Vision, Modeling, and Visualization (2012)
M. Goesele, T. Grosch, B. Preim, H. Theisel, and K. Toennies (Eds.)

Screen Space Spherical Harmonic Occlusion

S. Herholz[1,2], T. Schairer[1], A. Schilling[1], W. Straßer[1]

[1]University of Tübingen WSI/GRIS, Germany
[2]Stuttgart Media University, Germany

Abstract

In this paper we present a new algorithm for real-time directional occlusion sampling. We combine the real-time capabilities of Screen Space Ambient Occlusion (SSAO) with the Spherical Harmonics (SH) representation of local directional occlusion. SH are well established and used in modern off-line rendering implementations such as PantaRay [PFHA10].
Through our combination we are able to transfer a method for realistic local directional occlusion effects from off-line rendering to dynamic real-time applications. These local occlusion effects react to the environmental lighting situation and lead to dynamic and colored local occlusion shadows while only generating a small computational overhead compared to SSAO. Unlike other real-time directional occlusion algorithms such as Screen Space Direction Occlusion (SSDO) [RGS09] our occlusion sampling is separated from the actual lighting process and therefore can be easily integrated in existing SH lighting methods such as Irradiance Volumes [GSHG98]. We furthermore extend our algorithm to include first bounce indirect illumination effects.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

1. Introduction

The major goal in rendering is to solve the rendering equation introduced by Kajiya [Kaj86]. Especially when it comes to the simulation of global illumination effects solving the integral over the incoming radiance and computing the visibility of the upper hemisphere of a point is a major task. A complete solution of this integral for each point in a scene is extremely complex and time consuming. This is why modern rendering techniques make approximations to get a good result in an acceptable time. In off-line rendering for stills and animated movies, methods such as stochastic raytracing, point based global illumination (PBGI) [Chr10], and ambient occlusion (AO) are used to generate realistic global illumination effects. Especially the use of AO has become very popular in recent years.

1.1. Ambient Occlusion

AO was first introduced by Millers [Mil94]. It assumes that local shading effects only depend on the neighboring geometry of a point and therefore can be represented by a scalar factor describing the percentage of occlusion by local geometry. This concept makes the calculation of AO independent from the irradiance calculation. Local occlusion shadows are added to a scene by multiplying the calculated irradiance by the AO factor. Langer and Bülthoff [LB00] have shown that the use of AO increases the perceived realism of a scene, especially when fine structures are involved. In modern film industry methods based on Monte-Carlo raytracing or PBGI are used to calculate the AO of a point in the scene. Because of the complexity of these methods they are not directly applicable for real-time applications such as games, where the AO of a dynamic scene needs to be updated at every frame. Therefore Mittring [Mit07] and Shanmugan et al. [SA07] developed a screen space sampling algorithm (SSAO) to approximate the AO factor of a point in the scene by just taking the depth information of the surrounding pixels of this point in screen space into account. The current work of McGuire et al. [MOBH11] and Hoang et al. [HL10] optimized the original algorithm in terms of speed, artistic adjustability and reliability. Figure 1 shows an example of the use of a SSAO algorithm.

Even though AO increases the perceived realism of a scene it has one drawback: Due to the fact that AO repre-

DOI: 10.2312/PE/VMV/VMV12/071-078

Figure 1: *An example of a scene rendered using SSAO: (**left**) the diffuse lighting of the scene, (**middle**) the ambient occlusion factor for the scene calculated using SSAO and (**right**) the combined image.*

sents the local occlusion of a point just by a scalar factor, all directional information of the occlusion is disregarded. This leads to static shadows which appear to be grayish even under complex lighting environments. The AO shadows also do not change if the incoming lighting situation of a point changes.

One way to overcome this shortcoming is presented by Sloan et al. [SKS02]. They use spherical harmonics (SH) to represent the directional occlusion of a point in the spherical frequency domain. In combination with a SH representation of the environmental lighting situation as described by Ramamoorthi and Hanrahan [RH01] the irradiance integral $E(\mathbf{p})$ can be evaluated as the scalar product of the coefficient vectors $\mathbf{c}[\text{Lin}]$ and $\mathbf{c}[\text{t}]$ for the incoming light $L_{\text{in}}(\mathbf{s})$ and a SH representation of a transfer function $t(\mathbf{s})$ (e.g. cosine lobe and the directional occlusion) in the frequency domain.

$$E(\mathbf{p}) = \int_S L_{\text{in}}(\mathbf{s}) t(\mathbf{s}) ds \approx \mathbf{c}[\text{Lin}] \cdot \mathbf{c}[\text{t}] \qquad (1)$$

As a result of this combination the local occlusion reacts to the environmental lighting situation, causing colored and dynamic shadows. This method is extensively used by Pantaleoni et al. [PFHA10] for the feature film *Avatar*. Because the calculation of the SH representation of the directional occlusion function is based on raytracing its calculation is not real-time capable and therefore not applicable for interactive dynamic applications.

The recent work of Ritschel et al. [RGS09] and Klehm et al. [KRES11] integrate local directional occlusion effects in interactive real-time applications.
With their "Screen Space Direction Occluion" (SSDO) algorithm Ritschel et al. [RGS09] combine the calculation of local occlusion and lighting in screen space. During the screen space occlusion sampling the incoming light for each unoccluded direction is evaluated by sampling an environment map. Because only light from unoccluded directions is taken into account, the resulting local shadows depend on the direction of the incoming environmental light.
The "Screen Space Bent Normals and Cones" (SSBN) approach described by Klehm et al. [KRES11] combines the concept of bent normals from Landis [Lan02] with the SSAO sampling method. In a separate pass a bent normal is calculated for each point using a screen space sampling algorithm. This bent normal is then used with a set of pre-convolved environment maps during lighting calculation.
While both concepts generate directional occlusion effects such as colored shadows they either need to sample the occlusion during the lighting calculation (SSDO), making it hard to reuse the directional occlusion information in existing pipelines, or need time consuming pre-calculations (SSBN), making it hard to react on changes of the environmental lighting setup.

2. Our Technique

With our new "Screen Space Spherical Harmonic Occlusion" (S3HO) algorithm we combine the benefits of spherical harmonics in the lighting calculation as described by Sloan et al. [SKS02] with the real-time capabilities of SSAO algorithms. This enables us to calculate the SH coefficients for the directional occlusion in real-time for dynamic scenes. When combined with a SH lighting method (as described in Section 2.2) the algorithm generates realistic local occlusion effects when compared to a raytraced directional occlusion implementation. Our algorithm even outperforms SSDO in terms of noise and computational time and SSBN in terms of accuracy in the color of the occlusion shadows.
Because the calculation of the SH occlusion coefficients is independent from the lighting calculation, it can be done in a separate rendering pass. Therefore it should be easy to integrate them in existing SH lighting methods such as described by Tatarchuk [Oat05], which is extensively used by *Disney* in there game related to *Cars 2* [HHE11].
We also present an extension of our algorithm to support screen space indirect illumination effects.

2.1. Screen Space Spherical Harmonic Occlusion Sampling

To determine the SH representation of the local directional occlusion for each pixel all directions of the upper hemisphere have to be checked for occluders till a defined distance is reached. The traditional method as described by Sloan et al. [SKS02] is using a raytracing-based method to evaluate the local visibility. This method is not real-time capable and thus not usable for interactive dynamic scenes.
In our S3HO approach, we calculate the SH coefficients for the local occlusion of each pixel by evaluating the visibility function in screen space. Since we want to use the SH coefficient vector later to evaluate the diffuse irradiance of a point we add a normal oriented cosine lobe to the function. We calculate the SH coefficient $\mathbf{c}[\text{S3HO}]_i$ for the SH basis function y_i by using a Monte-Carlo integration of the product of

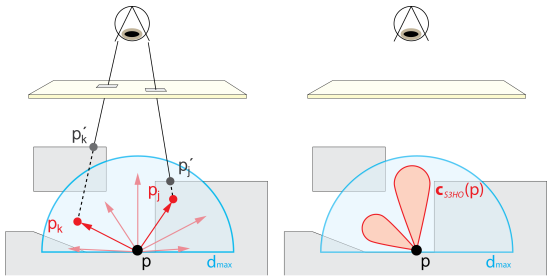

Figure 2: *Graphical representation of the S3HO sampling method: (**left**) example of the screen space occlusion sampling method and (**right**) the SH representation of the occlusion function generated with S3HO.*

the visibility function V_{SS} and the SH basis function.

$$\mathbf{c}[\text{S3HO}]_i(\mathbf{p}) = \frac{4\pi}{2N}\sum_{j=1}^{N} V_{SS}(\mathbf{p},\mathbf{p}_j,d_{\max})y_i(\mathbf{s}_j)(\mathbf{n}\cdot\mathbf{s}_j) \quad (2)$$

To evaluate the visibility of a pixel in screen space, we just need the position and normal of each visible point of the scene represented by a screen pixel. We calculate the SH coefficient $\mathbf{c}[\text{S3HO}]_i$ for the point $\mathbf{p}$ using a set of N sample points $\mathbf{p}_j$, which are uniformly distributed in all directions $\mathbf{s}_j$ of the upper hemisphere of $\mathbf{p}$. Since we want to sample the direction from $\mathbf{p}$ to the border of the local upper hemisphere, the distance of the sampling points and $\mathbf{p}$ is distributed between 0 and $d_{\max}$. The diffuse cosine lobe is added by multiplying the visibility by the scalar product of the surface normal $\mathbf{n}$ at $\mathbf{p}$ and the sampling direction $\mathbf{s}_j$.
The left image in Figure 2 shows how the screen space visibility function V_{SS} evaluates the visibility of a point $\mathbf{p}$ using screen space information. The samples $\mathbf{p}_j$ and $\mathbf{p}_k$ are positions picked from inside the upper hemisphere of $\mathbf{p}$. The points $\mathbf{p'}_j$ and $\mathbf{p'}_k$ are the positions of scene objects at the projected screen space positions of $\mathbf{p}_j$ and $\mathbf{p}_k$. The point $\mathbf{p'}_j$ is closer to the camera as $\mathbf{p}_j$ therefore the direction $\mathbf{p}_j$ is tagged as occluded and $V_{SS}(\mathbf{p},\mathbf{p}_j)$ returns 0. The point $\mathbf{p'}_k$ lies outside the upper hemisphere of $\mathbf{p}$. Therefore, the direction $\mathbf{p}_k$ is tagged as unoccluded and $V_{SS}(\mathbf{p},\mathbf{p}_k)$ returns 1, although it is closer to the camera than $\mathbf{p}_k$. For a smooth local occlusion effect a visibility increasing with the distance d such as $(1-\frac{1}{1+\lambda*d})$ can be used to weight the visibility for occluded directions. Lambda is used to control the strength of the effect.
The right image in Figure 2 shows the SH directional occlusion function generated using this sampling method. The two red cones represent the unoccluded directions from where light can access the point $\mathbf{p}$.

2.2. Diffuse Spherical Harmonics Lighting

We calculate the diffuse lighting of a scene by using a deferred shading pass and the concept of spherical harmonics lighting as described in Section 1.1. When using SH for evaluating the lighting of a scene the irradiance integral is calculated by using the scalar product of the SH coefficient vectors of the incoming radiance and a transfer function. We calculate the diffuse irradiance E_{diff} of a point $\mathbf{p}$ by using the incoming radiance $\mathbf{c}[\text{Lin}]$ and our previously calculated directional occlusion function combined with the cosine lobe.

$$E_{\text{diff}}(\mathbf{p}) = \mathbf{c}[\text{Lin}](\mathbf{p})\cdot\mathbf{c}[\text{S3HO}](\mathbf{p}) \quad (3)$$

Due to the additivity of light and spherical harmonics coefficients the SH coefficient vector $\mathbf{c}[\text{Lin}]$ for the incoming light function for each pixel at the scene position $\mathbf{p}$ can be calculated by accumulating the SH coefficient vectors for the environment/ambient light function $\mathbf{c}[\text{env}]$ and the SH coefficient vectors $\mathbf{c}[\text{l}](\mathbf{p})$ for each light source l with respect to $\mathbf{p}$.

$$\mathbf{c}[\text{Lin}](\mathbf{p}) = \mathbf{c}[\text{env}] + \sum_{l=1}^{n}\mathbf{c}[\text{l}](\mathbf{p}) \quad (4)$$

Since the environmental/ambient light is typically static its SH representation is the same for each point in the scene. It can be generated from an ambient color, an environment map, or from a HDR light probe such as presented by Debevec [Deb05].
We continue the calculation of $\mathbf{c}[\text{Lin}]$ by evaluating the incoming radiance arriving at $\mathbf{p}$ from each light source. Similar to Sloan [Slo08] we are using zonal harmonics and the SH representation of the Hanning function to generate an SH light function from the direction of the light source relative to $\mathbf{p}$.
Figure 3 shows two examples for this lighting procedure where two points get lit by the same incoming light function but with different directional occlusion functions. The incoming light function $\mathbf{c}[\text{Lin}]$ is projected on a sphere where the color on the surface of the sphere is equal to the light coming from this direction. The scenario contains two light sources: one orange one from the upper left and one blue one from the upper right. The SH representation of directional occlusion of the points $\mathbf{p}_1$ and $\mathbf{p}_2$ are projected on a sphere where the distance to the origin is equal to the value of the occlusion function in this direction, forming red cones in the unoccluded directions. In the left image $\mathbf{p}_1$ is occluded from the right. Through the SH occlusion function only light from the upper left can access $\mathbf{p}_1$, so only orange light can light $\mathbf{p}_1$ and its color gets orange. In the right image the situation is vice versa.

2.3. Screen Space Indirect Illumination

In their paper Sloan et al. [SKS02] present a method that uses the SH transfer function to not only represent the visibility and diffuse reflection but also integrates indirect illumination effects. In combination with Ritschel and colleagues [RGS09] idea of using the identified screen space occluders as indirect illumination sender, we extended the

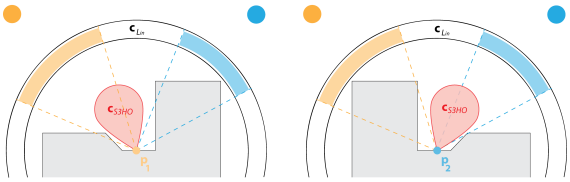

Figure 3: *Examples of spherical harmonic lighting using an occlusion transfer function. Both images are lit by the same incoming light setting but have different occlusion situations: (**left**)* $\mathbf{p}_1$ *is occluded from the right and (**right**)* $\mathbf{p}_2$ *is occluded from the left. This different occlusion situations lead to different irradiance colours for* $\mathbf{p}_1$ *(orange) and* $\mathbf{p}_2$ *(blue).*

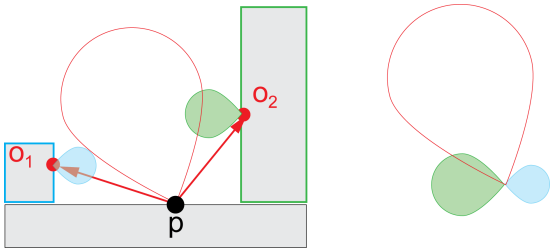

Figure 4: *Example of the integration of 1-bounce indirect illumination effect in the SH occlusion function. (**Left**) the SH occlusion function for* $\mathbf{p}$ *and the SH transfer functions for the indirect illumination from screen space occluders. (**Right**) illustration of the accumulated occlusion function.*

SH transfer function of our S3HO algorithm to take indirect illumination into account. The new transfer function $\mathbf{c}[\text{S3HOGI}]$ for a point $\mathbf{p}$ is an additive combination of the SH coefficient vector $\mathbf{c}[\text{S3HO}](\mathbf{p})$ and an SH coefficient vector for the indirect illumination $\mathbf{c}[\text{SSGI}](\mathbf{p})$. Because the indirect illumination can be of different colors than the material at point $\mathbf{p}$ the $\mathbf{c}[\text{S3HOGI}]$ and $\mathbf{c}[\text{SSGI}]$ consist of three separate coefficient vectors, one for each RGB color. The SH coefficient vectors for the indirect illumination are generated during the occlusion sampling process. For each of the m samples identified as an occluder $\mathbf{o}_i$ is treated as an indirect illumination sender. Therefore the form factor f between $\mathbf{p}$ and $\mathbf{o}_i$ is calculated. The indirect illumination which $\mathbf{p}$ receives from $\mathbf{o}_i$ is represented by the product of the diffuse SH transfer function and the diffuse BRDF ($\frac{\rho}{\pi}$) of $\mathbf{o}_i$. The diffuse transfer function of $\mathbf{o}_i$ is a SH coefficient vector $\mathbf{c}[\cos]$ of a cosine lobe oriented to the surface normal of $\mathbf{o}_i$.

$$\mathbf{c}[\text{SSGI}](\mathbf{p}) = \sum_{i=1}^{m} f(\mathbf{p},\mathbf{o}_i)\frac{\rho}{\pi}\mathbf{c}[\cos] \tag{5}$$

Instead of approximating an unoccluded transfer function for each occluder it is possible to reuse the calculated occlusion information of an occluder from the previous frame, as done by Ritschel and colleagues [RGS09]. However, since the colored occlusion transfer function contains three SH-coefficient vectors for each RGB color this would increase amount of data needed to be accessed for each occluder, which would affect the execution time of the algorithm adversely.

Figure 4 shows an example of the generation of a SH coefficient vector for the occlusion and indirect illumination for a point $\mathbf{p}$. The left image shows the calculated SH occlusion function $\mathbf{c}[\text{S3HO}]$ of point $\mathbf{p}$. During the screen space sampling process the points $\mathbf{o}_1$ and $\mathbf{o}_2$ are identified as occluders of $\mathbf{p}$. The surface of $\mathbf{o}_1$ is blue and a blue SH representation of a cosine lobe is generated and scaled using the BRDF of $\mathbf{o}_1$ and the form-factor between $\mathbf{p}$ and $\mathbf{o}_1$. The surface of $\mathbf{o}_2$ is green which leads to a green scaled SH representation of a cosine lobe.
The right image shows the combination of the SH occlusion function for $\mathbf{p}$ and SH transfer functions for the indirect illumination from $\mathbf{o}_1$ and $\mathbf{o}_2$. When light comes from the left, green indirect illumination from $\mathbf{o}_2$ is added to the evaluated irradiance for $\mathbf{p}$. Otherwise, when light comes from the right blue indirect illumination is added from $\mathbf{o}_1$.

3. Implementation

Our algorithm is implemented on the GPU using OpenGL and GLSL fragment shaders. Each step is implemented in a separate deferred rendering pass. To access the screen space information we are using a GBuffer holding the position and normal in camera space for each visible point of the scene, with both information stored in a `Float16` render target.
The spherical function of the local directional occlusion is stored in a SH representation using four bands, which leads to SH coefficient vectors with 16 coefficients. Ramamoorthi et al. [RH01] showed that for diffuse lighting and without taking the local occlusion into account three SH bands are sufficient for representing the diffuse transfer function. Due to the integration of the local occlusion more high frequency components are introduced in the transfer function and we increased the number of bands to four. This is a good choice since the 16 coefficients can compactly be stored in two `Float32` rendering targets using Nvidia's `pack_2half` function to store two `Float16` values in one `Float32` value.
To evaluate the SH basis functions during the sampling process in real-time on the GPU we use the real spherical harmonics representation listed at the end of Sloan's article [Slo08].

Sampling For the sampling process itself we are using a set of 32 samples with directions uniformly distributed over

the surface of the upper hemisphere. Since 32 samples are not enough to fully sample all directions of the upper hemisphere the approximated SH representation of the local occlusion can be incorrect. To reduce the error of the estimated local occlusion we are using a 4x4 interleaved sampling pattern as introduced by Keller et al. [KH01]. The high frequency noise that is introduced by the interleaved sampling pattern is reduced by using a geometry aware 8x8 bilateral filter ([Tom98]). For the bilateral filter we are using two weighting functions: One for taking the distance and the other for taking the orientation of the surface of the neighboring pixel into account.
The result of the sampling can also be improved by using ray-marching, where a direction is sampled on multiple lengths as described by Ritschel et al. [RGS09].

4. Results

As test environment we used a Core i7 2.8 GHz machine with 8GB of RAM and a Nvidia GTX 470 with 1GB of RAM. All images were rendered at a resolution of 1280x720. At this resolution it takes the S3HO algorithm 16.5ms to calculate and store the SH representation of the local directional occlusion when using 16 samples and two ray marching steps. An equivalent SSAO implementation using the same sampling pattern took 14.9ms to calculate the AO factor without any directional information.
With S3HO the directional information of the local occlusion is preserved and in combination with the spherical harmonics lighting technique described in Section 2.2 this information is taken into account during the diffuse lighting calculation. This leads to local occlusion shadows, which consider the current light situation from the unoccluded directions.
Figure 5 shows a scene consisting of boxes, where diffuse lighting is calculated using S3HO. The scene is lit by two light sources: a pink one from the left and a turquoise one from the right. The figures on the middle left and right show SH representations of the directional occlusion for the points $\mathbf{p}_1$ and $\mathbf{p}_2$ calculated with S3HO. The occlusion situation at these points is similar to the ones in Figure 3. The figure at the middle center shows the SH representation of the incoming light projected on a unit sphere where the color of the surface is equal to the incoming light function in this direction. Because the local occlusion at $\mathbf{p}_1$ is blocking all light coming from the right $\mathbf{p}_1$ gets mainly lit by the pink light source and the local shadow is pinkish. At $\mathbf{p}_2$ the local occlusion is blocking all light coming from the left and so $\mathbf{p}_2$ is mainly lit by the turquoise light source and the local shadows at $\mathbf{p}_2$ get greenish.

In bottom image of Figure 5 the positions of the light sources are switched. Due to the change of the incoming illumination function the color of the local occlusion shadows changed. This shows that the images generated with S3HO react dynamically to changes of the surrounding light environment.

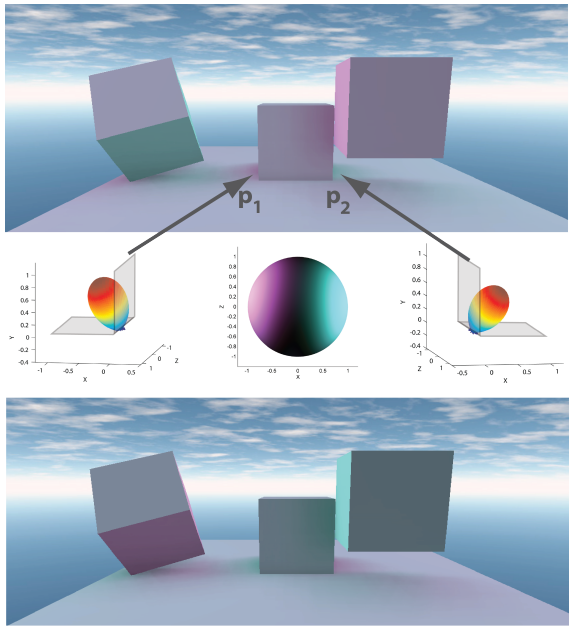

Figure 5: *The diffuse lighting of a scene using S3HO: (**top/bottom**) the rendered scene consisting of 3 boxes and (**middle**) the SH representation for the occlusion (**middle left/right**) and for the incoming light (**middle center**). On the bottom image the positions of the light sources are switched.*

In Figure 6 we compare our algorithm against a standard SSAO implementation using the same sampling method as S3HO. The scene shows some rectangular boxes as they would appear in a skyscraper scene. It is lit by a cloudy blue skylight environment map with bright yellow sun light coming from the right. The left image shows the scene illuminated using S3HO and the right one is using SSAO. The occlusion shadows in the right image all have the same color and equally distributed around the boxes. In the left S3HO image on the other hand the occlusion shadows are more oriented to the left (left closeup), because the light from the bright sun is blocked and the shadows to the right are lit up by the bright sun light that they seem to vanish (right closeup). Because at the left shadows the sunlight is blocked the blue skylight illuminated these shadows and they get a blue tint. These shadows are also darker then the SSAO shadows because in the S3HO image the bright sunlight is not taken into account in the illumination evaluation. In the SSAO image the sunlight is taken into account and then multiplied by the occlusion factor leading to brighter occlusion shadows.

In the following we compare the results of our algorithm with the recent work of Ritschel et al. [RGS09] (SSDO) and

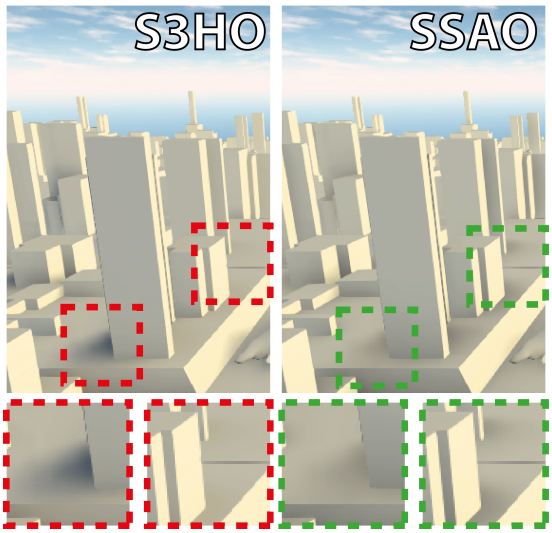

Figure 6: *Comparison between S3HO (**left/red**) and SSAO (**right/green**). The closeups on the bottom visualize the differences between both methods.*

Klehm et al. [KRES11] (SSBN). To focus the comparison on the basic concept of each algorithm, the same number of samples and ray marching steps are used. Figure 7 depicts the results of the different algorithms compared to a raytraced ground truth result (RTDO). The scene is illuminated by a filtered version of the *Grace Cathedral* HDR-lightprobe. All objects in the scene consist of a white lambertian material. A detailed view of the differences of the algorithms is presented in Figure 8. Except for the RTDO image two ray marching steps and 32 samples are used for the screen space sampling process.

Because the lightprobe contains a number of strong light sources in different colors the colors of the local occlusion shadows in the RTDO image vary according to their orientation. The occlusion shadows generated with S3HO and SSDO have the same colors as the ones in the RTDO image. Compared to S3HO the SSDO occlusion shadows contain more noise. That is because the left over high frequencies in the pre-filtered HDR-lightprobe are still to high to be accurately sampled with 32 samples. A way to overcome this problem would be to use a stronger filter on the lightprobe. Finding the right filter size depends on the frequencies in the lightprobe and the number of samples used. A too large filter kernel would smooth the light probe too much and directional occlusion effects would get lost. While a too small kernel would lead to leftover noise as seen in Figure 7. The smoothing of the lightprobe is done implicitly by SSBN and S3HO through the pre-convolution step and the SH projection of the lightprobe. Note that SSDO does not tend to so strong noise if LDR-lightprobes are used.

In contrast the local occlusion shadows generated by SSBN are as smooth as the occlusion shadows generated by S3HO. Only the color of the occlusion shadows is not always the same as in the S3HO or in the original RTDO image. The reason for that is, that the bent normals and bent cones can lead to false assumption of the incoming light. For example, at a 90 degree corner (as seen in the 2nd row of Figure 8) the generated bent normals of both surfaces will point in the same direction. Because both surfaces have the same amount of local occlusion the same size of a cone around the bent normal will be used to gather the incoming light. The result is that both surfaces are illuminated by the same color, even if the actual lighting situation would contribute different colored light from the original unoccluded normal direction of the surfaces.
The fourth row in Figure 8 shows how the different algorithms react to a decrease of the number of samples used for the occlusion estimation. For the presented images 8 samples and 2 ray marching steps where used. The noise generated by the reduced number of samples is rather low in the S3HO ans SSBN images. The occlusion shadows of these two algorithms are still smooth and keep the same color. The noise in the SSDO image on the other side increase more in the SSDO image, while also the colors of the incoming light suffers from the under-sampling of the HDR-lightprobe.

Table 4 shows the computational times needed by the dif-

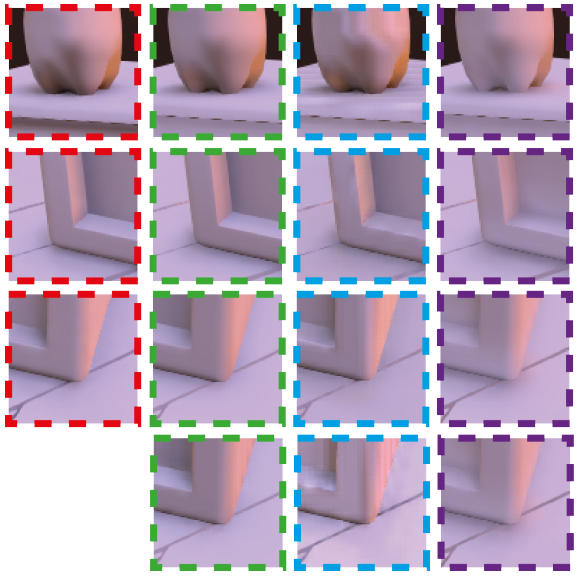

Figure 8: *Closeups on detailed views of the comparison images from Figure 7: RTDO (**red**), S3HO (**green**), SSDO (**blue**) and SSBN (**purple**). Row 1-3 uses 32 samples and two ray marching steps. In the 4th row 8 samples and two ray marching steps are used.*

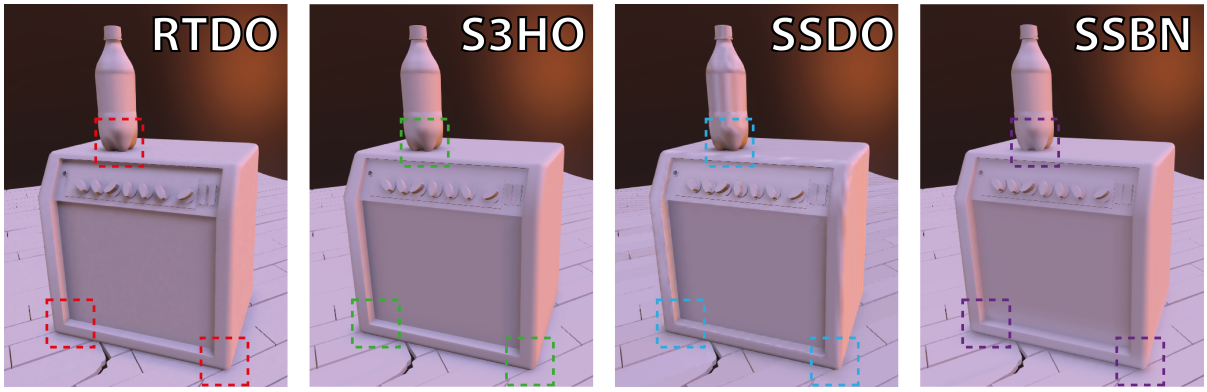

Figure 7: *Comparison between different directional occlusion algorithms: RTDO (**red**), S3HO (**green**), SSDO (**blue**) and SSBN (**purple**). Detailed closeups are presented in Figure 8.*

ferent algorithms compared to a reference SSAO implementation using two ray marching steps and three different numbers of samples.

#Samples	*SSAO*	*S3HO*	*SSDO*	*SSBN*
32	26.3	27.8	40.7	27.0
16	13.3	14.1	20.3	13.6
8	6.9	7.4	10.1	6.9

Table 1: *The computational times in ms needed for the different algorithms to calculate local occlusion effects.*

From the listed algorithms SSDO requires the most computation time because for each unoccluded sampling direction additional textures accesses are preformed to read from the environment map. The computation time for SSBN and S3HO is minimal higher than for SSAO. Because SSBN only needs to calculate and store the mean unoccluded bent normal it is slight faster then S3HO. This little overhead is negligible if we take the increase of realism of the calculated lighting achieved by these algorithms into account.

In Section 2.3, we described an extension of our S3HO algorithm to integrate screen space global illumination for first indirect lighting effects. Figure 9 shows the Crytek *Sponza* scene rendered with S3HO and its global illumination extension. The top image depicts the textured scene with first diffuse bounced indirect illumination. The center and bottom images show the calculated untextured diffuse radiance of the scene calculated by S3HO (center) and SSDO (bottom). These images are extracted before the bilateral filter is used to reduce the sampling noise. Closeups are used to highlight this noise. Our extension generates similar first bounce global illumination effects as SSDO, with less noise than SSDO. The higher amount of noise in the SSDO image (green closeup) is caused by the under sampling of the environment map during the direct illumination sampling. The execution time for the extension of our S3HO algorithm increased to 59ms since each sample needs two more GBuffer texture accesses for the normal and diffuse color of each occluder sample. Also the storage space needed for the coefficient vector increases by a factor of three, since a separate vector for each RGB color channel is needed.

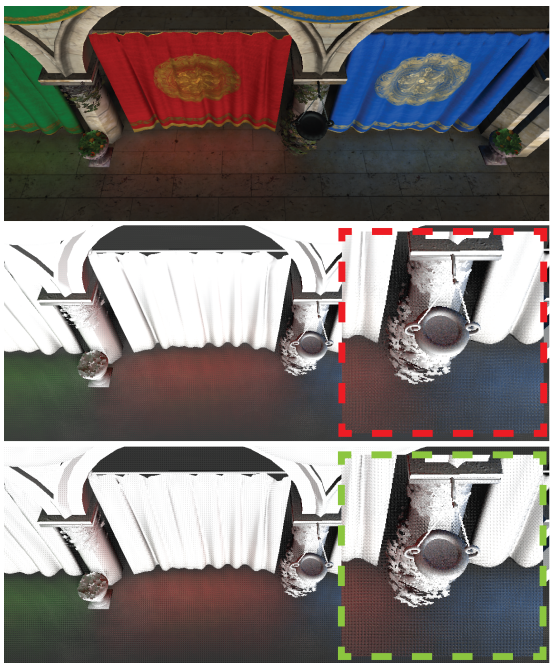

Figure 9: *The Crytek Sponza scene lit with the indirect illumination extension of S3HO: (**top**) the lit scene with texture and the calculated untextured diffuse radiance, before the use of the bilateral filter. The **center** image is the result of S3HO and the **bottom** image of SSDO. The closeups highlight the sampling noise of the different algorithms.*

5. Limitations

In the following we want to discuss the limitations of S3HO. Because the algorithm is based on screen space occlusion sampling it shares the limitations of other SSAO based sampling algorithms like SSDO and SSBN (e.g. objects outside the view frustum or surfaces not visible in screen space are not taken into account). Some of these limitation can be resolved by adding depth peeling or multi-view sampling as described by Ritschel et al. [RGS09].
Because the SH coefficients are additive they can be interpolated the same way as an AO factor. Therefore the same methods to improve SSAO can also be used to improve S3HO (e.g. the method described by Mattausch et al. [MSW10] for better temporal coherence).
Another limitation of S3HO is caused by the use of only four SH bands to represent the local directional occlusion and the incoming light function. High frequency occlusion situations can not be accurately represented and fine occlusion effects can be missed or smoothed out through the low frequency representation.

6. Conclusion

We presented a new algorithm for calculating and storing the local directional occlusion of a dynamic scene in real-time by using screen space information. The directional occlusion information is stored efficiently by using SH coefficient vectors. In combination with a SH lighting algorithm the rendered result contains colored and dynamic local occlusion shadows, which plausibly react to changes in the environmental lighting situation. The computational overhead needed to calculate the SH representation of the local occlusion is minimal compared to other screen space occlusion algorithms such as SSAO or SSBN, while the accuracy of the approximated lighting increased significantly. We extended our algorithm to integrate screen space indirect illumination effects. Because the sampling of the directional occlusion is separated from the lighting calculation it should be easy to integrate our algorithm in existing rendering pipelines, which are based on a SH lighting technique such as *Irradiance Volumes* [Tat05], [GSHG98] or *Light Propagation Volumes* [Kap09].
For future work it would be useful to store the SH coefficient vectors in a more compact way, so it would be possible to use more SH bands for better approximation of the local occlusion function.

7. Acknowledgment

We want to thank the anonymous reviewers for their valuable feedback. A special thank goes to Prof. Dr. Lensch for proofreading this paper.

References

[Chr10] CHRISTENSEN P.: Point-based global illumination for movie production. *ACM SIGGRAPH* (2010). 1

[Deb05] DEBEVEC P.: Image-based lighting. In *ACM SIGGRAPH Courses* (2005). 3

[GSHG98] GREGER G., SHIRLEY P., HUBBARD P. M., GREENBERG D. P.: The irradiance volume. *IEEE Comput. Graph. Appl. 18*, 2 (Mar. 1998), 32–43. 1, 8

[HHE11] HALL C., HALL R., EDWARDS D.: Rendering in Cars 2. 2

[HL10] HOANG T.-D., LOW K.-L.: Multi-resolution screen-space ambient occlusion. In *ACM Symposium on Virtual Reality Software and Technology* (2010), pp. 101–102. 1

[Kaj86] KAJIYA J.: The rendering equation. *ACM SIGGRAPH Computer Graphics 20*, 4 (1986), 143–150. 1

[Kap09] KAPLANYAN A.: Light Propagation Volumes in CryEngine 3. In *ACM SIGGRAPH Course* (2009). 8

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Eurographics Workshop on Rendering Techniques* (2001), pp. 269–276. 5

[KRES11] KLEHM O., RITSCHEL T., EISEMANN E., SEIDEL H.: Bent Normals and Cones in Screen-space. In *Vision, Modeling and Visualization* (2011). 2, 6

[Lan02] LANDIS H.: Production-ready global illumination. *ACMM SIGGRAPH Course* (2002). 2

[LB00] LANGER M., BÜLTHOFF H.: Depth discrimination from shading under diffuse lighting. *Perception 29*, 6 (2000), 649–660. 1

[Mil94] MILLER G.: Efficient algorithms for local and global accessibility shading. In *ACM SIGGRAPH* (1994), pp. 319–326. 1

[Mit07] MITTRING M.: Finding next gen: Cryengine 2. In *ACM SIGGRAPH courses* (2007), pp. 97–121. 1

[MOBH11] MCGUIRE M., OSMAN B., BUKOWSKI M., HENNESSY P.: The alchemy screen-space ambient obscurance algorithm. In *ACM High Performance Graphics* (2011), pp. 25–32. 1

[MSW10] MATTAUSCH O., SCHERZER D., WIMMER M.: High-quality screen-space ambient occlusion using temporal coherence, 2010. 8

[Oat05] OAT C.: Irradiance volumes for games. *Presentation at Game Developers Conference* (2005). 2

[PFHA10] PANTALEONI J., FASCIONE L., HILL M., AILA T.: Pantaray: fast ray-traced occlusion caching of massive scenes. In *ACM SIGGRAPH* (2010), pp. 37:1–37:10. 1, 2

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.: Approximating dynamic global illumination in image space. In *ACM Interactive 3D graphics and games* (2009), pp. 75–82. 1, 2, 3, 4, 5, 8

[RH01] RAMAMOORTHI R., HANRAHAN P.: An Efficient Representation for Irradiance Environment Maps. In *ACM SIGGRAPH* (2001), pp. 497–500. 2, 4

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *ACM Interactive 3D graphics and games* (2007), pp. 73–80. 1

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM SIGGRAPH* (2002), pp. 527–536. 2, 3

[Slo08] SLOAN P.: Stupid spherical harmonics (sh) tricks. In *Game Developers Conference* (2008), pp. 320–321. 3, 4

[Tat05] TATARCHUK N.: Irradiance Volumes for Games. 8

[Tom98] TOMASI C.: Bilateral filtering for gray and color images. *Computer Vision* (1998). 5