Adaptively Layered Statistical Volumetric Obscurance

Quintijn Hendrickx[1] Leonardo Scandolo[1] * Martin Eisemann[1,2] † Elmar Eisemann[1] ‡

[1]Delft University of Technology [2]TH Köln

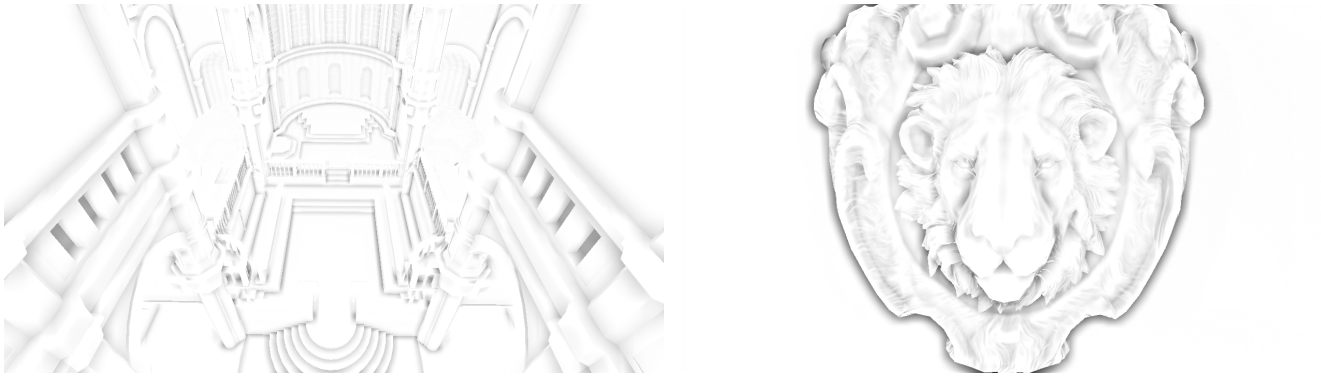


Figure 1: *Ambient Occlusion without shading. We can render images at 320 fps (*1280*x*720 *resolution, 294 MPixels/s) on a GTX 770.*

Abstract

We accelerate volumetric obscurance, a variant of ambient occlusion, and solve undersampling artifacts, such as banding, noise or blurring, that screen-space techniques traditionally suffer from. We make use of an efficient statistical model to evaluate the occlusion factor in screen-space using a single sample. Overestimations and halos are reduced by an adaptive layering of the visible geometry. Bias at tilted surfaces is avoided by projecting and evaluating the volumetric obscurance in tangent space of each surface point. We compare our approach to several traditional screen-space ambient obscurance techniques and show its competitive qualitative and quantitative performance. Our algorithm maps well to graphics hardware, does not require the traditional bilateral blur step of previous approaches, and avoids typical screen-space related artifacts such as temporal instability due to undersampling.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture;

Keywords: SSAO, Summed Area Tables, global illumination

1 Introduction

Efficient computation of global illumination is still one of the hardest problems in computer graphics. In consequence, real-time approximations often make very simplifying assumptions. *Ambient Occlusion* (AO) is an example and focuses on the evaluation of ambient light reaching a point on a surface [Landis 2002] by considering only local geometry as occluders in the scene. Attenuating the ambient light term based on local occlusion creates important contact cues improving overall depth perception.

Historically, AO was first applied in static scenes where its effect could be baked into occlusion maps [Landis 2002]. However, this approach does not work well for dynamic scenes and would need be applied per frame. In recent years, advances in graphics hardware and the development of screen-space approximations have led to real-time implementations of AO [Mittring 2007; Shanmugam and Arikan 2007]. These screen-space ambient occlusion (SSAO) techniques compute the amount of occlusion as a postprocessing pass based on a depth image from the camera's point of view. Traditionally, the occlusion factor is approximately estimated per pixel using a few samples and smoothed using a subsequent bilateral blur step. Most current rendering engines incorporate such solutions.

We aim at an approach that has the low computational complexity of screen-space ambient occlusion (SSAO) approaches, but avoids the usual drawbacks, such as banding, noise or blurriness caused by undersampling. In order to eliminate these artifacts, we have to account for *all* of the local geometry visible in screen-space. To this extent, we reverse the typical order of operations applied in existing SSAO approaches. Instead of taking samples and blurring the result afterwards, we compute a statistical model of the surrounding geometry at a pixel's world position and use it directly for AO computation. Because we do not use traditional sampling there is no need for randomization or blurring of the result [Mittring 2007].

Specifically, our contributions are:

- A screen-space ambient-occlusion approximation, evaluated using a single sample;
- An adaptive depth-slicing technique to efficiently compute this model;
- A GPU-friendly and highly-parallel implementation

In the following, we introduce an approximation for volumetric obscurance and how to compute it efficiently (Sec. 3) and describe how to improve quality via depth slicing (Sec. 4). For acceleration purposes, we introduce an adaptive technique (Sec. 5) and remove bias in the result by incorporating the surface normal into the computation (Sec. 6). We introduce important optimizations, like ap-

*e-mail:l.scandolo@tudelft.nl

†e-mail:martin.eisemann@fh-koeln.de

‡e-mail:e.eisemann@tudelft.nl

proximate summed-area tables (SAT) and differential SAT computation (Sec. 7). We evaluate and compare our approach to common screen-space ambient occlusion and volumetric-obscurance techniques (Sec. 8), before concluding (Sec. 9).

2 Related Work

The idea of approximating ambient illumination to account for local geometry was first described by Zhukov ([Zhukov et al. 1998]) and Landis [Landis 2002] showed the importance of AO in improving depth perception through contact cues and soft shadows. [Luft et al. 2006] employed a similar idea for artistic purposes. AO has since gathered a significant amount of interest resulting in numerous techniques. The algorithms can be divided in roughly two categories, geometry-based and screen-space ambient occlusion.

Geometry-based ambient occlusion incorporates all available geometry into the AO computation. Geometrical data in form of surface elements can be conveniently grouped in a hierarchy based on their distance to evaluate local AO [Bunnell 2005]. Alternatively, AO contributions can be scattered by each primitive via surrounding occlusion volumes [McGuire 2010]. While providing high-quality results the performance depends heavily on the geometrical complexity and AO radius. Density information can also be used for computing AO ([Hernell et al. 2010], [Grottel et al. 2012]) in the context of volume rendering.

Screen-space ambient occlusion techniques compute occlusion based on information in the depth buffer leading to (almost) geometry-independent evaluations. Such an approach was introduced by Crytek [Mittring 2007]. They sampled a sphere around a pixel's world position and reprojected the samples into the depth map to determine if they were occluded by geometry. For real-time performance the approach requires aggressive undersampling, which subsequently leads to noise artifacts that require an additional and costly bilateral blur step [McGuire et al. 2012].

Line sampling [Loos and Sloan 2010] improved upon the Crytek implementation by taking samples in the 2D projection of the sphere and integrating over line segments, thus computing the amount of geometry inside the sample sphere. Concurrently, [Szirmay-Kalos et al. 2010] presented a volumetric approach for estimating ambient occlusion based only on screen space depth values. Horizon-based AO [Bavoil et al. 2008] aims at finding a maximum horizon angle at which light can reach the sample point. Rays in randomized directions are marched and the maximum elevation angle of these are averaged to estimate the occlusion. Line-sweep AO [Timonen 2013] computes oclussion along a set of principal directions and is efficient due to sharing samples between the screen pixels aligned along these directions.

Statistical approaches aim at improving undersampling issues which arise when balancing performance and quality in SSAO-oriented methods. An example is the use of Summed-area tables (SAT), which are an efficient data structure to compute local averages of the depth values per pixel that can be used to approximate AO [Slomp et al. 2010; Díaz et al. 2010]. However, naively applying SATs leads to strong artifacts at depth discontinuities (halos or overestimations). We build upon these approaches and show how to remove such artifacts using adaptive depth layers.

Multiple depth layers have been used to improve AO and global illumination effects. Vardis et al. [2013] use depth information from different views to improve the estimate of ambient occlusion. Deep screen space [Mara et al. 2014] is a technique to create a depth buffer containing non visible fragments that can be used to compute ambient occlusion and indirect illumination effects. Deep G-Buffers [Nalbach et al. 2014], which contain the first two visible layers in the scene, use the enhanced geometrical information to compute global illumination effects. Altough our layering scheme only uses visible surfaces, applying our method to multiple depth layers could be explored in the future.

3 Statistical Volumetric Obscurance

3.1 Background

AO improves upon standard ambient illumination terms in popular shading models by capturing variations due to subtle shadowing caused by surrounding geometry. The amount of ambient occlusion at a point $\mathbf{x}$ on a surface is related to the ratio of outgoing rays that are able to leave a sample volume as opposed to rays that are being blocked by surrounding geometry (Fig. 2a) [Loos and Sloan 2010]. AO is formally defined as:

$$AO(\mathbf{x},\vec{n}) = \frac{1}{\pi}\int_{\Omega}\rho(d(\mathbf{x},\vec{\omega}))\vec{n}\cdot\vec{\omega}\mathrm{d}\vec{\omega} \ , \tag{1}$$

where $\mathbf{x}$ is the position in the scene, and $\vec{n}$ the normal at $\mathbf{x}$. Here, Ω represents the sample directions, usually a surface aligned hemisphere, and d is the distance to the first intersection.

The fall-off function ρ is used to simulate rays with a limited extent to model only local occlusion. In practice a piecewise constant, linear or quadratic fall-off function is used.

While an exact evaluation of AO based on this definition can be computationally costly, different models exist that can approximate the correct result. In particular, Volumetric Obscurance (VO) [Loos and Sloan 2010] estimates the amount of occlusion using the geometric density within a surrounding sample sphere S:

$$VO(\mathbf{x}) = \frac{1}{Vol(S)}\int_{S}\rho(d(\mathbf{x},s))O(s)\mathrm{d}s \ , \tag{2}$$

where the occupancy function O is 0 if s is inside of the geometry and 1 otherwise and $Vol(S)$ is the volume of the sample sphere S.

The assumption used by the VO model is that if a large portion of the sample sphere is inside a closed geometry, it will be less probable for ambient light rays to reach $\mathbf{x}$. While this intuition does not relate to a physical process, results are similar to AO in practice.

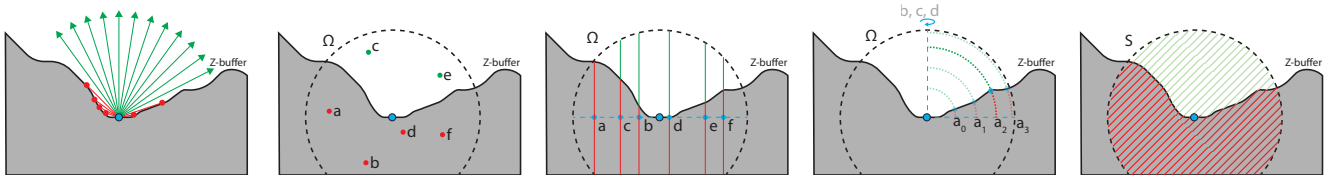
The normal $\vec{n}$, if known, can be used to restrict S to a hemisphere, thus eliminating occlusion caused by geometry below the sample.

3.2 Overview of our method

The method we will present is based on a statistical estimation of volumetric obscurance. For each pixel we define a sampling volume and approximate the amount of space in that volume which is inside the geometry of the scene. We present a model that uses that approximation to compute volumetric obscurance.

Initially we show how to efficiently average the depth of all pixels within a screen aligned sample box centered at each pixel. That sample box may contain pixels representing distant points in 3D space, which will wrongly influence the average depth value. We show how to solve this problem by separating the pixels in different layers and computing a different average per layer.

Finally, we demonstrate how to account for the surface normal by shifting our focus into computing the average of the view space pixel coordinates and projecting the average to the surface normal.



(a) *Screen Space Ambient Occlusion* **(b)** *Point sampling* **(c)** *Line sampling* **(d)** *Horizon-based sampling* **(e)** *Volumetric obscurance*

Figure 2: ***Screen Space Ambient Occlusion:*** *(a) SSAO at a sample point is defined by the ratio of rays that can escape the sample volume. Point sampling (b) and line sampling (c) approximate local ambient occlusion by sampling points within a sample volume. Horizon-based sampling (d) marches in randomized directions to compute the maximum angle at which rays can escape the sample volume. (e) Volumetric obscurance approximates ambient occlusion by computing the percentage of the sample volume that is inside a closed geometry.*

3.3 Our Model

SSAO techniques, such as the one presented in this paper, work solely on the depth map of the rendered image, optionally with an additional normal map. Our solution is based upon the VO model but introduces some important changes that make it more suitable for current graphics hardware and avoids sampling artifacts.

First, we use a sample box (Fig. 3a) instead of a sample sphere to estimate geometric density. The VO assumption is the same, i.e. we determine the percentage of the box which is filled with geometry and assume that it correlates with the amount of occlusion.

Second, we estimate the 3D obscurance function in Eq. (2) with a 2D version as follows. We assume that the geometry in the scene is composed of a single surface whose depth is a continuous function $G : \mathbb{R}^2 \rightarrow \mathbb{R}$ with $G(x,y) = d_{x,y}$ where the values at each pixel position x,y are stored in our Z-buffer. Therefore, every sample point whose depth is greater than what is stored in the Z-buffer is assumed to be occupied, and conversely each sample point whose depth value is less than the corresponding z-value in the depth buffer is deemed unoccupied. The key observation is that we can average the depth function $G(x,y)$ around a sample point and use this value to approximate the occupancy.

The mean value μ of G over a domain V is:

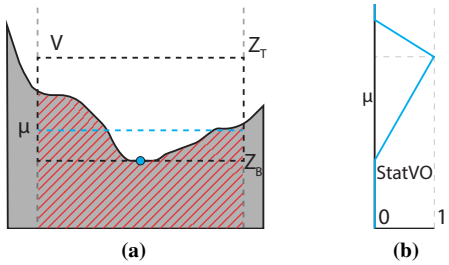
$$\mu = \frac{1}{A_V} \int_V G(x)dx \,, \tag{3}$$

where A_V is the area of the integration domain. The mean value $\mu(x)$ of the screen space depth within a sample box is then used to estimate the geometric occupancy around a pixel (Fig. 3a). Let $z_B(\mathbf{x})$ be the depth value of the bottom face of the sample box and $z_T(\mathbf{x})$ that of the top face. As we are only interested in the relative amount of occupancy, we can cancel A_V out and define our statistical volumetric obscurance (StatVO) model as:

$$StatVO(\mathbf{x}) = \psi\left(\frac{z_B(\mathbf{x}) - \mu(\mathbf{x})}{z_B(\mathbf{x}) - z_T(\mathbf{x})}\right). \tag{4}$$

The function $\psi(x)$ clamps the final value to 0 for negative values of x and behaves linearly for values in $[0,1]$, but will drop off to 0 with a user-defined slope for values greater than 1 (Fig. 3a). In consequence, the VO value results in zero, if the average depth is too far from the surface value, which reflects that the considered sample points fall outside the sample box.

The extent of the sample box in screen-space is computed based on its world position (the screen-space extent of sample boxes should be large nearby and smaller far away). To derive the size of its rectangular projection, we rely on the pixel's linearized depth value. Next, to estimate the occlusion, we need a way to quickly compute the mean depth value inside such screen-aligned rectangle of arbitrary size. To this extent, we make use of *Summed-Area Tables* [Crow 1984]. They allow us to retrieve, at a constant cost, the average in an arbitrary rectangular region around a pixel. In each pixel, they store the sum of all pixels in the upper left quadrant of the texture. The construction can be done using a fast recursive algorithm[Hensley et al. 2005]. To query the average of a rectangular region, the sum of all its interior pixels can be retrieved by combining the values from its four corners.



(a) **(b)**

Figure 3: ***Statistical volumetric obscurance:*** *Our method (a) approach computes the volume integral of a box as an approximation of ambient occlusion. The graph at the right shows how occlusion increases as the average μ rises, after the average leaves the sample box, occlusion falls-off back to zero.*

Approximating the VO via a mean value implies that the fall-off function in Eq. (2) cannot be applied to each sample separately. Furthermore, note that in Eq. (4) the sample box is not restricted in the z coordinate, meaning that sample points outside of the sample box influence μ as well. We would want to reduce the influence of samples that are far from the surface point. In the next section, we explain how to incorporate this idea.

4 Depth Layering

Not applying a fall-off function to each sample when computing volumetric obscurance leads to an overestimation at depth discontinuities, which leads to halos (Fig. 6a). To counteract overestimation, we divide the depth map into m uniformly arranged layers orthogonal to the viewing direction based on the maximum and minimum view-space depths. Each depth pixel is assigned to the layer which overlaps with the corresponding depth value (Fig. 4). Non-

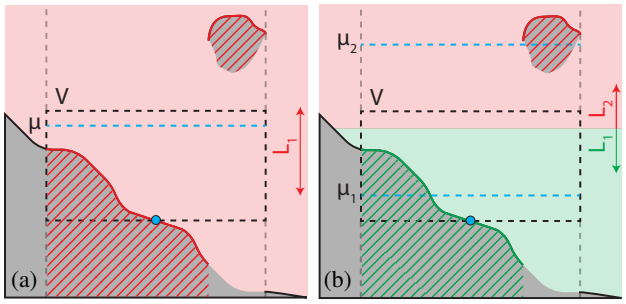


Figure 4: ***Depth Layering:*** *While a single layer will result in a single average over all geometry (a), we can slice our scene in multiple depth layers to obtain averages for each layer separately (b).*

assigned pixels are set to 0. The depth buffer is processed and split among these multiple layers. During the splitting operation, we use an additional (color) channel to keep track of sample/pixel validity, i.e., the channel is one if a depth sample was assigned to the corresponding pixel and zero otherwise. We then generate SAT's for each layer and channel separately. Contributions from each layer L_i overlapping with the sample volume V are weighted by the corresponding sample count n_i (the number of samples assigned to L_i), to account for the missing values, and combined into a final obscurance value $StatVO_{\text{Layered}}(\mathbf{x})$ as follows:

$$StatVO_{\text{Layered}}(\mathbf{x}) = \frac{1}{\sum_{i \in V} n_i(\mathbf{x})} \sum_{i \in V} StatVO_i(\mathbf{x}) n_i(\mathbf{x}) \quad , \qquad (5)$$

where $StatVO_i$ is the statistical volumetric obscurance defined in Eq. (4) computed on layer L_i.

Because of function ψ in Eq. (4), depth slices with depth values significantly different from the surface depth will have no influence on the final computed obscurance. However, the computational effort is linear in the number of layers m and larger scenes require a high number of depth layers, resulting in computation times that are no longer competitive compared to other real-time AO techniques. For example, we found that the Sibenik cathedral requires around 64 layers for good results (Fig. 6e).

5 Adaptive Depth Slicing

To improve upon the linear depth-slicing approach, we propose to use depth layers which adapt to the local geometry. We drew inspiration from higher-dimensional filtering approaches [Gastal and Oliveira 2012] as our technique also builds on a recursive process that partitions the current depth map of a layer into two disjoint sets in each recursion. The intuition behind this step is that as long as pixels with very different depth values are further apart than the screen-space size from the corresponding sample area then they do not influence each other during the obscurance computation.

Our algorithm Fig. 5) works as follows: Initially we have the original depth map and its corresponding SAT. Using this SAT, we can compute the average depth value μ around each pixel, as described in Sec. 3, which amounts to having a smoothed version of the original depth map. We then assign each depth value of the original depth buffer to the upper or lower layer based on its relative depth value compared to μ, hereby often successfully separating locally far and near samples. The intuition behind this approach is that around depth discontinuities, pixels closer to the camera will all have a depth value smaller than the average, and pixels further away will have a depth that is greater than the average. Once each pixel is

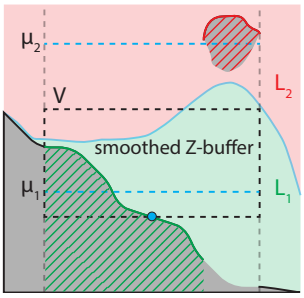
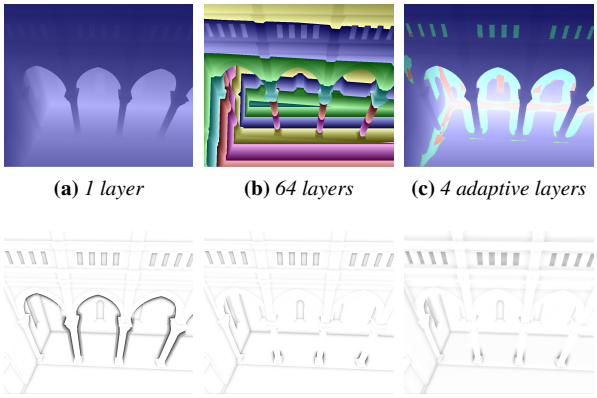


Figure 5: ***Adaptive Depth Slicing:*** *In each recursion a smoothed Z-buffer is constructed, each depth sample is either assigned to the upper (red) or lower layer (green).*



(a) *1 layer* **(b)** *64 layers* **(c)** *4 adaptive layers*

(d) *1 layer, 1 ms* **(e)** *64 layers, 55 ms* **(f)** *4 adaptive layers, 2-3 ms*

Figure 6: ***Comparison****: Using a single layer (a) results in dark halos at depth discontinuities (d). Splitting the scene into multiple layers solves this problem. (b) However, this impacts performance (e). Using adaptive slicing adds additional layers only at depth discontinuities (c). This allows us to eliminate halos and achieve good performance (f).*

assigned to one of the two layers, we compute a new SAT for each layer. As in the uniform depth slicing approach (Sec. 4), we keep track of the amount of active pixels by using an additional channel in these new SATs as well. Once we have the two new SATs, the process can be repeated for each newly created layer in order to further differentiate samples.

During rendering we simply evaluate all layers using Eq. (5). The fall-off function ψ automatically adjusts the influence of each layer, hereby reducing the influence of samples outside the sample box.

The adaptive depth slicing dramatically reduces the number of required layers. As few as four adaptive layers can achieve results that are comparable to the naive 64 uniform layers implementation (Fig. 6e and 6f) for our test scenes.

6 Surface Normal Incorporation

Until now, the sample box was always aligned with the viewing direction. However, when viewing a surface from a grazing angle a bias is introduced into our VO approximation when parts of this surface are hidden in the depth map by pixels closer to the camera (Fig. 7a and b). In Fig. 7a the mean value of layer L_1 (green) is only slightly above the sample position. In Fig. 7b the mean value is

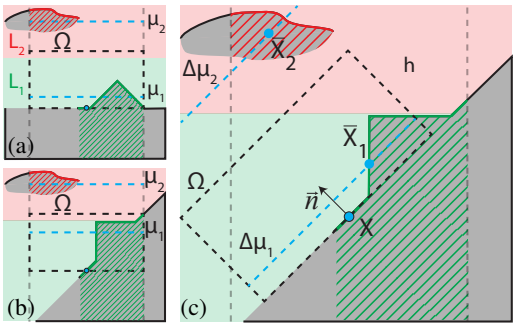


Figure 7: ***Normal Integration:*** *(a) and (b) Considering only the hemisphere/hemibox around a surface point in the viewing direction leads to different occlusion results depending on the slope of the surface. (c) Projection of the mean of all depth samples in 3D space onto the surface normal removes this bias.*

higher since the object in layer L_2 hides a part of the surface underneath. Performing VO computation only for the positive half-space in the direction of the surface normal can enhance the perception of finer scale details [Loos and Sloan 2010]. We make use of the surface normal by extending our approach to 3D and orienting the sample box, so its bottom side is aligned with the surface (Fig. 7c).

After the adaptive layer computation from Sec. 5, we reproject each depth value in each layer into view space to acquire its 3D position. We save the results in RGB maps and compute the corresponding SATs for each. Instead of computing an average depth value, we now compute the average position $\bar{\mathbf{x}}$ of all reprojected depth samples and project it onto the surface normal $\vec{n}$, which conveniently reveals the average height of all samples along the surface normal:

$$\Delta\mu = (\bar{\mathbf{x}} - \mathbf{x}) \cdot \vec{n} \quad , \tag{6}$$

where $\mathbf{x}$ is the surface position. The oriented statistical surface obscurance $StatVO_i$ per layer L_i is then:

$$StatVO_i = \psi\left(\frac{\Delta\mu}{h}\right) \quad , \tag{7}$$

where h is the height of the sample box. Eq. (5) is used to compute the final obscurance value.

Further improvement regarding the approximation quality can be achieved by evaluating four quadrants of the sample region around the pixel separately. The region can be split into four equally-sized parts, for which the result is evaluated independently, by retrieving nine (corners, midpoints, center) instead of four values from the SATs. The final VO value is then computed by averaging the results (Fig. 8). The overall cost increases by roughly 25% to 40%.

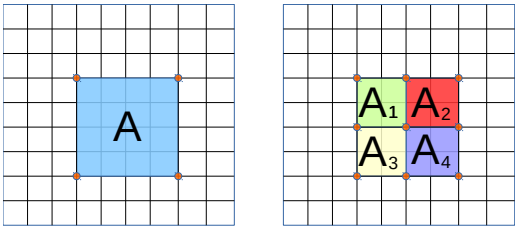
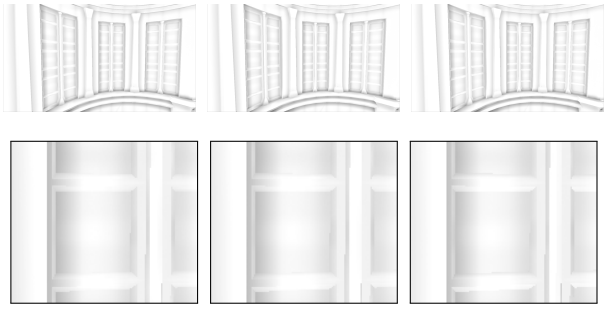


Figure 8: *Left: One region, right: four quadrants. All other images in the paper rely on a single region*



(a) *Full resolution* **(b)** *Half resolution* **(c)** *Quarter resolution*

Figure 9: *Approximate SATs*

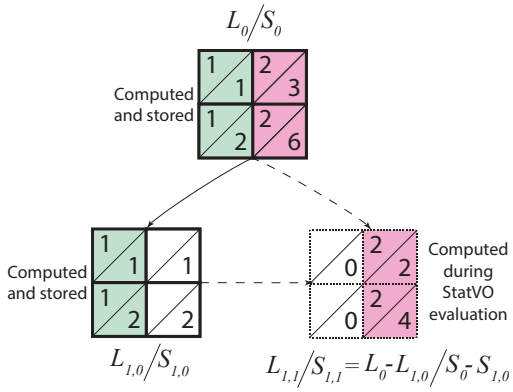


Figure 10: ***Differential SAT Computation:*** *We can spare the computation of one depth layer (L) and one SAT (S) in each partitioning step as they can be reconstructed from their parents and siblings.*

7 Optimizations

We introduce two important performance improvements for our technique; approximate SATs and differential SAT computation.

7.1 Approximate SATs

The most costly computation of our algorithm is the SAT creation. While we experimented with other prefiltering techniques such as Mipmaps, N-Buffers [Décoret 2005] or Y-Maps [Schwarz and Stamminger 2008], SATs provided the highest quality.

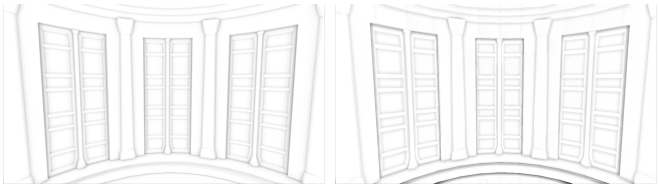
Instead of computing full-resolution SATs, we downsample the input by a factor of 2-4 in both width and height by averaging the depth values, reducing computation times by a factor of 4 to 16 with little impact on quality. We then upscale the low-resolution SATs with linear interpolation to approximate the full resolution input. It is important to note that the additional channel used for sample counting must be handled carefully during downsampling to keep track of the sample count. As linear interpolation is hardware-accelerated, upsampling the SATs is very fast. Fig. 9 shows a comparison between using a full-resolution SAT and downsampled versions. The SAT is queried with sample rectangles, which makes this acceleration suitable for our context. For other sampling strategies, such a solution can be harmful (compare supplementary material).

7.2 Differential SAT Computation

Let L_0 be the (downsampled) original depth map, and $L_{1,0}$ and $L_{1,1}$ be the first two adaptive sublayers resulting from partitioning L_0. Each subdivision of a layer requires the computation of a corresponding SAT S. An important observation here is that while the samples contained in L_0 are distributed among the sublayers $L_{1,0}$ and $L_{1,1}$, their total sum does not change. Thus, subtracting SAT $S_{1,0}$ from S_0 results in $S_{1,1}$ (Fig. 10) removing the need to compute it explicitly. Alg. 1 shows the pseudo-code for the SAT generation routine of four adaptive layers when using differential SATs.

For four adaptive layers, we need to compute only four out of seven SATs explicitly (for more layers the ratio approaches 1:2). During rendering, we compute the value of the missing SATs on-the-fly by subtracting all ancestral and the sibling layer from the root SAT S_0. The pseudo-code in Alg.2 shows how to query the SAT for all four leaf layers $S_{2,0}, S_{2,1}, S_{2,2}$ and $S_{2,3}$ given only $S_0, S_{1,0}, S_{2,0}$ and $S_{2,2}$.

8 Results



(a) *11 line samples (1.47ms)* **(b)** *StatVO (0.93ms)*

Figure 11: ***Comparison to Line Sampling:*** *Both figures evaluate the occlusion at full resolution of 1280×720. Line sampling (a) using 8 samples with a 4×4 randomization kernel with an 8×8 bilateral blur applied. StatVO (b) with 4 adaptive layers with quarter resolution SATs.*

We have implemented our technique using OpenGL/C++. All statistics were measured at 1280×720-pixel resolution on an Intel Core i5 4590 with 8GB of RAM and an NVIDIA GTX 770 graphics card. We implemented the SAT generation algorithm as an OpenGL compute shader [Sellers et al. 2013].

Performance Table 1 shows a detailed performance analysis of our algorithm with four adaptive layers and using full resolution SATs and half resolution in width and height. As the SAT computation is the most costly part of our algorithm, performance increases by a factor of four if width and height are halved. We found that in many scenes, the overall quality loss was small even when reducing the resolution along each axis by a factor of four (Fig. 9). Aggressive downsampling can result in temporal flickering around depth discontinuities due to undersampling when the camera moves.

Algorithm 1 Pseudo-code to compute the needed SATs with 4 adaptive layers.

Given: depth buffer D_0

$S_0 \leftarrow \text{computeSAT}(D_0)$
$D_{1,0} \leftarrow \text{partition}(D_0, S_0)$
$S_{1,0} \leftarrow \text{computeSAT}(D_{1,0})$

$D_{2,0} \leftarrow \text{partition}(D_{1,0}, S_{1,0})$
$D_{2,2} \leftarrow \text{partition}(D_{1,1}, S_{1,1})$

$S_{2,0} \leftarrow \text{computeSAT}(D_{2,0})$
$S_{2,2} \leftarrow \text{computeSAT}(D_{2,2})$

Algorithm 2 Pseudo-code to compute the areas in a two level differential SAT hierarchy

Given: S_0, $S_{1,0}$, $S_{2,0}$, $S_{2,2}$ computed previously

$A_0 \leftarrow \text{sampleSAT}(S_0)$
$A_{1,0} \leftarrow \text{sampleSAT}(S_{1,0})$
$A_{1,1} \leftarrow A_0 - A_{1,0}$
$A_{2,0} \leftarrow \text{sampleSAT}(S_{2,0})$
$A_{2,1} \leftarrow A_{1,0} - A_{2,0}$
$A_{2,2} \leftarrow \text{sampleSAT}(S_{2,2})$
$A_{2,3} \leftarrow A_{1,1} - A_{2,2}$

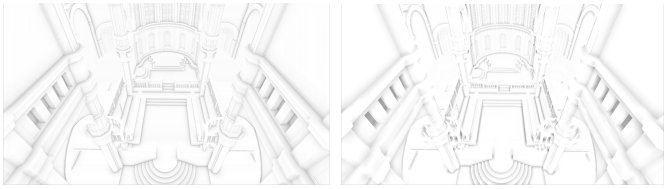
Step	T_{full} (ms)	T_{half} (ms)	Speed-up
Downsample depth buffer	0.13	0.05	×2.6
Compute root SAT	1.40	0.32	×4.4
Compute first level SAT	1.86	0.43	×4.3
Compute second level SATs	3.72	0.84	×4.4
Evaluate StatVO	1.65	0.50	×3.3
Total	8.86	2.14	×4.1

Table 1: ***Performance evaluation:*** *Breakdown of computational cost of our algorithm for full resolution SATs and half resolution SATs when using 4 adaptive layers.*

Memory requirements For our four final SATs, we use four 32bit floating point channels, the first three are used to store view-space coordinates and one is used to mark valid samples in each layer. When computing SATs in full HD, using quarter resolution in width and height, we require a total of 8MB of memory.

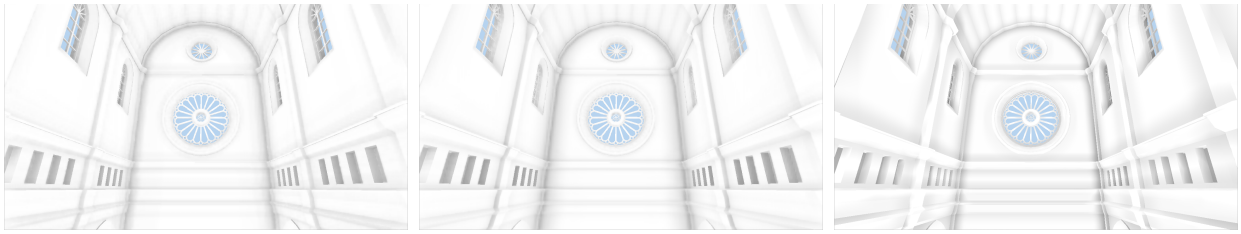
Comparison to other techniques We compare our technique to the classic point and line sampling SSAO techniques, which are the most commonly used [Mittring 2007; Loos and Sloan 2010]. By choosing a very high sample count (256 point samples per pixel), we additionally generated a reference image for volumetric obscurance. In Fig. 13, we show that our technique can generate results that are comparable in quality.

In Fig. 12, we compare our technique to point and line sampling. We chose the number of samples so that all approaches produce visually similar quality. Using a quarter resolution SAT our approach



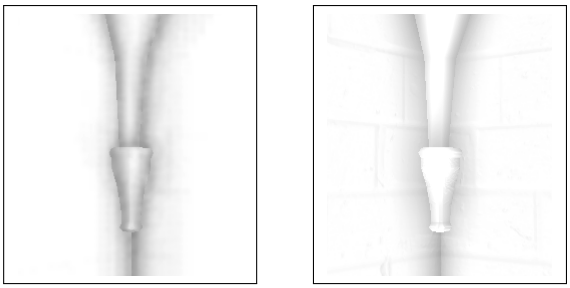
(a) *VO reference* **(b)** *StatVO*

Figure 13: ***Comparison to a Reference VO:*** *(a) Reference from 256 point samples without a randomization kernel or a blur filter. Our method (b) shows comparable results using 4 adaptive layers and full resolution SATs.*



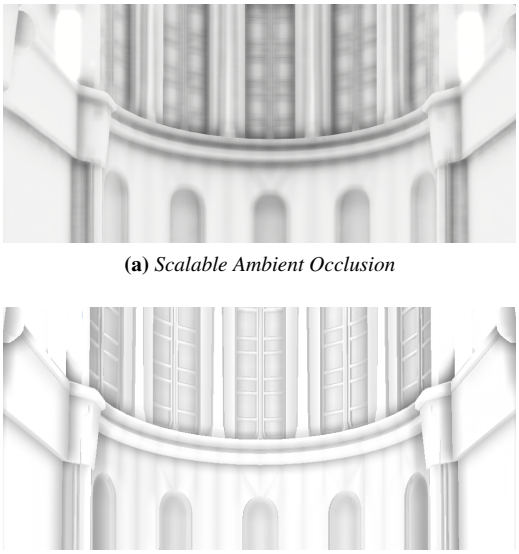
(a) *17 point samples (2.0ms)* **(b)** *12 line samples (1.7ms)* **(c)** *StatVO (1.6ms)*

Figure 12: ***Comparison to Point and Line Sampling:*** *(a) uses a point sampling approach with 17 samples and a 4×4 randomization kernel, occlusion is evaluated at half resolution and a bilateral upsampling with 7× 7 blur kernel is applied. (b) achieves similar results but only uses 12 line samples. In (c) we evaluate occlusion at the full resolution using 4 adaptive layers with quarter resolution SATs.*



(a) *16 point samples (0.77ms)* **(b)** *StatVO (0.65 ms)*

Figure 14: ***Comparison to Point Sampling:****(a) shows a close-up of a point sampling configuration were occlusion is evaluated at half resolution, using 16 samples with a 4×4 randomization kernel. The result is upsampled and an 8×8 bilateral blur filter is applied. (b) StatVO evaluated at full resolution with 4 adaptive layers and quarter resolution SATs.*



(a) *Scalable Ambient Occlusion*

(b) *StatVO*

Figure 15: ***Comparison to Scalable Ambient Occlusion****: (a) shows part of a scene rendered with SAO, (b) shows the same scene rendered with our approach using quarter resolution SATs. Our method does not exhibit blurring despite using downsampled SATs.*

is slightly faster than both. Increasing the AO radius our performance stays the same, whereas the performance of line and point sampling decreases, due to an increase of the required samples and bilateral blur radius, which is mandatory to diminish the increasing undersampling artifacts. This means that our algorithm scales well to higher resolutions compared to point and line sampling (Note that we used a relatively small image resolution of 1024×768 pixels and a similar sample region on a higher resolution image would have to be scaled up).

If only few samples are computed for point or line-sampling (e.g., for very high performance), visible undersampling artifacts appear (Fig. 14). Our approach does not suffer from undersampling and leads to more details, e.g., on the wall.

Fig. 15 shows a comparison of our method to scalable ambient occlusion (SAO) [McGuire et al. 2012], which is currently one of the fastest methods for computing ambient occlusion. SAO exhibits a large amount of blurring due to its usage of mipmapping during sampling and a bilateral blurring step. Although our approach is around 50% slower, it produces crisp results at all depths. Further, our approach does rely on SATs instead of mipmaps, which leads to an additional cost, but proved more stable for animation.

Limitations of screen-space approaches The approach we present is subject to some of the usual limitations associated with screen space AO methods, namely the need for a guard band and the inability to account for surfaces occluded from the view. Due to the statistical nature of our approach, the contribution of thin surfaces parallel to the viewing direction may be underrepresented in the AO computation. Also, as stated previously, downsampling the depth buffer and SATs too aggressively may result in a small amount of temporal flickering. Nonetheless, even when reducing resolution by a factor of 1/4 along each axis, in most areas of the image, there are no visible differences. This result stems from our algorithmic design. An adaptive downsampling strategy could be a promising direction for future research.

9 Conclusion

Statistical Volumetric Obscurance is an alternative to traditional screen-space ambient occlusion, which does not rely on typical sampling. Due to the usage of a sample box, the evaluation of local obscurance is reduced to a simple mean-value computation over the sample area, which is efficiently computed on the GPU using specialized summed-area tables. Previous approaches in this direction suffered from artifacts such as halos. The adaptive depth slicing avoids these and preserves fine-scale features, leading to a quality

similar to previous approaches with many more samples.

For best results the amount of nearby depth discontinuities should be limited. An extreme case, like looking along a row of aligned pillars, breaks this assumption and small halos and dark creases are introduced. Still, our method performs well in these cases (compare accompanying video). Locally adaptive layering would be an interesting future work to address such issues.

As with any SSAO technique, the depth map represents only the visible geometry, whereas important information of the overall scene is lost. Rendering the occluded geometry into the depth layers after they have been created would allow us to incorporate even these occluded parts for more precise results beyond the capabilities of traditional SSAO techniques.

10 Acknowledgements

The work was partially funded by the EU FP7-323567 project Harvest4D, and the Intel VCI at Saarland University. The Sibenik cathedral scene used is a project by Marko Dabrovic and the Sponza atrium scene is freely distributed by Crytek.

References

BAVOIL, L., SAINZ, M., AND DIMITROV, R. 2008. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 Talks*, 22:1–22:1.

BUNNELL, M. 2005. Dynamic ambient occlusion and indirect lighting. *Gpu gems 2*, 2, 223–233.

CROW, F. C. 1984. Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics 18*, 3, 207–212.

DÉCORET, X. 2005. N-buffers for efficient depth map query. In *Computer Graphics Forum*, vol. 24, Wiley Online Library, 393–400.

DÍAZ, J., VÁZQUEZ, P.-P., NAVAZO, I., AND DUGUET, F. 2010. Real-time ambient occlusion and halos with summed area tables. *Computers & Graphics 34*, 4, 337–350.

GASTAL, E. S., AND OLIVEIRA, M. M. 2012. Adaptive manifolds for real-time high-dimensional filtering. *ACM Transactions on Graphics (TOG) 31*, 4, 33.

GROTTEL, S., KRONE, M., SCHARNOWSKI, K., AND ERTL, T. 2012. Object-space ambient occlusion for molecular dynamics. In *Pacific Visualization Symposium (PacificVis), 2012 IEEE*, 209–216.

HENSLEY, J., SCHEUERMANN, T., COOMBE, G., SINGH, M., AND LASTRA, A. 2005. Fast summed-area table generation and its applications. In *Computer Graphics Forum*, vol. 24, Wiley Online Library, 547–555.

HERNELL, F., LJUNG, P., AND YNNERMAN, A. 2010. Local ambient occlusion in direct volume rendering. *Visualization and Computer Graphics, IEEE Transactions on 16*, 4, 548–559.

LANDIS, H. 2002. Production-ready global illumination. *Siggraph course notes 16*, 2002, 11.

LOOS, B. J., AND SLOAN, P.-P. 2010. Volumetric obscurance. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, ACM, 151–156.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image enhancement by unsharp masking the depth buffer. In *ACM SIGGRAPH 2006 Papers*, 1206–1213.

MARA, M., MCGUIRE, M., NOWROUZEZAHRAI, D., AND LUEBKE, D. 2014. Fast global illumination approximations on deep g-buffers. Tech. rep., NVIDIA Corporation.

MCGUIRE, M., MARA, M., AND LUEBKE, D. 2012. Scalable ambient obscurance. In *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics*, EGGH-HPG'12, 97–103.

MCGUIRE, M. 2010. Ambient occlusion volumes. In *Proceedings of High Performance Graphics 2010*.

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

NALBACH, O., RITSCHEL, T., AND SEIDEL, H.-P. 2014. Deep screen space. In *Proceedings of the 18th Meeting of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '14, 79–86.

SCHWARZ, M., AND STAMMINGER, M. 2008. Quality scalability of soft shadow mapping. In *Graphics Interface 2008*, 147–154.

SELGRAD, K., DACHSBACHER, C., MEYER, Q., AND STAMMINGER, M. 2014. Filtering multi-layer shadow maps for accurate soft shadows. In *Computer Graphics Forum*, Wiley Online Library.

SELLERS, G., WRIGHT, R., AND HAEMEL, N. 2013. *OpenGL SuperBible: Comprehensive Tutorial and Reference*, sixth ed. Addison-Wesley Professional.

SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, 73–80.

SLOMP, M., TAMAKI, T., AND KANEDA, K. 2010. Screen-space ambient occlusion through summed-area tables. In *Networking and Computing (ICNC), 2010 First International Conference on*, IEEE, 1–8.

SZIRMAY-KALOS, L., UMENHOFFER, T., TOTH, B., SZECSI, L., AND SBERT, M. 2010. Volumetric ambient occlusion for real-time rendering and games. *Computer Graphics and Applications, IEEE 30*, 2, 70–79.

TIMONEN, V. 2013. Line-sweep ambient obscurance. *Computer Graphics Forum 32*, 4, 97–105.

VARDIS, K., PAPAIOANNOU, G., AND GAITATZES, A. 2013. Multi-view ambient occlusion with importance sampling. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '13, 111–118.

ZHUKOV, S., IONES, A., AND KRONIN, G. 1998. An ambient light illumination model. In *Rendering Techniques '98*. 45–55.