Approximating Dynamic Global Illumination in Image Space

Tobias Ritschel Thorsten Grosch Hans-Peter Seidel
MPI Informatik

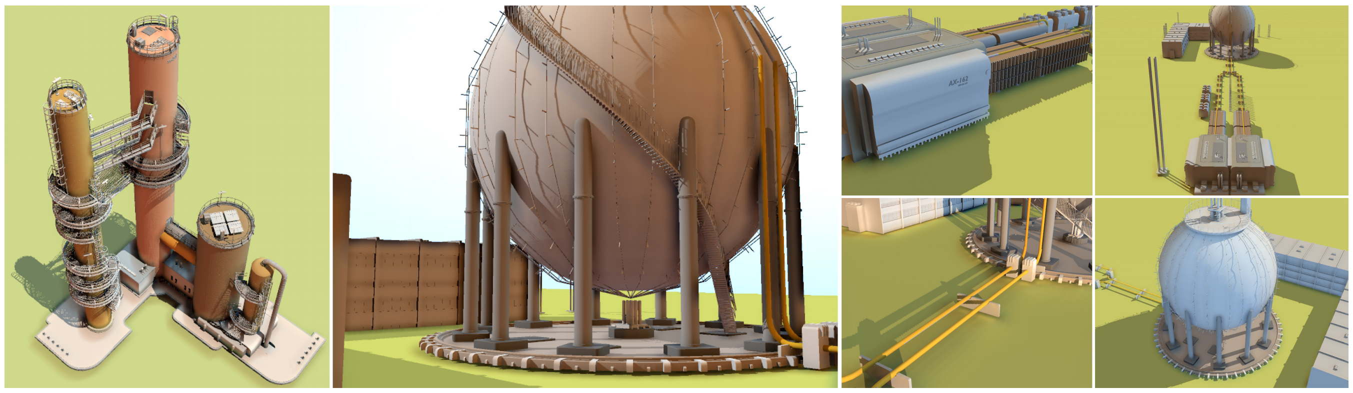

Figure 1: *We generalize screen-space ambient occlusion (SSAO) to directional occlusion (SSDO) and one additional diffuse indirect bounce of light. This scene contains 537k polygons and runs at 20.4 fps at 1600×1200 pixels. Both geometry and lighting can be fully dynamic.*

Abstract

Physically plausible illumination at real-time framerates is often achieved using approximations. One popular example is ambient occlusion (AO), for which very simple and efficient implementations are used extensively in production. Recent methods approximate AO between nearby geometry in screen space (SSAO). The key observation described in this paper is, that screen-space occlusion methods can be used to compute many more types of effects than just occlusion, such as directional shadows and indirect color bleeding. The proposed generalization has only a small overhead compared to classic SSAO, approximates direct and one-bounce light transport in screen space, can be combined with other methods that simulate transport for macro structures and is visually equivalent to SSAO in the worst case without introducing new artifacts. Since our method works in screen space, it does not depend on the geometric complexity. Plausible directional occlusion and indirect lighting effects can be displayed for large and fully dynamic scenes at real-time frame rates.

CR Categories: I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism; I.3.3 [COMPUTER GRAPHICS]: Color, Shading, Shadowing and Texture

Keywords: radiosity, global illumination, constant time

1 Introduction

Real-time global illumination is still an unsolved problem for large and dynamic scenes. Currently, sufficient frame rates are only achieved through approximations. One such approximation is ambient occlusion (AO), which is often used in feature films and computer games, because of its high speed and simple implementation. However, AO decouples visibility and illumination, allowing only for a coarse approximation of the actual illumination. AO typically displays darkening of cavities, but all *directional* information of the incoming light is ignored. We extend recent developments in screen-space AO towards a more realistic illumination we call *screen-space directional occlusion* (SSDO). The present work explains, how SSDO

- accounts for the direction of the incoming light,
- includes one bounce of indirect illumination,
- complements standard, object-based global illumination and
- requires only minor additional computation time.

This paper is structured as follows: First, we review existing work in Section 2. In Section 3 we describe our generalizations of ambient occlusion for the illumination of meso-structures. Section 4 explains extensions to improve the visual quality. In Section 5 the integration of our method into a complete global illumination simulation is described. We present our results in Section 6, we discuss in Section 7 before concluding in Section 8.

2 Previous Work

Approximating physically plausible illumination at real-time framerates has recently received much attention. *Ambient occlusion* (AO) [Cook and Torrance 1981][Zhukov et al. 1998] is used in production extensively [Landis 2002] because of its speed, simplicity and ease of implementation. While physically correct illumination computes the integral over a product of visibility and illumination for every direction, AO computes a product of two individual integrals: one for visibility and one for illumination. For static scenes, AO allows to pre-compute visibility and store it as a scalar field over the surface (using vertices or textures). Combining static AO and dynamic lighting using a simple multiplication gives perceptually plausible [Langer and Bülthoff 2000] results at high framerates. To account for dynamic scenes, Kontkanen et al. [2005] introduced AO Fields, which allow rigid translation and rotation of objects and specialized solutions for animated characters exist [Kontkanen and Aila 2006].

Deforming surfaces and bounces of indirect light are addressed by Bunnell [2005] using a set of disks to approximate the geometry. A more robust version was presented by Hoberock and Jia [2007], which was further extended to point-based ambient occlusion and interreflections by Christensen [2008]. Mendez et al. [2006] compute simple color bleeding effects using the average albedo of the surrounding geometry.

These methods either use a discretization of the surface or rely on ray-tracing, which both do not scale well to the amount of dynamic geometry used in current interactive applications like games. Therefore, instead of computing occlusion over surfaces, recent methods approximate AO in *screen space* (SSAO)[Shanmugam and Arikan 2007; Mittring 2007; Bavoil et al. 2008; Filion and McNaughton 2008]. The popularity of SSAO is due to its simple implementation and high performance: It is output-sensitive, applied as a post-process, requires no additional data (e.g. surface description, spatial acceleration structures for visibility like BVH, kD-trees or shadow maps) and works with many types of geometry (e.g. displacement / normal maps, vertex / geometry shaders, iso-surface raycasting). Image-space methods can also be used to efficiently simulate subsurface scattering [Mertens et al. 2005]. At the same time, SSAO is an approximation with many limitations, that also apply to this work, as we will detail in the following sections.

AO is a coarse approximation to general light transport as eg. in PRT [Lehtinen and Kautz 2003], which also supports *directional occlusion* (DO) and interreflections. Pre-computation requires to store high amounts of data in a compressed form, often limiting the spatial or directional resolution. Our approach allows to both resolve very small surface details and all angular resolutions: „no-frequency" AO, all-frequency image-based lighting and sharp shadows from point lights. While PRT works well with distant lighting and static geometry of low to moderate complexity, its adaptation to real applications can remain involved, while SSAO is of uncompromised simplicity.

In summary, our work takes advantage of information that is already computed during the SSAO process [Shanmugam and Arikan 2007] to approximate two significant effects which contribute to the realism of the results: *Directional occlusion* and *indirect bounces*, both in real-time, which was previously impossible for dynamic, high-resolution geometry.

3 Near-field Light Transport in Image Space

To compute light transport in image space, our method uses a framebuffer with positions and normals [Segovia et al. 2006] as input, and outputs a framebuffer with illuminated pixels using two rendering passes: One for *direct light* and another one for *indirect bounces*.

Direct Lighting using DO Standard SSAO illuminates a pixel by first computing an average visibility value from a set of neighboring pixels. This occlusion value is then multiplied with the unoccluded illumination from all incoming directions. We propose to *remove this decoupling* of occlusion and illumination in the following way:

For every pixel at 3D position $\mathbf{P}$ with normal $\mathbf{n}$, the direct radiance L_{dir} is computed from N sampling directions ω_i, uniformly distributed over the hemisphere, each covering a solid angle of $\Delta\omega = 2\pi/N$:

$$L_{\text{dir}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\text{in}}(\omega_i) V(\omega_i) \cos\theta_i \Delta\omega.$$

Each sample computes the product of incoming radiance L_{in}, visibility V and the diffuse BRDF ρ/π. We assume that L_{in} can be efficiently computed from point lights or an environment map. To avoid the use of ray-tracing to compute the visibility V, we approximate occluders in *screen space* instead. For every sample, we take a step of random length $\lambda_i \in [0 \ldots r_{\max}]$ from $\mathbf{P}$ in direction ω_i, where $r_{\max}$ is a user-defined radius. This results in a set of sampling points $\mathbf{P} + \lambda_i\omega_i$, located in a hemisphere, centered at $\mathbf{P}$, and oriented around $\mathbf{n}$. Since we generated the sampling points as 3D positions in the local frame around $\mathbf{P}$, some of them will be above and some of them will be below the surface. In our approximate visibility test, all the sampling points *below* the surface of the nearby geometry are treated as *occluders*. Fig. 2 (left) shows an example

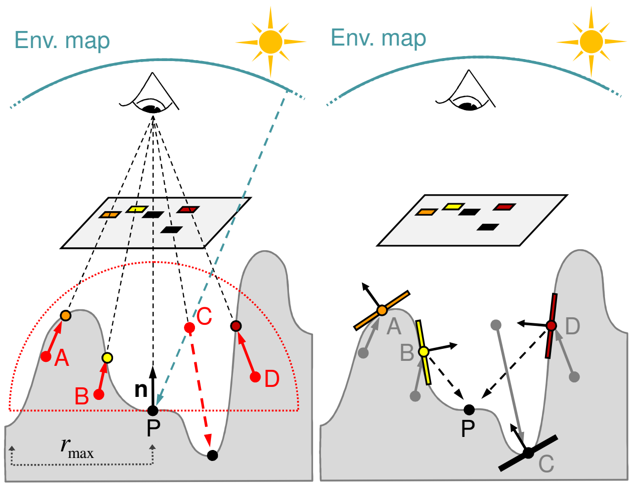

Figure 2: *Left: For direct lighting with directional occlusion, each sample is tested as an occluder. In the example, point P is only illuminated from direction C. Right: For indirect light, a small patch is placed on the surface for each occluder and the direct light stored in the framebuffer is used as sender radiance.*

with $N = 4$ sampling points A,B,C and D: The points A, B and D are below the surface, therefore they are classified as *occluders* for $\mathbf{P}$, while sample C is above the surface and classified as *visible*. To test if a sampling point is below the surface, the sampling points are back-projected to the image. Now, the 3D position can be read from the position buffer and the point can be projected onto the surface (red arrows). A sampling point is classified as *below* the surface if its distance to the viewer *decreases* by this projection to the surface. In the example in Fig. 2, the samples A,B and D are below the surface because they move towards the viewer, while sample C moves away from the viewer. In contrast to SSAO, we do not compute the illumination from all samples, but only from the visible directions (Sample C). Including this directional information can improve the result significantly, especially in case of incoming illumination with different colors from different directions. As shown in Fig. 3, we can correctly display the resulting colored shadows, whereas SSAO simply displays a grey shadow at each location.

Indirect Bounces To include one indirect bounce of light, the direct light stored in the framebuffer from the previous pass can be used: For each sampling point which is treated as an occluder (A, B, D), the corresponding pixel color L_{pixel} is used as the *sender radiance* of a small patch, oriented at the surface (see Fig. 2 right). We consider the sender normal here to avoid color bleeding from back-facing sender patches. The additional radiance from the surrounding

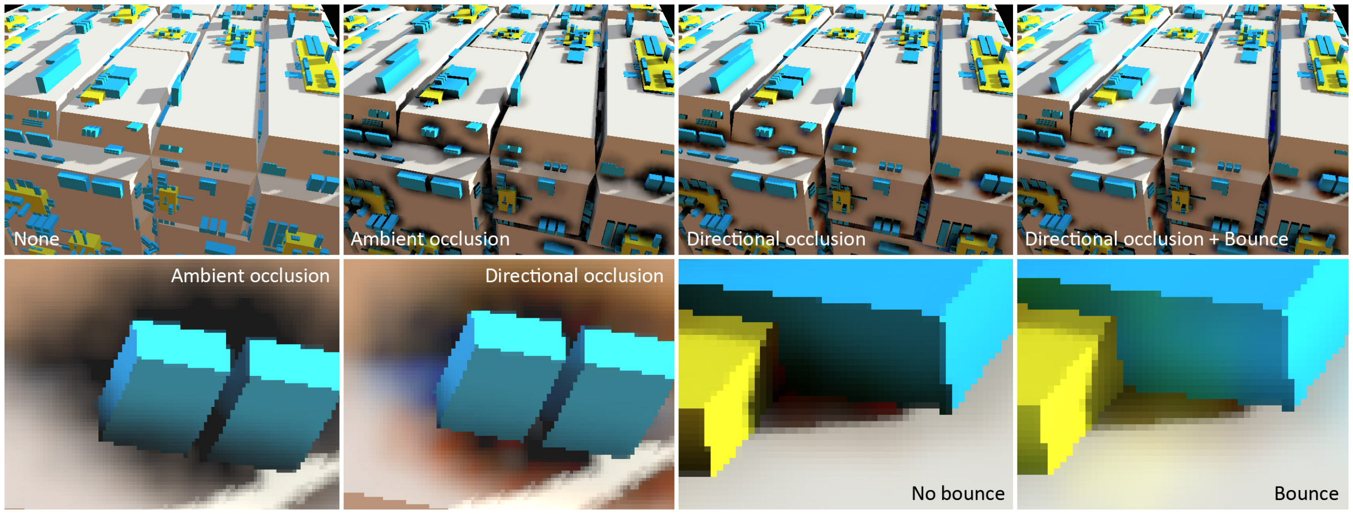

Figure 3: *The top row shows the difference between no AO, standard SSAO, our method with directional occlusion (SSDO) and one additional bounce. In this scene an environment map and an additional point light with a shadow map are used for illumination. The insets in the bottom row show the differences in detail. With SSDO, red and blue shadows are visible, whereas AO shadows are completely grey (bottom left). The images on the bottom right show the indirect bounce. Note the yellow light, bouncing from the box to the ground. The effect of dynamic lighting is seen best in the supplemental video.*

geometry can be approximated as

$$L_{\mathrm{ind}}(\mathbf{P}) = \sum_{i=1}^{N} \frac{\rho}{\pi} L_{\mathrm{pixel}}(1 - V(\omega_i)) \frac{A_{\mathrm{s}} \cos\theta_{\mathrm{s}_i} \cos\theta_{\mathrm{r}_i}}{d_i^2}$$

where d_i is the distance between $\mathbf{P}$ and occluder i (d_i is clamped to 1 to avoid singularity problems), θ_{s_i} and θ_{r_i} are the angles between the sender/receiver normal and the transmittance direction. A_{s} is the area associated to a sender patch. As an inital value for the patch area we assume a flat surface inside the hemisphere. So the base circle is subdivided into N regions, each covering an area of $A_{\mathrm{s}} = \pi r_{\max}^2/N$. Depending on the slope distribution inside the hemisphere, the actual value can be higher, so we use this parameter to control the strength of the color bleeding manually. In the example in Fig. 2, no indirect lighting contribution is calculated for patch A, because it is back-facing. Patch C is in the negative half-space of P, so it does not contribute, too. Patches B and D are senders for indirect light towards $\mathbf{P}$. Fig. 3 shows bounces of indirect light.

Implementation Details Note, that classic SSAO [Shanmugam and Arikan 2007] has similar steps and computational costs. Our method requires more computation to evaluate the shading model, but a similar visibility test. In our examples, we use additional samples for known important light sources (e.g. the sun), applying shadow maps that capture shadows from distant geometry instead of screen-space visibility. We use an $M \times N$-texture to store M sets of N pre-computed low-discrepancy samples $\lambda_i \omega_i$. At runtime, every pixel uses one out of the M sets. In a final pass we apply a geometry-sensitive blur [Segovia et al. 2006] to remove the noise which is introduced by this reduction of samples per pixel.

4 Multiple Pixel Values

Since we are working in screen space, not every blocker or source of indirect light is visible. Fig. 4 shows an example where the color bleeding is smoothly fading out when the source of indirect illumination becomes occluded. There are no visually disturbing artifacts, as shown in the accompanying video, but the results are biased. For a less biased solution, we present two approaches overcoming such limitations: *Depth peeling* and *additional cameras*.

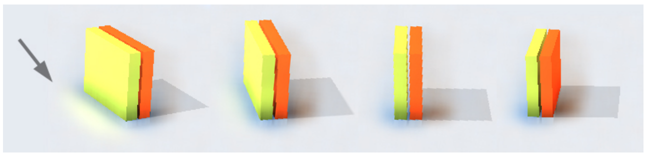

Figure 4: *Four frames from an animation. In the first frame light is bounced from the yellow surface to the floor (see arrow). While the yellow surface becomes smaller in screen space the effect fades out smoothly.*

Single-depth Limitations The blocker test described in the previous section is an approximation, since only the first depth value is known and information about occluded geometry is lost in a single framebuffer. Sampling points can therefore be misclassified, as shown in Fig. 5 (left). In certain situations, we can miss a gap of incoming light or classify a blocked direction as visible. While a missed blocker (sample B) can be corrected by simply increasing the number of samples for this direction, the gap at sample A (and the indirect light coming from the vicinity of A) can not be detected from the single viewpoint, because we have no information about the scene behind the first depth value z_1.

Depth Peeling When using depth peeling [Everitt 2001], the first n depth values are stored after n render passes for each pixel in the framebuffer. This allows us to improve the blocker test, since we have more information about the structure of our scene. Instead of just testing if a sampling point is *behind* the first depth value z_1, we additionally test if the sampling point is *in front* of the second depth value z_2. When using two-manifold geometry, the first and second depth value correspond to a front - and a backface of a solid object, so a sampling point between the two faces must be inside this object (see Fig. 5 right). To reconstruct all shadows for scenes with higher depth complexity, all pairs of consecutive depth

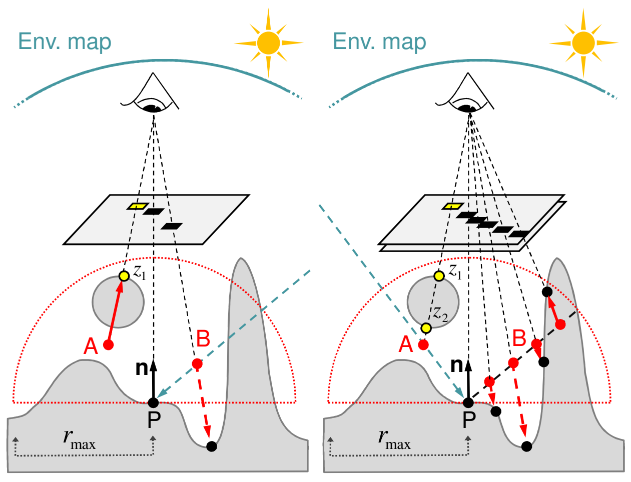

Figure 5: *Problems with screen-space visibility (left): The visible sample A is classified as an occluder because its projected position is closer to the viewer. Sample B is above the surface, but the corresponding direction is blocked, so P is incorrectly illuminated from this direction. Solutions (right): Using depth peeling with two layers, sample A is classified as visible, because it is not between the first and second depth value. When using more samples for the direction of B, the occluder can be found.*

values (third and forth depth value and so on) must be evaluated in the same way [Lischinski and Rappoport 1998]. Fig. 6 shows screen-space shadows of a point light. For a single point light, the N samples are uniformly distributed on a line segment of length $r_{\max}$, directed towards the light source. Since there is no information about the actual *width* of the blocker from a single depth buffer, the shape of the shadow depends on $r_{\max}$. A correct shadow can only be displayed with depth peeling (avg. overhead +30%). In addition to the visibility, color bleeding effects from backfacing or occluded geometry can be displayed with multiple layers, since we can compute the direct lighting for each depth layer.

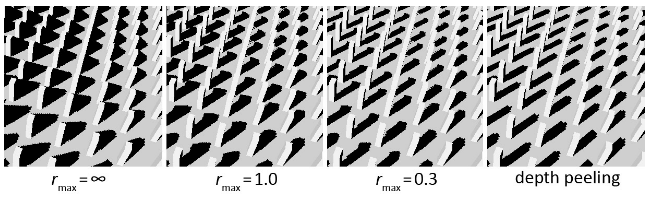

Figure 6: *Screen-space shadows for different values of* $r_{\max}$.

Additional Cameras Depth peeling removes many problems related to SSDO. Alternatively, different camera positions can be used instead of different depth layers to view hidden regions. Beside gaining information about *offscreen* blockers, a different camera position can be especially useful for polygons which are viewed under a *grazing angle*. These polygons cannot faithfully be reproduced, so color bleeding from such polygons vanishes. When using an additional camera, these sources of indirect light can become visible. The best-possible viewpoint for an additional camera would be completely different from the viewer camera, e.g. rotated about 90 degrees around the object center to view the grazing-angle polygons from the front. However, this introduces the problem of occlusion: In a typical scene many of the polygons visible in the viewer camera will be occluded by other objects in the additional camera view. Of course, this can be solved by using depth peeling for the additional camera as well. A faster solution would be a large value for the near clipping plane, adjusted to the sample radius $r_{\max}$, but this is hard to control for the whole image. As a compromise, we used *four* additional cameras with a *standard* depth buffer, all directed to the same center-of-interest as the viewer camera. The relative position of an additional camera to the viewer camera is set manually with a normalized displacement vector. To adapt to the scene size, each displacement vector is scaled with the radius of the bounding sphere of the scene.

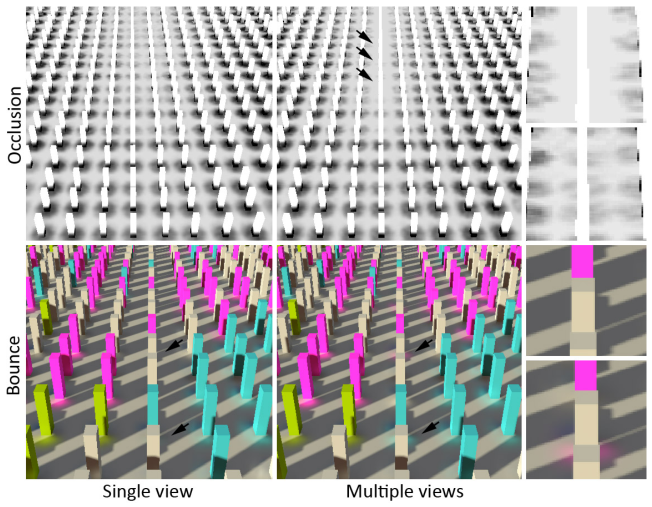

Figure 7: *Comparing occlusion (top) and bounces (bottom) of a single view (left, 49.2 fps) with multiple views (middle, 31.5 fps). Multiple views allow surfaces occluded in the framebuffer to contribute shadows or bounces to unoccluded regions. Using multiple views, occluded objects still cast a shadow that is missed by a single view (top right). An occluded pink column behind a white column bounces light that is missed using a single view (lower right).*

Using an additional camera can display color bleeding from occluded regions, offscreen objects and grazing-angle polygons (Fig. 7 (bottom)). We use framebuffers with a lower resolution for the additional cameras. Therefore, memory usage and fillrate remains unaffected. The drawback of this extension is that vertex transformation time grows linearly with the number of cameras. For a scene with simple geometry (e.g. Fig. 7), the average overhead using four cameras is +58% while for large scenes (e.g. Fig. 12, 1M polygons) it is +160%. When using an alternative method like iso-surface raycasting or point rendering, the number of rays resp. points per view can also be kept constant, giving multiple views the same performance as a single view.

5 Integration in Global Illumination

We believe that screen-space approaches can improve the results of global illumination simulations in case of complex geometry. To avoid a time-consuming solution with the original geometry, the basic idea is to first compute the global illumination on a *coarse representation* of the geometry. Then, the lighting details can quickly be added in screen space at runtime. We demonstrate this idea for the two cases of 1. Environment Map Illumination and 2. Instant Radiosity [Keller 1997], where indirect light is computed from a reflective shadow map [Dachsbacher and Stamminger 2005], resulting in a set of virtual point lights (VPLs). In contrast to our previous application (Sec. 3), we now compute *all* occlusions instead of only the small-scale shadows. Therefore, the indirect illumination is computed from each VPL and the indirect visibility is computed with a shadow map for each VPL. When using shadow mapping, a simplified, sampled representation of the scene (the depth map) is created for visibility queries. Additionally, a *depth bias* must be added to the depth values in the shadow map to avoid wrong self-occlusions. While this removes the self-occlusion artifacts, shadows of small surface details and contact shadows are lost, especially when the light source illuminates from a grazing angle. These contact shadows are perceptually important, without them, objects seem to float over the ground. Instead of applying high-resolution shadow maps in combination with sophisticated methods to eliminate the bias problems, we can use our screen-space approach to reconstruct the shadows of the small details and the contact shadows.

Shadow Mapping and Depth Bias Shadow Mapping [Williams 1978] generates a depth texture from the point of view of the light source and compares the depth values stored in the texture with the real distance to the light source to decide whether a point is in shadow or not. Since each texel stores the depth value of the pixel center, we can think of a texel as a small patch, located on the surface of the geometry (see Fig. 8). The orientation of the patches is parallel to the light source, so the upper half of each patch is above the surface. Since this part is closer to the light than the actual surface, self-occlusion artifacts appear (Fig. 9 left). To remove this artifact, the patches must be moved completely *below* the surface. The standard solution for this is to add a depth bias $b = \frac{p}{2} \cdot tan(\alpha)$, where p is the size of one patch in world coordinates and α is the angle between the light direction and the surface normal (see Fig. 8). The patch size p can be computed from the distance to the light, the texture resolution and the opening angle of the spot.

Screen Space Shadow Correction Fig. 8 shows that we will not be able to display shadows of small details, e.g. at a point $\mathbf{P}$, with a shadow map. Therefore we test each framebuffer pixel, which is not occluded from the shadow map, for missing shadows in *screen space*. Since we know the amount of bias b we introduced, the basic idea is to place a few samples in this *undefined* region. More precisely, we place a user-defined number of samples on a line segment between $\mathbf{P}$ and $\mathbf{P} + b \cdot l$, where l is a unit vector pointing to the light source. For each of the sampling points we perform the same occlusion test as described in Section 3 and 4. If one of the sampling points is classified as an occluder for $\mathbf{P}$, we draw a shadow at the corresponding pixel in the framebuffer. In this way, we adapt our shadow precision to the precision of the visible occluders: As soon as an occluder becomes visible in screen space, we can detect its shadow in screen space. Occluders smaller than the framebuffer resolution will not throw a shadow. Fig. 9 shows how contact shadows can be filled in screen space.

Global Illumination Fig. 10 shows how SSDO can be used for natural illumination. Here, an environment map is represented by a set of point lights with a shadow map for each light. Note the lost shadows of small details due to shadow mapping which can be seamlessly recovered in screen space. For each shadow map, 8 samples were used to detect occluders in screen space. Multiplying a simple ambient occlusion term on top of the image would not recover such illumination details correctly [Stewart and Langer 1997].

Fig. 11 shows the integration of our method into an Instant Radiosity simulation. Again, missing contact shadows can be added in screen space. A related problem with Instant Radiosity is, that clamping must be used to avoid singularity artifacts for receivers close to a VPL. We can add such bounces of light in screen space for nearby geometry, as shown in Fig. 12. Although our estimated formfactor has a singularity too, the local density of samples is much higher than the density of the VPLs. This idea of correction in screen space can be further extended to any type of illumination which is represented from VPLs, e.g. illumination from area lights and all other approaches to visibility having a limited resolution $r_{\max}$ [Lehtinen and Kautz 2003; Ren et al. 2006; Ritschel et al. 2008].

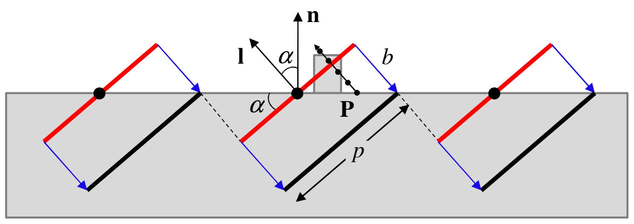

Figure 8: *To remove the wrong self-occlusion, each shadow map texel (shown in red) must be moved below the surface (black). This bias b depends on the slope α of the incoming light direction. Due to the coarse shadow map resolution and this bias, shadows of small scale geometry are lost in this way, for example at point P. To reconstruct these shadows, a few samples are created on a line segment of length b, starting at P, directed towards the light source. For each of the samples, we use our image-based blocker test.*

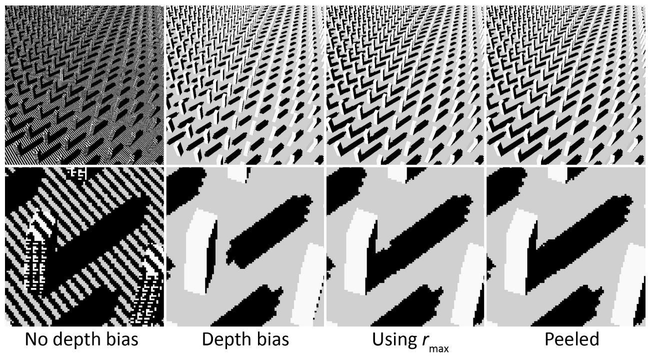

Figure 9: *Shadows from small occluders (*1024×768 *framebuffer,* 1024×1024 *depth map), from left to right: Without depth bias, self-shadowing artifacts are visible. Classic depth bias removes the wrong self-occlusion, but shadows at contact points disappear (218 fps). Screen-space visibility removes depth bias using a single depth buffer (16 samples, 103 fps) or depth peeling (8 samples, 73 fps).*

6 Results

In the following we present our results, rendered with a 3 GHz CPU and an NVIDIA GeForce 8800 GTX.

Performance Our method works completely in image space, therefore we can display directional occlusion and indirect bounces for large scenes at real-time framerates. Table 1 shows the timing values for the factory scene (Fig. 1, 537k polygons) using a single camera and a single depth layer. Without depth peeling or additional cameras, including directional occlusion adds only a modest amount of overhead to SSAO: $+3.6\%$ for DO and $+31.1\%$ for DO with one bounce. While adding DO effects incurs a negligible overhead, an additional diffuse bounce requires a little more computation time. We assume that the shader is bound by bandwidth and not by computation.

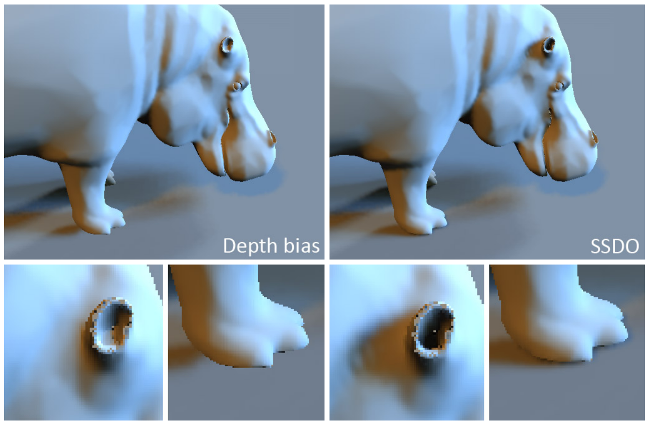

Figure 10: *Depth bias in this natural illumination rendering* (512×512, *54.0 fps) is removed using SSDO combined with shadow mapping (25.2 fps). We use 256 VPLs with a* 512×512 *depth map each.*

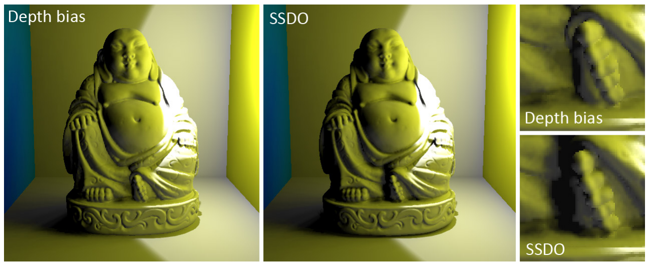

Figure 11: *Instant Radiosity with screen-space shadow correction. Shadows of small details, like the toes, are lost due to the depth bias. These small shadows can be restored in image space.*

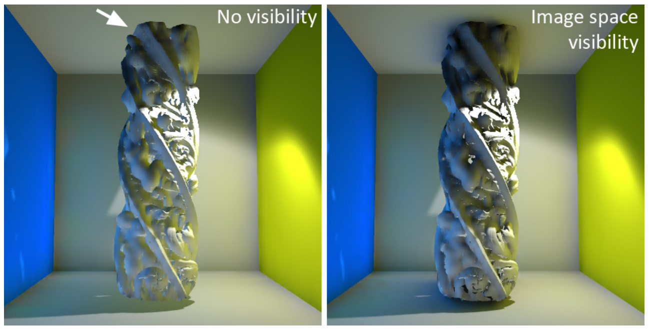

Figure 12: *Instant Radiosity with shadow correction and an additional indirect bounce in screen space. Note the additional contact shadows and the color bleeding effects. The indirect light is slightly scaled here to make the additional effects more visible.*

Table 1: *Typical frame rates, using an Nvidia Geforce 8800 GTX. For SSAO we use a single pre-filtered environment map lookup instead of one environment lookup per sample for DO. Overhead is relative to SSAO alone. The frame rate for pure rendering without any occlusion is 124 fps at 1600×1200. M is set to 4×4.*

Resolution	Samples	Time costs				
		SSAO	SSDO		+ 1 Bounce	
		fps	fps	+%	fps	+%
1024×768	$N = 8$	81.0	81.3	0.2	65.8	19.3
	$N = 16$	58.0	56.5	2.6	40.5	30.2
1200×960	$N = 8$	37.7	37.5	0.5	25.5	32.4
	$N = 16$	25.4	23.6	7.1	14.7	22.2
1600×1200	$N = 8$	24.8	24.2	2.4	15.9	35.9
	$N = 16$	16.8	15.3	8.9	9.0	46.4

Time-Quality Tradeoff Including depth peeling and additional cameras results in an overhead of $30\% - 160\%$ for our test scenes (2 depth layers or 4 cameras). The use of these extensions is a time-quality tradeoff. For environment map illumination, we observed good results without them, since illumination comes from many directions and most of the visible errors are hidden. Fig. 13 shows a comparison of our (unextended) method with a ground truth path tracing image, generated with PBRT [Pharr and Humphreys 2004].

Animated Scenes Since our method operates completely in image space, animated scenes can be displayed without any restrictions. The accompanying video shows moving colored blocks and animated objects without temporal flickering.

7 Discussion

7.1 Perception

Adding bounces and DO improves the perception of meso-structures, bridging geometry and material in a consistently lit way. One problem with classic AO is, that lighting is completely ignored: everything in an occluder's vicinity is darkened equally, even if the lighting is not isotropic. In many scenarios, like skylights (Fig. 1 and 3), lighting is smooth but still has a strong directional component. Using classical AO under varying illumination introduces objectionable artifacts since the contact shadows remain static while the underlying shading changes. This results in an impression of dirt, because a change in reflectivity (dust, dirt, patina) is the perceptually most plausible explanation for such equal darkening under varying illumination. With proper DO, details in the meso-structure cast appropriate miniature shadows (softness, size, direction, color) that relate to the macro-illumination, as illustrated in Fig. 3.

7.2 Quality

A screen-space approach approximates the geometry in 3D spatial proximity of pixel p with a set of nearby pixels $\mathcal{P}$ in 2D. This requires that the local geometry is sufficiently represented by the nearby pixels. By adjusting the radius $r_{\max}$ of the hemisphere, shadows from receivers of varying distance can be displayed. Using a small radius results in shadows of only tiny, nearby cavities whereas larger shadows of even distant occluders can be displayed with a large hemisphere. Fig. 14 shows the effect of varying the size of $r_{\max}$. We use a smooth-step function to fade out shadows from blockers close to the border of the hemisphere to avoid cracks in the shadow if a large blocker is only partially inside the hemisphere. The radius is adjusted manually for each scene, finding a good value automatically is difficult. For a closed polygonal model, an inital value for $r_{\max}$ can be found by computing the average distance between two local maxima in the geometry. With only a single depth layer, wrong shadows can appear at the edges of objects (an effect similar to depth darkening [Luft et al. 2006]) if a large value is chosen for $r_{\max}$. This happens because the frontmost object can be misclassified as an occluder, as shown in Fig. 6 left. While this effect enhances the appearance of still images, a sudden darkening around edges can be distracting for moving images (see the accompanying video). When reducing the radius, the width of the darkening around edges becomes smaller, but some real shadows disappear, too. Depth

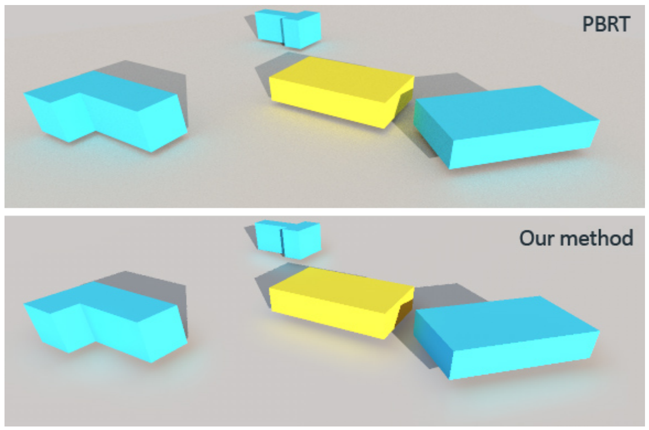

Figure 13: *Comparison of our approach with a one-bounce path tracing result of PBRT. The scene is illuminated by an environment map with one additional point light. Several lights and shadows are in the wrong place and appear two-dimensional. Although some haloing artifacts are introduced, the directional occlusion and indirect lighting effects present in the ground truth rendering are faithfully reproduced with our technique.*

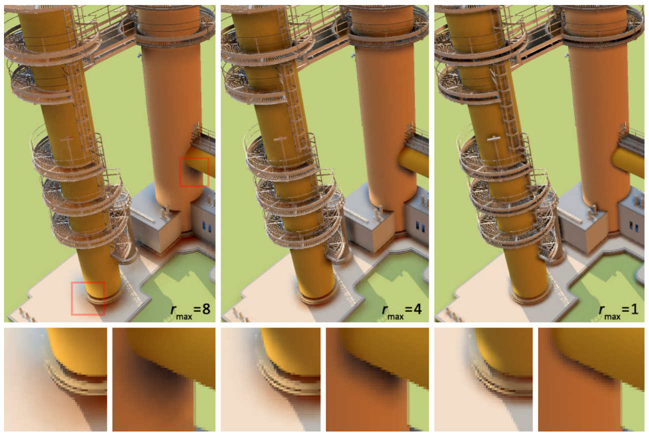

Figure 14: *SSDO with one indirect bounce for different values of* $r_{\max}$. *In this case, the illumination is computed from a pointlight with a shadow map and an additional environment map. Note that using a smaller value for* $r_{\max}$ *decreases the size of the screen-space shadows as well as the influence region of the indirect bounce.*

peeling can solve this problem, but with additional rendering time. From a theoretical point of view, all shadows can be correctly displayed in screen space when depth peeling and additional cameras are used, because the scene structure is captured completely. However, a large radius and many sampling points N must be used to avoid missing small occluders, so standard techniques like shadow mapping should be preferred for distant blockers, SSDO then adds the missing small-scale details.

Besides the limitations (and solutions) discussed in Section 4, *sampling* and illumination *bias* affect the quality.

Sampling When $\mathcal{P}$ gets small in the framebuffer, because it is far away or appears under grazing angles, the SSAO effect vanishes. Gradually increasing the distance or the angle will result in a smooth, temporally coherent fading of our additional effects. Increasing distance or increasing the angle gradually, will also give a gradual fading and will be temporally coherent. However, this issue can be the most distracting: pixels belonging to surfaces only a few pixels in size sometimes receive no bounces or occlusion at all. Still, this is an underlying problem of the original SSAO and our examples circumvent it by using high-resolution framebuffers (e.g. 1600×1200).

Bias in Illumination Since our algorithm is geared towards real-time display, we accept a biased solution which is different from a physical solution. However, some of the errors can be reduced by spending more computation time. When using high-resolution depth buffers in combination with multiple cameras and depth peeling (adjusted to the depth complexity of the scene), the whole scene is captured and a correct visibility test can be computed. This means that SSDO can compute the correct illumination for a point when increasing the number of samples, the remaining error is a discretization in visibility. The bounce of indirect light is more complicated: Here we use the *projected* positions of uniformly distributed samples. After the projection, the samples no longer follow a known distribution. This means that some regions have more samples while other regions suffer from undersampling. Adding more depth layers and multiple viewpoints can make every source of indirect light visible, but the sampling distribution is still non-uniform. Additionally, we only use an approximated form factor. In our experiments, we have not observed visible artifacts from these sources of error. We consider the unbiased solution for indirect light in screen space as future work.

We conclude, that our method does not introduce any additional (temporal) artifacts. In the worst case our approach fades into traditional SSAO which again is equivalent to no AO at all in the worst case. We expect the distant and bounced incident radiance to be sufficiently smooth, like the skylight in Fig. 1. High-frequency environment maps require either pre-smoothing or higher sampling rates. We have not observed any noise caused by high frequency changes in radiance (e.g. at direct shadow boundaries).

8 Conclusions

We presented two generalizations of screen-space ambient occlusion that add directional occlusion and diffuse indirect bounces. Both extensions considerably improve realism in the best case and look like the unextended technique in the worst case. We incur a modest computational overhead without introducing any additional artifacts, allowing real-time display of complex and dynamic scenes. In future work, we will investigate if an unbiased computation of indirect light is possible in screen space. Additionally, blocking information could be stored for each pixel and reused for the next frame. Other possible extensions would be multiple indirect bounces, specular materials, subsurface scattering and importance sampling. One could think about omitting the projection step entirely: Instead of using the framebuffer to find proximity between pixels, an acceleration structure like a hash table could be used to find proximity between points in space. We believe that the optimal combination of screen-space and object-space approaches is an interesting avenue of further research bringing together physically plausible illumination of dynamic complex scenes and real-time frame rates.

Acknowledgements

We thank Zhao Dong, Derek Nowrouzezahrai, Karol Myszkowski, Tom Annen and the anonymous reviewers for their useful comments and suggestions. The animal animation sequences are courtesy of MIT CSAIL, the factory scene is from Dosch Design. The volumetric skull data sets are 3D scans from the University of Texas at

Austin (UTCT). The tortile column model is scanned data from VCL PISA, the buddha model is taken from the aim-at-shape repository.

References

BAVOIL, L., SAINZ, M., AND DIMITROV, R. 2008. Image-space horizon-based ambient occlusion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks*, ACM, New York, NY, USA, 1–1.

BUNNELL, M. 2005. Dynamic Ambient Occlusion and Indirect Lighting. In *GPU Gems 2*, M. Pharr, Ed. Addison Wesley, Mar., ch. 2, 223–233.

CHRISTENSEN, P. H. 2008. Point-Based Approximate Color Bleeding. *Pixar Technical Memo 08-01*.

COOK, R. L., AND TORRANCE, K. E. 1981. A reflectance model for computer graphics. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, 307–316.

DACHSBACHER, C., AND STAMMINGER, M. 2005. Reflective Shadow Maps. In *Proceedings of the ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, 203–213.

EVERITT, C., 2001. Introduction Interactive Order-Independent Transparency. NVidia Technical Report.

FILION, D., AND MCNAUGHTON, R. 2008. Effects & techniques. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, ACM, New York, NY, USA, 133–164.

HOBEROCK, J., AND JIA, Y. 2007. High-Quality Ambient Occlusion. In *GPU Gems 3*. Addison-Wesley, Reading, MA, ch. 12.

KELLER, A. 1997. Instant Radiosity. In *Proceedings of ACM SIGGRAPH*, 49–56.

KONTKANEN, J., AND AILA, T. 2006. Ambient Occlusion for Animated Characters. In *Eurographics Symposium on Rendering*.

KONTKANEN, J., AND LAINE, S. 2005. Ambient Occlusion Fields. In *Proceedings of ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games*, 41–48.

LANDIS, H. 2002. RenderMan in Production. In *ACM SIGGRAPH 2002 Course 16*.

LANGER, M. S., AND BÜLTHOFF, H. H. 2000. Depth Discrimination From Shading Under Diffuse Lighting. *Perception 29*, 6, 649 – 660.

LEHTINEN, J., AND KAUTZ, J. 2003. Matrix Radiance Transfer. In *2003 ACM Symposium on Interactive 3D Graphics*, 59–64.

LISCHINSKI, D., AND RAPPOPORT, A. 1998. Image-Based Rendering for Non-Diffuse Synthetic Scenes. In *Ninth Eurographics Workshop on Rendering*, Eurographics, 301–314.

LUFT, T., COLDITZ, C., AND DEUSSEN, O. 2006. Image Enhancement by Unsharp Masking the Depth Buffer. *ACM Trans. Graph. 25*, 3, 1206–1213.

MENDEZ, A., SBERT, M., CATA, J., SUNYER, N., AND FUNTANE, S. 2006. Realtime Obscurances with Color Bleeding. In *ShaderX4: Advanced Rendering Techniques*. Charles River Media, 121–133.

MERTENS, T., KAUTZ, J., BEKAERT, P., REETH, F. V., AND SEIDEL, H.-P. 2005. Efficient Rendering of Local Subsurface Scattering. *Comput. Graph. Forum 24*, 1, 41–49.

MITTRING, M. 2007. Finding Next-Gen: CryEngine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, ACM, New York, NY, USA, 97–121.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering : From Theory to Implementation*. Morgan Kaufmann.

REN, Z., WANG, R., SNYDER, J., ZHOU, K., LIU, X., SUN, B., SLOAN, P.-P., BAO, H., PENG, Q., AND GUO, B. 2006. Real-time Soft Shadows in Dynamic Scenes using Spherical Harmonic Exponentiation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM, New York, NY, USA, 977–986.

RITSCHEL, T., GROSCH, T., KIM, M. H., SEIDEL, H.-P., DACHSBACHER, C., AND KAUTZ, J. 2008. Imperfect Shadow Maps for Efficient Computation of Indirect Illumination. *ACM Trans. Graph. (Proc. of SIGGRAPH ASIA 2008) 27*, 5.

SEGOVIA, B., IEHL, J.-C., MITANCHEY, R., AND PÉROCHE, B. 2006. Non-interleaved Deferred Shading of Interleaved Sample Patterns. In *SIGGRAPH/Eurographics Graphics Hardware*.

SHANMUGAM, P., AND ARIKAN, O. 2007. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In *Proceedings of ACM Symposium in Interactive 3D Graphics and Games*, ACM, B. Gooch and P.-P. J. Sloan, Eds., 73–80.

STEWART, A. J., AND LANGER, M. S. 1997. Towards Accurate Recovery of Shape from Shading under Diffuse Lighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19*, 1020–1025.

WILLIAMS, L. 1978. Casting Curved Shadows on Curved Surfaces. In *Proceedings of ACM SIGGRAPH*, 270–274.

ZHUKOV, S., IONES, A., AND KRONIN, G. 1998. An Ambient Light Illumination Model. In *Eurographics Rendering Workshop 1998*, 45–56.