more artifacts.

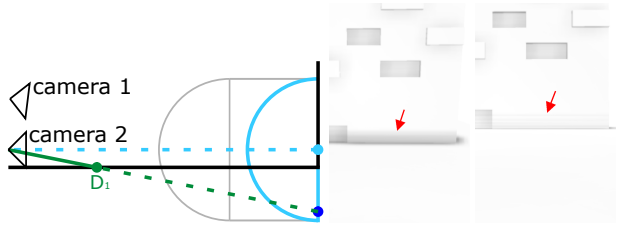
VAO++ [BPB17] presented some different optimizations which are orthogonal to our work: They introduce a culling pre-pass, which evaluate AO in a lower resolution texture first. Afterwards, full screen AO is only evaluated for partially occluded pixels of the low resolution AO. This would primarily benefit the first phase of our VAO. Additionally, they use an adaptive sample count. Based on the projected radius in screen space, they vary between 2 and 32 samples per pixels. Ultimately, this might also be beneficial for the second phase of our VAO since less rays or stochastic depth samples need to be evaluated for distant geometry.

Increasing the number of samples beyond 8 is possible, but requires more bandwidth. Since one bit per sample is required for our visibility bitmask, sample counts up to 128 would be possible with current hardware. Note that our bitmask is created with a render target, instead of a stencil target, because our hardware does not support writing arbitrary stencil values from the pixel shader. We create the stencil target in a separate full screen pass, that writes a 1 for each pixel where the bitmask is nonzero, which is sufficient for the stencil test. Therefore, the maximum size of 8 bits per pixel for stencil formats is not a limitation.

It would be benefical to be able to limit the maximum number of rays or stochastic depth samples per frame for a guaranteed frame

Figure 12: *SD-VAO (top) and HBAO+SD [VSE21] (bottom) in the Bistro [Lum17]*

Figure 13: *SD-VAO (top) and HBAO+SD [VSE21] (bottom) in the Bistro [Lum17]*



(a) *sketch* **(b)** *camera 1* **(c)** *camera 2*

Figure 14: *VAO has problems when looking from grazing angles like camera 2. Intersection D_1 is not identified as occluder because it is outside of the halo area. The same problem exists in HBAO.*

rate in real-time applications. Alternatively, it would also be possible to evaluate the primary depth map in full resolution and to perform the second phase of VAO in half- or quarter-resolution. This should remove most aliasing artifacts, since the majority of samples comes from the primary depth map.

The most significant problem of our VAO is depicted in Fig. 14: Objects that are faced towards the camera can have half of their sampling hemisphere below the ground in the worst case. If the viewer looks at grazing angles, a sample below the ground is unlikely to be detected as occluded, even when using the halo prevention model. This is problematic for two reasons: First, corners are shaded brighter than they should be. This effect can be alleviated by choosing a higher thickness τ. Second, such depth samples will be counted as occluded which forces an additional ray or stochastic depth sample. This has a negative impact on the overall performance and is the main problem of the Emerald Square scene: Here, the grass blades in the park are placed perpendicular on the ground which causes a huge area of the scene to be reevaluated. Removing the grass would result in a major performance boost in that scenario. Still, we would like to work on a more sophisticated technique to handle such cases.

6. Conclusion

We proposed the combination of Volumetric Obscurance [LS10] with stochastic depth maps [VSE21] and ray tracing to improve the visual quality. In addition, we introduced a novel two phase pipeline for AO to accelerate the execution of the presented methods. The actual performance improvements depend on the complexity of the scene and range from 2% to 51% with one exception. The stencil optimizations also appear to scale favorably with increasing screen resolution. Furthermore, we modified the thickness model with a configurable parameter τ to trade minor artifacts for increased performance and also to alleviate artifacts in corners.

References

[BJ13] BAVOIL, LOUIS and JANSEN, JON. "Particle Shadows and Cache-Efficient Post-Processing". *Game Developer Conference* (2013). https://developer.nvidia.com/gdc-2013 Accessed: 2022-03-07 2.

[BPB17] BOKŠANSKÝ, JAKUB, POSPÍŠIL, ADAM, and BITTNER, JIŘÍ. "VAO++: Practical Volumetric Ambient Occlusion for Games". *Eurographics Symposium on Rendering - Experimental Ideas and Implementations*. Ed. by ZWICKER, MATTHIAS and SANDER, PEDRO. The Eurographics Association, 2017. DOI: 10.2312/sre.20171192 2, 7.

[BS09] BAVOIL, LOUIS and SAINZ, MIGUEL. "Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion". *SIGGRAPH 2009: Talks*. SIGGRAPH '09. New Orleans, Louisiana: Association for Computing Machinery, 2009. DOI: 10.1145/1597990.1598035 2, 5.

[BSD08] BAVOIL, LOUIS, SAINZ, MIGUEL, and DIMITROV, ROUSLAN. "Image-space horizon-based ambient occlusion". *ShaderX7 Advanced Rendering Techniques*. July 2008. DOI: 10.1145/1401032.1401061 1, 2.

[Bun05] BUNNELL, MICHAEL. "Dynamic Ambient Occlusion and Indirect Lighting". *GPU Gems 2*. Ed. by PHARR, MATT. Addison-Wesley, 2005, 223–233 2.

[ED04] EISEMANN, ELMAR and DURAND, FRÉDO. "Flash Photography Enhancement via Intrinsic Relighting". *ACM Transactions on Graphics (Proceedings of Siggraph Conference)* 23 (Aug. 2004). DOI: 10.1145/1015706.1015778 2.

[Eve01] EVERITT, CASS. *Interactive Order-Independent Transparency*. Oct. 2001. URL: https://pdfs.semanticscholar.org/99b8/940b5a6dab8527198e966c0eb7e2a02ee28c.pdf 2.

[FM08] FILION, DOMINIC and MCNAUGHTON, ROBERT. "Effects & techniques". *SIGGRAPH '08*. 2008 2.

[Gau20] GAUTRON, PASCAL. "Real-Time Ray-Traced Ambient Occlusion of Complex Scenes Using Spatial Hashing". *ACM SIGGRAPH 2020 Talks*. SIGGRAPH '20. Virtual Event, USA: Association for Computing Machinery, 2020. ISBN: 9781450379717. DOI: 10.1145/3388767.3407375. URL: https://doi.org/10.1145/3388767.3407375 2.

[HSEE15] HENDRICKX, QUINTJIN, SCANDOLO, LEONARDO, EISEMANN, MARTIN, and EISEMANN, ELMAR. "Adaptively Layered Statistical Volumetric Obscurance". *Proceedings of the 7th Conference on High-Performance Graphics*. HPG '15. Los Angeles, California: Association for Computing Machinery, 2015, 77–84. DOI: 10.1145/2790060.2790070 2, 3.

[JWPJ16] JIMENEZ, JORGE, WU, XIAN-CHUN, PESCE, ANGELO, and JARABO, ADRIAN. "Practical Real-Time Strategies for Accurate Indirect Occlusion". *SIGGRAPH 2016 Courses: Physically Based Shading in Theory and Practice*. 2016. URL: https://www.activision.com/cdn/research/Practical_Real_Time_Strategies_for_Accurate_Indirect_Occlusion_NEW%20VERSION_COLOR.pdf 2.

[KCK*22] KALLWEIT, SIMON, CLARBERG, PETRIK, KOLB, CRAIG, et al. *The Falcor Rendering Framework*. Jan. 2022. URL: https://github.com/NVIDIAGameWorks/Falcor 5.

[KL05] KONTKANEN, JANNE and LAINE, SAMULI. "Ambient Occlusion Fields". *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*. I3D '05. Washington, District of Columbia: Association for Computing Machinery, 2005, 41–48. DOI: 10.1145/1053427.1053434 2.

[Lan02] LANDIS, HAYDEN. "Production-Ready Global Illumination". *SIGGRAPH Courses*. 2002, 87–102 2.

[LKA13] LAINE, SAMULI, KARRAS, TERO, and AILA, TIMO. "Megakernels Considered Harmful: Wavefront Path Tracing on GPUs". *Proceedings of the 5th High-Performance Graphics Conference*. HPG '13. New York, NY, USA: Association for Computing Machinery, 2013, 137–143. DOI: 10.1145/2492045.2492060 5.

[LS10] LOOS, BRADFORD JAMES and SLOAN, PETER-PIKE. "Volumetric Obscurance". *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. Washington, D.C.: Association for Computing Machinery, 2010, 151–156. DOI: 10.1145/1730804.1730829 1–3, 9.

[Lum17] LUMBERYARD, AMAZON. *Amazon Lumberyard Bistro, Open Research Content Archive (ORCA)*. 2017. URL: http://developer.nvidia.com/orca/amazon-lumberyard-bistro 4–6, 8.

[McG17] MCGUIRE, MORGAN. *Computer Graphics Archive*. 2017. URL: https://casual-effects.com/data 6.

[Mit07] MITTRING, MARTIN. "Finding next gen: CryEngine 2." *SIGGRAPH Courses*. Ed. by MCMAINS, SARA and SLOAN, PETER-PIKE. ACM, 2007, 97–121. URL: http://dblp.uni-trier.de/db/conf/siggraph/siggraph2007courses.html#Mittring07 2.

[MML12] MCGUIRE, MORGAN, MARA, MICHAEL, and LUEBKE, DAVID. "Scalable Ambient Obscurance". *Proceedings of ACM SIGGRAPH / Eurographics High-Performance Graphics 2012 (HPG '12)* (2012). High-Performance Graphics 2012. URL: https://casual-effects.com/research/McGuire2012SAO/index.html 2.

[MMNL16] MARA, MICHAEL, MCGUIRE, MORGAN, NOWROUZEZAHRAI, DEREK, and LUEBKE, DAVID. "Deep G-Buffers for Stable Global Illumination Approximation". *Proceedings of the High Performance Graphics 2016*. HPG. 2016, 11. URL: https://casual-effects.com/research/Mara2016DeepGBuffer/index.html 2.

[NAM*17] NALBACH, OLIVER, ARABADZHIYSKA, ELENA, MEHTA, DUSHYANT, et al. "Deep Shading: Convolutional Neural Networks for Screen Space Shading". *Comput. Graph. Forum* 36.4 (2017), 65–78. DOI: 10.1111/cgf.13225. URL: https://doi.org/10.1111/cgf.13225 2.

[NHB17] NICHOLAS HULL, KATE ANDERSON and BENTY, NIR. *NVIDIA Emerald Square, Open Research Content Archive (ORCA)*. 2017. URL: http://developer.nvidia.com/orca/nvidia-emerald-square 6.

[RBA09] REINBOTHE, CHRISTOPH, BOUBEKEUR, TAMY, and ALEXA, MARC. "Hybrid Ambient Occlusion". *EUROGRAPHICS 2009 Areas Papers* (2009) 2.

[RGS09] RITSCHEL, TOBIAS, GROSCH, THORSTEN, and SEIDEL, HANS-PETER. "Approximating Dynamic Global Illumination in Image Space". *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*. I3D '09. Boston, Massachusetts: Association for Computing Machinery, 2009, 75–82. DOI: 10.1145/1507149.1507161 2.

[SA07] SHANMUGAM, PERUMAAL and ARIKAN, OKAN. "Hardware Accelerated Ambient Occlusion Techniques on GPUs". *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*. I3D '07. Seattle, Washington: Association for Computing Machinery, 2007, 73–80. DOI: 10.1145/1230100.1230113 2.

[Sal13] SALVI, MARCO. "Pixel synchronization: solving old graphics problems with new data structures". *Advances in Real-time Rendering* (2013) 2, 5.

[She21] SHENDIYU. *Noelle*. The model copyright belongs to miHoYo. 2021. URL: https://www.aplaybox.com/details/model/s620OQ2UgRu7 1.

[SKUT*09] SZIRMAY-KALOS, LÁSZLÓ, UMENHOFFER, TAMÁS, TÓTH, BALÁZS, et al. "Volumetric Ambient Occlusion". *IEEE Computer Graphics and Applications - CGA* (Jan. 2009). DOI: 10.1109/MCG.2009.106 2, 3.

[TP16] TATARINOV, ANDREI and PANTELEEV, ALEXEY. "Advanced Ambient Occlusion Methods for Modern Games". *Game Developer Conference*. https://developer.download.nvidia.com/gameworks/events/GDC2016/atatarinov_alpanteleev_advanced_ao.pdf Accessed: 2022-03-07. 2016 2, 4, 5.

[van35] VAN DER CORPUT, J. G. "Verteilungsfunktionen. I". German. *Proc. Akad. Wet. Amsterdam* 38 (1935), 813–821. ISSN: 0370-0348 3.

[VSE21] VERMEER, JOP, SCANDOLO, LEONARDO, and EISEMANN, ELMAR. "Stochastic-Depth Ambient Occlusion". *Proceedings of the ACM on Computer Graphics and Interactive Techniques*. I3D '21 4.1 (2021). DOI: 10.1145/3451268 1, 2, 4, 5, 7–9.

[Win19] WINKELMANN, MIKE. *Zero-Day, Open Research Content Archive (ORCA)*. 2019. URL: https://developer.nvidia.com/orca/beeple-zero-day 6.

[ZIK98] ZHUKOV, SERGEY, IONES, ANDREI, and KRONIN, GRIGORIJ. "An Ambient Light Illumination Model." *Rendering Techniques*. Ed. by DRETTAKIS, GEORGE and MAX, NELSON L. Eurographics. Springer, 1998, 45–56. URL: http://dblp.uni-trier.de/db/conf/rt/rt1998.html#ZhukovIK98 2.

7. Appendix

7.1. Inversion Sampling

The normalized *pdf* for the volume integral in Eq. 1 is:

$$pdf(r,\varphi) = \frac{3}{4\pi} h(r,\varphi) = \frac{3}{2\pi}\sqrt{1-r^2} \tag{9}$$

For this method, we determine how to choose a sample radius r_i based on a uniformly distributed random number $\psi_i \in [0,1]$:

$$\psi_i = \int_0^{r_i}\int_0^{2\pi} pdf(r,\varphi)\, r d\varphi\, dr \tag{10}$$

This simplifies to:

$$\psi_i = \int_0^{r_i} 3\sqrt{1-r^2} r\, dr \tag{11}$$

$$\psi_i = \left[-(1-r^2)^{3/2}\right]_0^{r_i} \tag{12}$$

$$\psi_i = 1-(1-r_i^2)^{3/2} \tag{13}$$

Due to the uniform distribution, we can replace ψ_i by $1-\psi_i$:

$$(1-r_i^2)^{3/2} = 1-\psi_i = \psi_i \tag{14}$$

$$1-r_i^2 = \psi_i^{2/3} \tag{15}$$

$$r_i = \sqrt{1-\psi_i^{2/3}} \tag{16}$$