Modern Methods of Realistic Lighting in Real Time

István Szentandrási
Supervised by: Adam Herout

Faculty of Information Technology
Brno University of Technology
Brno / Czech Republic

Abstract

Physically plausible illumination in real-time is often achieved using approximations. Recent methods approximate global illumination in the screen space by exploiting the capabilities of modern graphics cards. Two of these techniques, screen-space ambient occlusion and screen-space directional occlusion, are described in this work. Screen-space directional occlusion is a generalized version of screen-space ambient occlusion. It supports one indirect bounce of diffuse light and depends on the direction of incoming light. The main goal of this project is to further experiment with these methods and improve them. For a uniform distribution of the sampling points, the Halton sequence is used. In order to reduce the noise, geometry-aware bilateral filtering is presented. Methods are further sped up by computing them in a lower resolution, and they are restored to full resolution using joint bilateral upsampling in order to create the final image.

Keywords: global illumination, ambient occlusion, screen-space ambient occlusion, screen-space directional occlusion, halton sequence, bilateral filtering

1 Introduction

Computing global illumination in real-time has been and still is a major challenge in computer graphics. Due to the complexity of light transport and some material properties, real-time frame rates can only be achieved at the cost of trade-offs and rough approximations. Perceptually among the most important optical phenomena belong soft shadows and indirect lighting. There have been many attempts to simulate either of these in real time. A handful of these attempts are based on ambient occlusion (AO) [13], which is very popular in the film industry as well as in games. The main advantage of these techniques lies in their speed and simple implementation.

As in every approximation, ambient occlusion has some limitations, too. The basic method [13] displays darkening of cavities; however, it does not take into account the direction and intensity of light coming from light sources or environmental maps. A better method has been introduced by Ritschell et al. [11] called screen-space directional occlusion (SSDO). SSDO provides more realistic illumination: it accounts for the direction of the incoming light and supports a single indirect bounce of light.

The aim of this work is to experiment with these methods and improve them. The improvements are mostly focused around speeding up the methods. This work is structured as follows. In Section 2 we describe the present state of methods used in the area. Then Section 3 we will present screen-space ambient occlusion (SSAO) and SSDO.

Section 4 presents possible modifications and optimizations to the SSDO method: speed optimizations, an alternative method for uniform distribution of sampling points using Halton sequence and using a variable number of sampling points for each pixel based on local scene complexity. Section 5 summarizes the achieved results.

2 Related Work

Ray tracing and radiosity have always been the two basic approaches to approximate physically correct lighting. However, both methods require massive computations. In the case of radiosity, it solves a system of equations. In the case of ray tracing, the major slowing factors are the number of object-ray intersections and visibility tests. There are many techniques to accelerate these methods, such as final gathering, irradiance caching, sparse sampling, advanced space division structures, etc. Even with the recent growth in processor speeds and the introduction of GPGPU solutions, these methods are still too slow for interactive applications. Rendering on GPU remains the superior solution for real-time rendering. It still has a major lead, especially for dynamic scenes. This barrier caused the development of alternative methods which did not try to simulate physically correct lighting. They just aim to give perceptually convincing approximations.

Since the introduction of ambient occlusion [2][13], it has been widely adopted both in gaming and the film industry. Ambient occlusion computes the visibility of the hemisphere at each point of the scene. The method is often calculated by casting rays in every direction over the hemisphere using Monte Carlo sampling. The calculated factor is used to modulate ambient lighting, just as the name suggests. [4]

Casting rays in every point still requires too much computing power, so a few alternative methods were introduced. These methods compute AO less accurately in order to achieve higher frame-rates. Instead of computing occlusion over surfaces in 3D, these methods usually approximate AO in the screen space [12][8][1][5]. SSAO is very popular due to its simplicity and speed. It does not require any additional data and can be applied as a post-process to the scene.

Ambient occlusion is just a rough approximation of general light transport. It does not take into account any directional information or other more expensive illumination effects (interreflections, caustics, subsurface scattering). A different family of techniques, the precomputed radiance transfer (PRT) [4], does support the aforementioned features. On the other hand, PRT algorithms typically assume static scenes, distant lights or environment maps.

Screen-space directional occlusion (SSDO) [11] tries to combine the speed and simplicity of SSAO methods with directional information of lighting and near field indirect color bleeding.

In order to avoid some limitations of screen space ambient occlusion, a hybrid method was introduced by Reinbothe et al. [10]. This method works in 3D space by voxelization of the scene, calculating occlusion based on this information and finally using bilateral filtering in the screen space to smooth the shadows.

A completely different approach was taken by Kaplanyan et al. [6]. They approximate indirect illumination in fully dynamic scenes using cascaded light propagation volumes. This method supports single bounce illumination with occlusion, but it can be extended to support multiple bounces and to handle participating media.

These techniques show that in order to generate visually convincing images, no physically precise computations are needed. Simple approximations using soft shadows, ambient occlusion, and optionally, a single bounce of indirect light can give out acceptable results even in real-time.

3 Real-Time Global Illumination Techniques

There are many techniques to approximate global illumination in real time. We focused on methods that can be computed in a postprocessing step so as to improve overall quality of the rendered images.

3.1 Screen-Space Ambient Occlusion

Screen-space ambient occlusion (SSAO) is a coarse approximation of ambient occlusion that works in the screen space in order to achieve real-time frame rates. The idea behind SSAO is to reuse the z-buffer data, which was already computed during the rendering of the scene. This approach is based on sampling the surrounding pixels combined with simple depth comparisons. Based on these results, an average visibility value can be computed. This visibility property is used as a darkening factor for cavities and corners in the scene. This way SSAO can be computed in one pass as a post-process over the image.

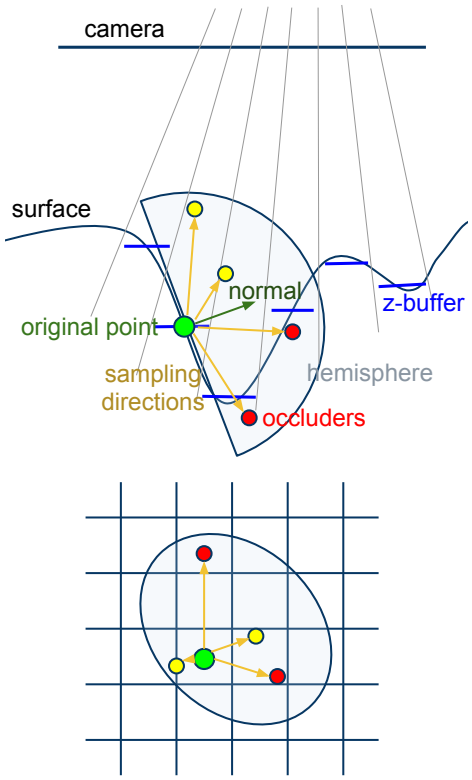


Figure 1: SSAO principle. An area around the pixel is sampled in 2D (bottom image). Based on the normal, sampling points are generated inside the hemisphere (top image). The generated depths for each sampling point are compared to the depth of the appropriate pixel to determine if the pixel corresponds to an occluder object (yellow (light) and red (dark) points. Based on the results of the depth comparisons, the pixel's intensity will be darkened (on the image to half, since half of the sampling points are below scene surface).

Additional information can be used to achieve more precision and hence better looking results. The normals for each pixel could be a good choice. Without taking the normal into account and just randomly generating points in a close proximity of the pixel may cause darkening of unwanted parts of the scene, such as planes that are almost parallel with the view direction. Using the normal of the pixel sampling directions outside the hemisphere can be filtered out (Figure 1). However, random sampling may cause problems in the image which have to be sorted out. So as to avoid artifacts, the distribution of the generated random samples over the hemisphere is needed to be as uniform as possible. Also, the noise caused by non-uniform sampling should be smoothed.

Usually in SSAO methods, the problem of generating

samples randomly with a uniform distribution is solved by precomputing a few uniformly distributed random directions in a sphere. Variation to the sampling points is added using random normals to reflect the directions. These new directions are then optionally reversed to be in the hemisphere around the normal of the pixel. In this project we used the Halton sequences with appropriately chosen bases.

Computing ambient occlusion from simple depth values has some drawbacks, too. First the precision of SSAO is highly dependent on the size of the scene. The bigger the scene, the coarser the object shape approximation. This can lead to unwanted effects, like darkening the whole silhouette of an object.

The second problem is caused by the limited area that is sampled for occluders. Let us consider two objects close to each other in the scene in 3D. Using a classical ambient occlusion method, such as using ray tracing, the two objects are darkened. However, using SSAO the distance between the projected positions of the objects might be larger then the sampled area. As a consequence SSAO will not detect any occluders and the objects will not be darkened.

3.2 Screen-Space Directional Occlusion

SSDO is a fast approximation of global illumination. It works in the screen space, takes into account the direction of the light, and is able to handle one indirect bounce of diffuse light [11]. In order to compute light transport, SSDO uses the 3D positions and normals of each pixel in the screen space as input. The output is created in two passes. In the first pass, the direct illumination is computed. In the second pass, the indirect bounce of light is computed using the data from the previous pass.

3.2.1 Direct Illumination Using Directional Occlusion

While standard SSAO methods use only the positions and normals of each pixel, SSDO also takes into account the direction of the incoming light. The amount of directional light is computed as follows for each point $\mathbf{P}$ and normal $\mathbf{N}$:

$$L_{dir}(\mathbf{P}) = \frac{1}{\pi}\int_{\Omega}\frac{\rho}{\pi}L_{in}(\omega)V(\omega)(\mathbf{N}\cdot\omega)\,\mathrm{d}\omega, \qquad (1)$$

where $\frac{\rho}{\pi}$ is the diffuse BRDF, L_{in} is the incoming radiance from direction ω in the hemisphere Ω and V is the visibility test. When using Monte Carlo sampling the integral is replaced by a sum of K samples each covering a solid angle of $\Delta\omega = 2\pi/K$:

$$L_{dir}(\mathbf{P}) = \sum_{i=1}^{K}\frac{\rho}{\pi}L_{in}(\omega_i)V(\omega_i)(\mathbf{N}\cdot\omega_i)\Delta\omega. \qquad (2)$$

This method assumes that L_{in} can be efficiently computed from environment maps or point lights. Similar to SSAO, so as to avoid ray-tracing, the occluders are approximated in the screen space. The difference is that while in SSAO the samples are generated in 2D in image space, SSDO uses sample points generated in the hemisphere in 3D using the normal and the 3D position of the pixel. The sample points are then backprojected into the image space in order to determine if within the given direction is an occluder or not. This way SSDO does not suffer from the problem mentioned in SSAO, that is: objects further away in the screen-space, but closer in 3D in the scene may cause occlusion.

The sampling of the hemisphere is done in the following way: for every generated direction ω_i and a random step $r_i \in [0..\, r_{max}]$, the position of the sampling points is computed as $\mathbf{P} + r_i\omega_i$. The generated points are located in the hemisphere with a center of $\mathbf{P}$ and oriented around the normal $\mathbf{N}$. The depth of the backprojected sampling points is compared with the values from the original z-buffer. If the depth value from the original z-buffer is smaller than the depth of the sampling point, the sampling point is below the surface. The light from this direction is blocked by an occluder. Otherwise, the light has a clear path from this direction and the incoming radiance can be computed.

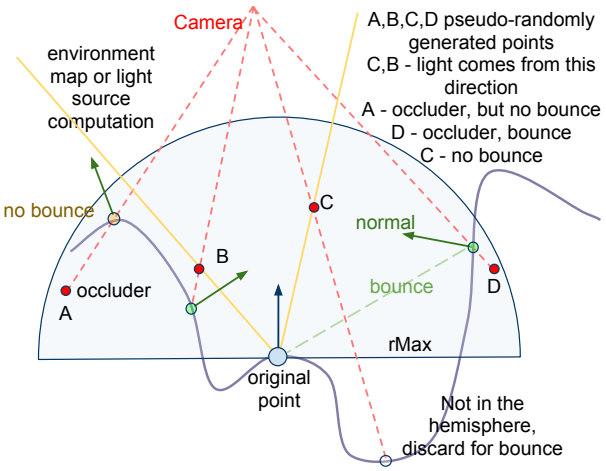


Figure 2: SSDO principle. Random samples are generated in 3D in the hemisphere. Samples under the surface are classified as occluders. Otherwise, the incoming radiance can be computed from the direction defined by the sampling point. The sampling points classified as occluders are projected on the surface. Based on the color and position of the pixels on the surface, indirect bounces are computed.

The method is demonstrated in Figure 2 for four sampling points A, B, C, D (red dots). The sampling points are generated randomly with a uniform distribution over the hemisphere with a random step from the original point and then backprojected onto the image. Now that the image space coordinates are known, the 3D coordinates can be computed or read from a frame buffer (green, orange and black dots). These points are again projected into the image in order to get the distance from the camera. If the sampling point is further than the appropriate point on a surface in the scene, the sampling point is classified as an

occluder (A and D points). Otherwise, the illumination can be computed from the direction defined by the point and the origin (B and C points). The direction is shown by the yellow line.

3.2.2 Indirect Bounce

Since the 3D position and normal are available for each sampling point projected to the surface, they can be used to get one indirect bounce of light from the given directions. In the original paper, only the points on the surface projected from sampling points classified as occluders were taken into account (A and D points). For each of these pixels the computed directional light intensity and the corresponding pixel color from the direct illumination pass is used as the base for indirect light. In order to calculate the indirect radiance sent to the origin, these pixels are treated as small patches oriented around the normal. Using the sender normal, back facing patches can be filtered out (for example, for sampling point A).

The equation for the additional incoming indirect radiance for a point $\mathbf{P}$:

$$L_{ind}(\mathbf{P}) = \frac{1}{\pi}\int_{\Omega}\frac{\rho}{\pi}L_{dir}(\mathbf{P}_{\omega})(1-V(\omega))\cdot \frac{A_s(\mathbf{N}\cdot\omega)(\mathbf{N}_{\omega}\cdot(-\omega))}{|\mathbf{P}-\mathbf{P}_{\omega}|^2}\,\mathrm{d}\omega, \quad (3)$$

where $\mathbf{P}_{\omega}$ and $\mathbf{N}_{\omega}$ are the point and normal from point and normal buffer, each corresponding to a sampling point taken from the hemisphere in direction ω. A_s is the area associated with the sender patch. This equation respects the mutual orientation of the surfaces based on the normals $((\mathbf{N}\cdot\omega)(\mathbf{N}_{\omega}\cdot(-\omega)))$ and that the intensity of the incoming radiance decreases quadratically with the distance of the surfaces.

The modified version of the equations for K samples is then:

$$L_{ind}(\mathbf{P}) = \sum_{i=1}^{K}\frac{\rho}{\pi}L_i(1-V(\omega_i))\frac{A_s(\mathbf{N}\cdot\omega_i)(\mathbf{N}_i\cdot(-\omega_i))}{|\mathbf{P}-\mathbf{P}_i|^2}\Delta\omega, \quad (4)$$

For the initial value for A_s, the base circle is subdivided into K regions, each covering $A_s = \pi r_{max}^2/K$. This value can also be used as a parameter to control the strength of the color bleeding manually.

In the example above (Figure 2), points A and D are the only occluders. After projecting these points onto the surface from the cameras viewpoint, the information about the normal, position and color are also available. The projected point for A has a back facing normal, so it will not contribute to the final bounce. The patch for sampling point D, on the other hand, will qualify as a sender of indirect light towards point $\mathbf{P}$.

4 Modifications and Implementation Notes

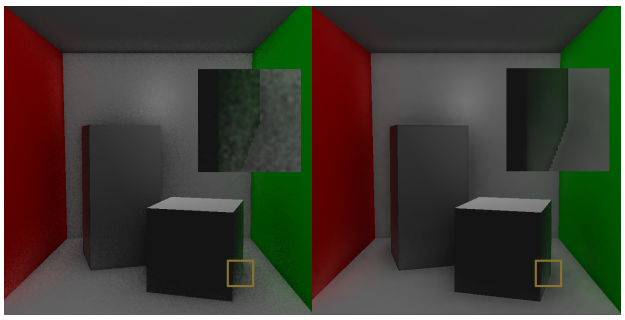
The aim of this project is to experiment with the methods described in the previous chapter and to potentially improve them. We will next describe three modifications and improvements we experimented with and which are potentially beneficial.

4.1 SSDO for Directional and Point Lights

When SSDO is computed for scenes with directional and point lights, several modifications are possible. Computing the pixels intensity in real implementations based on the equation (2), due to the random or pseudo-random sampling of the directions in the hemisphere, causes noise even on plain surfaces with no occluders nearby. So as to avoid this, SSDO can be computed as follows: when rendering the scene before computing SSDO to get the material, normal, depth information, the overall intensity of the pixels can be computed using the appropriate lighting model. The overall intensity should be equal to the value computed using equation (1) without the visibility function. When computing direct illumination (first step) in SSDO, instead of computing the intensity for the pixels, a modulation factor can be computed for each pixel. This modulation factor is the ratio of the intensity computed with the visibility function and without it using equation (2). The modulation factor can be used to darken the diffuse and/or the ambient component of the overall intensity when creating the final image. The modulation factor can either be per channel or a single value based on scene and lighting properties. When there are multiple lights in the scene with radically different colors, a darkening factor for each color channel is best. However, for most scenes a single modulation factor is sufficient, since the lights are usually white or the distance between them is large enough, so they have only a minor effect on objects close to other lights. A single modulation factor also helps to reduce memory usage.

This modulation factor can be further processed. In order to reduce noise, smoothing can be used. A simple Gaussian blur is not enough in this case. The usage of geometric information is necessary to prevent bleeding of values over edges and between distant pixels in 3D, but close in 2D. A geometry-aware filtering, like a modified bilateral filtering on the base of normals and depth value, is suitable. Firstly, this method was used by Reinbothe et al. [10]. So as to approximate the results of a full bilateral filtering, they separated the calculations into a vertical and horizontal pass. A combination of the results of these two one-dimensional filters improves the frame rates significantly and still provides an acceptable quality.

Since this modulation factor is actually a generalized version of the darkening factor computed with ambient occlusion, it changes, similar to the ambient occlusion fac-



(a) No filtering, 10 sampling points - 74FPS

(b) 2x subsampling (3x3 upsampling kernel, 20 sampling points per pixel and 11x11 bilateral filtering - 78 FPS

Figure 3: Comparison of non-filtered and smoothed results. Regions with the yellow border are shown in more detail on the right side of the images.

tor, slowly over spatial space on surfaces. This permits one to compute it in lower resolutions. As a consequence the method could be sped up radically; on the other side, the result should be upsampled correctly. For this joint bilateral upsampling [7] can be used with the same modifications as for smoothing to honor geometry properties (Figure 3).

More speed improvement can be achieved by merging the two steps of SSDO together. This means that the source color for the bounces is taken from the scene rendered without SSDO darkening (the same image, that is later darkened by SSDO). The modulation factor and indirect bounces are stored and filtered separately. In order to avoid pixels being too bright in darkened corners, the indirect bounces should also be darkened using the modulation factor.

4.2 Sampling and the Halton Sequence

In both SSAO and SSDO, Monte Carlo sampling is used to approximate the correct solutions. Absolute random distribution of samples in Monte Carlo methods can cause problems; one of the worst of these is clumping (when for small number of random numbers, the variance between the values is low). So as to decrease the effect of clumping in samples, quasi-Monte Carlo methods eliminate the randomness completely. Samples are deterministically computed to achieve a stochastic distribution as close to the uniform distribution as possible.

In order to describe how much the point distribution of a given method derivates from an ideal solution, a measure called discrepancy is used. Quasi-Monte Carlo methods try to minimize this discrepancy. There are several low-discrepancy sequences that are used for generating sampling points: Hammersley, Halton, Sobol, Niederreiter, etc. [4]

The Halton sequence generation is based on the radical inverse function applied to an integer i. This integer can be expressed in a base b with terms a_j:

$$i = \sum_{j=0}^{\infty} a_j(i) b^j. \tag{5}$$

The radical inverse function is computed by reflecting the resulting digit sequence around the decimal point:

$$\Phi_b(i) = \sum_{j=0}^{\infty} a_j(i) b^{-j-1}. \tag{6}$$

For generating multi-dimensional low-discrepancy sequences, a different radical-inverse sequence is used in each dimension. The ith point in the sequence is given as:

$$x_i = (\Phi_{b_1}(i), \Phi_{b_2}(i), \ldots, \Phi_{b_d}(i)), \tag{7}$$

where the bases b_j are relatively prime and d is the dimension of the sequence.

An intuitive explanation of the uniformness of the Halton sequence is as follows. Let us consider the generated floating point numbers as digit sequences in the given base expressed as strings. Before generating strings of length $m+1$, all the strings of length m are produced. This means that before generating a new point on an interval, all intervals of size $b-m$ will be visited first. This fact also suggests some kind of periodicity in similarity of the generated values. For example, let us consider a Halton sequence with base 2: $\Phi_2(50) = 0.296875, \Phi_2(50+16) = 0.2578125, \Phi_2(50+32) = 0.2890625, \Phi_2(50+64) = 0.3046875$. This periodicity can be expressed [3]:

$$\left|(\Phi_b(i) - \Phi_b(i+mN_g)\right| < \frac{1}{b^k}; N_g = lb^k, l > 0, k \geq 0, \tag{8}$$

where l, m, k are integers, and N_g is the period to generate similar sampling points. For multidimensional Halton sequences, the least common multiple of the periods for each dimension is used as N_g. This property is demonstrated in Figure 4.

This periodicity can be exploited to control the number of sampling points and still have a quasi uniform distribution. For example, with a period 10, 10 samples can be used for pixels where the low number of sampling points does not matter. For pixels, where more sampling points are needed to get a smoother result, $20, 30, \ldots, k \cdot 10; k \in N$ sampling points can be used (see Section 4.3). Another possibility for future improvements may be, for example, in the case of occlusion detection to filter out directions, where an occluder was already found.

4.3 Preprocessing

In SSDO, the same amount of sampling points is generated for every pixel. Setting this amount higher means better results, but it will cause a performance drop. Generating more samples for planes which do not have any

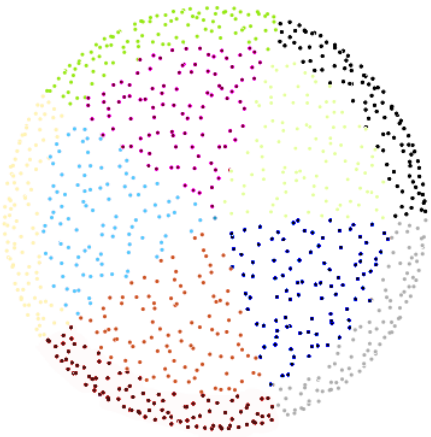


Figure 4: Projected view of the hemisphere - Halton sequences with bases 2, 5 and period 10. Each pixel is colored based on the index in the sequence and the period. Every 10th pixel has the same color.

occluders nearby is a waste of computing power. The solution would be to generate more sampling points for pixels which are potentially occluded and less for pixels where the probability of finding an occluder is low. The number of occluders is usually higher at parts of the image where the normals or the depth values differ significantly. So as to get areas where the higher number of sampling points should be generated, a preprocessing step could be added to SSDO calculations. In this preprocessing step, a weight is calculated to define the number of sampling points to be used. Areas where the changes in normal and depth values are bigger are given a higher weight. Planes, on the other hand, will have much smaller weight since the normals are constant for pixels on the same plane.

Figure 5: Preprocessed images with different sizes of filtered images. The darker regions represent higher number of sampling points. The blue color just shows the computed SSDO darkening factor.

In order to calculate this weight, a simple filtering using the normal and depth values can be used to get the gradient magnitude. This value can be used to determine the number of sampling points. So as to get wider areas around discontinuities and in order to speed up the filtering, it can be computed in much lower resolutions. Optionally combining results from different resolutions should give more precise results of areas where SSDO values may change more. Using just one resolution is not the best solution. A more advanced method was used by Nichols et al. [9] to get locations where higher resolution was required for computing image space radiosity.

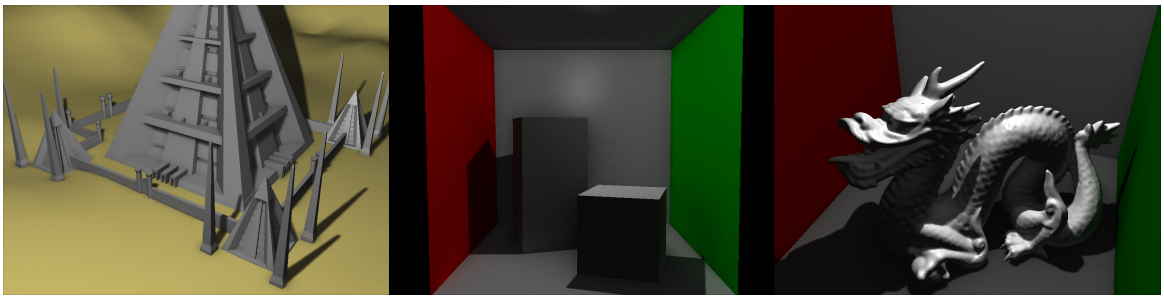
5 Results

With both methods (SSAO and SSDO) we were able to produce realistic images in real-time (Figure 7). The results of the achieved frame rates are summarized in Table 1. The methods were tested on three scenes with resolution 1024x768 (Figure 6). These numbers are just exemplary. The speed of these methods depends also to a great extent on the resolution, graphics hardware, as well as the degree of required smoothing.

SSAO was naturally the fastest. The additional computation and texture reads to get bounces for SSDO makes it 25% slower with the same number of sampling points. However, to get good results for SSDO with a larger hemisphere radius, many sampling directions are needed. From the measured frame-rates it is clear that the limiting factor for SSDO computation is the speed of the fragment shader and the number of texture lookups per pixel. These are the areas that should be more optimized in the future.

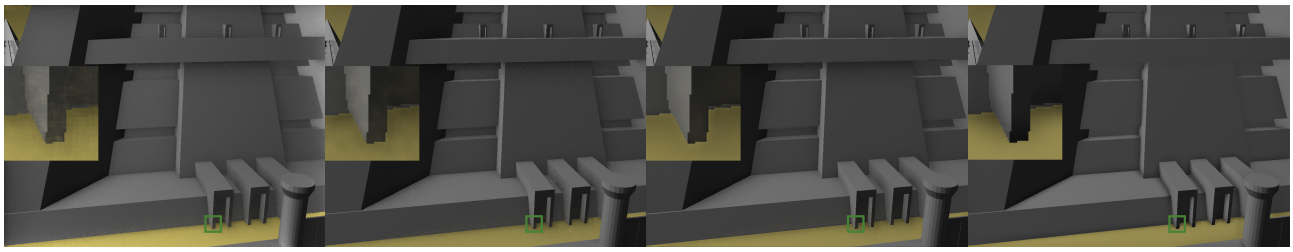
Due to the fact that SSDO is computed in the screen space, a few problems arise. The lack of complete knowledge of 3D causes occluders not visible from the camera's point of view to be discarded, hence making the results highly view-dependent. For more complex scenes even a small change in camera position may reveal parts of the scene previously hidden and cause new shadows and indirect bounces. The authors of the original paper suggested using depth peeling for partly solving this problem. Storing multiple depth values for each point in multiple passes gives more information on the scene structure; however, it makes the speed of the technique dependent on scene complexity. It also further slows down occlusion computations by forcing it to read values from multiple buffers and calculating the depth test for each. Alternatively, the authors also suggest using multiple viewpoints. This in theory could give better results than depth peeling, but correct positioning of the cameras may vary based on the scene type.

A comparison of SSAO and SSDO can be seen in Figure 8. The difference in the two techniques can be easily observed on the darkening factor around the tail of the dragon. While SSAO darkens only the silhouette, the tail in SSDO darkens the wall close to it in 3D. The next difference is when the light position is changed. While SSAO remains static with moving light, the shadows caused by occlusion in SSDO move a little in the opposite direction. This nice feature of SSDO with the addition of the bounce gives us much more believable results than SSAO.



(a) Pyramids (b) Cornell box (c) Dragon

Figure 6: Scenes rendered using SSDO with variable number of sampling points based on preprocessing (20, 30, 40 or 50), 2x subsampling, 11x11 kernel for bilateral smoothing and 3x3 upsampling filter. (Dragon model is from the The Stanford 3D Scanning Repository)



(a) SSDO 2 step (b) SSDO 10 points (c) SSDO with preprocessing and subsampling (d) SSAO

Figure 7: SSDO 2 step – separate step for computing direct and indirect illumination; SSDO 10 points – SSDO computed in one step using modulation factor; SSDO with preprocessing and subsampling – SSDO computed in one step with 2x subsampling, 3x3 kernel for upsampling and variable number of sampling points per pixel based on a preprocessing step (20, 30, 40 or 50 samples). For all methods 11x11 smoothing kernel was used. Regions with the green border are shown in more detail on the left side of the images.

[FPS]	Number of sampling points	Pyramids (15804 faces)	Cornell box (30 faces)	Dragon (201037 faces)
SSAO no subsampling	10	72	77	72
SSDO 2 steps	10	42	37	33
SSDO no subsampling	10	51	46	42
SSAO	20	115	127	113
SSDO	20	90.5	83	74
SSDO	50	67	55	49
SSDO with preprocessing	20-50	74	67	60

Table 1: Frame rates for each scene and method with 11x11 kernel for bilateral smoothing and 3x3 kernel for upsampling, if not specified otherwise. In the second column are the number of sampling points used per pixel. For the 'SSDO with preprocessing' row variable number of sampling points (20, 30, 40 or 50) was used based on a preprocessing step. For the rest of the rows constant number of sampling points was used. The 'SSDO 2 steps' row represents the original two-step algorithm with a separate step for computing direct and indirect illumination. For the other SSDO rows indirect illumination and modulation factor was computed in one pass. For testing an ATI Radeon™ HD 4850 GPU was used.

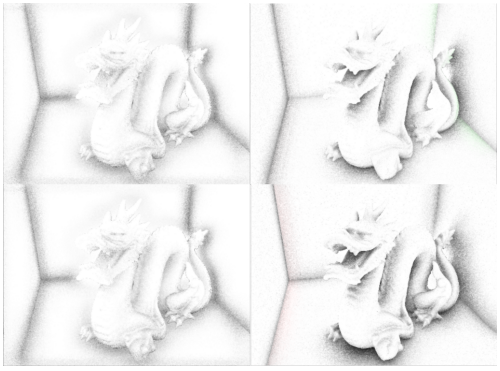


Figure 8: On the left SSAO; on the right SSDO with different light positions for the rows. The light is placed in front of the dragon, closer to the left wall in the top row of images and behind the dragon, closer to the right wall for the bottom row images.

6 Conclusions

Screen-space ambient occlusion is a very fast approximation of ambient occlusion, but it has some limitations. Screen-space directional occlusion includes two generalizations that add directional occlusion and diffuse indirect bounces. Both extensions improve realism considerably for a minor computational cost.

In this paper, a few experiments were made to the screen-space directional occlusion. Sample point generation based on the Halton sequence is an easy way to get uniform distribution of the sampling points. The periodic properties of the Halton sequence can also be used to potentially further optimize the method. This was exploited to generate a variable amount of sampling points, but still have a pseudo-uniform distribution. So as to reduce noise, a modified version of bilateral filtering was used, which took into account the geometry information as well to avoid color bleeding over edges and between objects. The SSAO and SSDO methods were computed in lower resolutions to speed up the method. For upsampling to the original resolution, joint bilateral upsampling was used to honor geometry properties.

In the future, more experiments can be made to SSAO and SSDO accompanied with more comprehensive testing based on the controllable features of each method. In order to get a clearer picture where these methods stand performance-wise, a few other methods could be explored, too.

References

[1] Louis Bavoil, Miguel Sainz, and Rouslan Dimitrov. Image-space horizon-based ambient occlusion. In *ACM SIGGRAPH 2008 talks*, SIGGRAPH '08, page 22:1, New York, NY, USA, 2008. ACM.

[2] Robert L. Cook and Kenneth E. Torrance. A reflectance model for computer graphics. *SIGGRAPH Comput. Graph.*, 15:307–316, August 1981.

[3] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Proceedings of the 13th Eurographics workshop on Rendering*, EGRW '02, pages 25–36, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[4] P. Dutré, K. Bala, and P. Bekaert. *Advanced global illumination*. Ak Peters Series. AK Peters, 2006.

[5] Dominic Filion and Rob McNaughton. Effects & techniques. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 133–164, New York, NY, USA, 2008. ACM.

[6] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 99–107, New York, NY, USA, 2010. ACM.

[7] Johannes Kopf, Michael F. Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[8] Martin Mittring. Finding next gen: Cryengine 2. In *ACM SIGGRAPH 2007 courses*, SIGGRAPH '07, pages 97–121, New York, NY, USA, 2007. ACM.

[9] G. Nichols, J. Shopf, and C. Wyman. Hierarchical image-space radiosity for interactive global illumination. page 11411149, 2009.

[10] C. Reinbothe, T. Boubekeur, and M. Alexa. Hybrid ambient occlusion. EUROGRAPHICS 2009 Areas Papers, 2009.

[11] Tobias Ritschel, Thorsten Grosch, and Hans-Peter Seidel. Approximating dynamic global illumination in image space. In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, I3D '09, pages 75–82, New York, NY, USA, 2009. ACM.

[12] Perumaal Shanmugam and Okan Arikan. Hardware accelerated ambient occlusion techniques on gpus. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, I3D '07, pages 73–80, New York, NY, USA, 2007. ACM.

[13] S. Zhukov, A. Inoes, and G. Kronin. An ambient light illumination model. In *proceedings of the Eurographics Workshop in Vienna, Austria*, Rendering Techniques '98, pages 45–56, Springer, 1998.