Volume 28 (2009), Number 4 pp. 1–12 COMPUTER GRAPHICS *forum*

High-Quality Screen-Space Ambient Occlusion using Temporal Coherence

O. Mattausch and D. Scherzer and M. Wimmer

Vienna University of Technology

Abstract

Ambient occlusion is a cheap but effective approximation of global illumination. Recently, screen-space ambient occlusion (SSAO) methods, which sample the frame buffer as a discretization of the scene geometry, have become very popular for real-time rendering. We present temporal SSAO (TSSAO), a new algorithm which exploits temporal coherence to produce high-quality ambient occlusion in real time. Compared to conventional SSAO, our method reduces both noise as well as blurring artifacts due to strong spatial filtering, faithfully representing fine-grained geometric structures. Our algorithm caches and reuses previously computed SSAO samples, and adaptively applies more samples and spatial filtering only in regions that do not yet have enough information available from previous frames. The method works well for both static and dynamic scenes.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Ambient occlusion (AO) describes the percentage of incident ambient light on a surface point that is occluded by surrounding geometry. AO is used for shading by modulating the ambient color of the surface point with it. From a physical point of view, ambient occlusion could be seen as the diffuse illumination due to the sky [Lan02]. Ambient occlusion of a surface point p with normal n_p is computed as [CT81]:

$$\mathrm{ao}(p,n_p) = \frac{1}{\pi}\int_{\Omega} V(p,\omega)\max(n_p \cdot \omega, 0)d\omega, \qquad (1)$$

where ω denotes all directions on the hemisphere and V is the (inverse) binary visibility function, with $V(p,\omega) = 1$ if visibility was blocked by an obstacle, 0 otherwise. The visibility term usually considers obstacles within a certain sampling radius only.

Ambient occlusion is heavily used in production rendering and recently also in real-time rendering and many high-profile games [Mit07,FM08], because it is a *local* effect (due to the limited sampling radius) that is very cheap compared to a full global illumination solution, but still greatly enhances the realism of the scene. *Screen Space Ambient Occlusion* (SSAO) techniques decouple the shading from the scene complexity by using the frame buffer as a discrete approximation of the scene geometry. The performance of SSAO depends mainly on the number of samples per frame, hence a relatively small number of samples is typically used to reach the desired performance, and the resulting noise is blurred with a spatial depth-aware filter. However, a small number of samples can be insufficient for shading complex geometry with fine details, especially if a large sampling radius is used. The final image will look either noisy or blurry, depending on the size of the filter kernel. Generally a careful tuning of the filter parameters is required in order to provide stable results, otherwise artifacts may appear, e.g., halos around small depth discontinuities.

In this paper, we present an algorithm that achieves high-quality ambient occlusion which is neither blurry nor prone to noise artifacts, with a minimum amount of samples per frame. We reuse the available AO information from previous frames by exploiting temporal coherence between consecutive image frames. We identify pixels describing identical world positions by means of temporal reprojection. The current state of the solution is cached in a so-called *ambient occlusion buffer*. Each frame we compute a few new AO samples, then blend these with the accumulated samples from the previous frames. The ambient occlusion solution is then

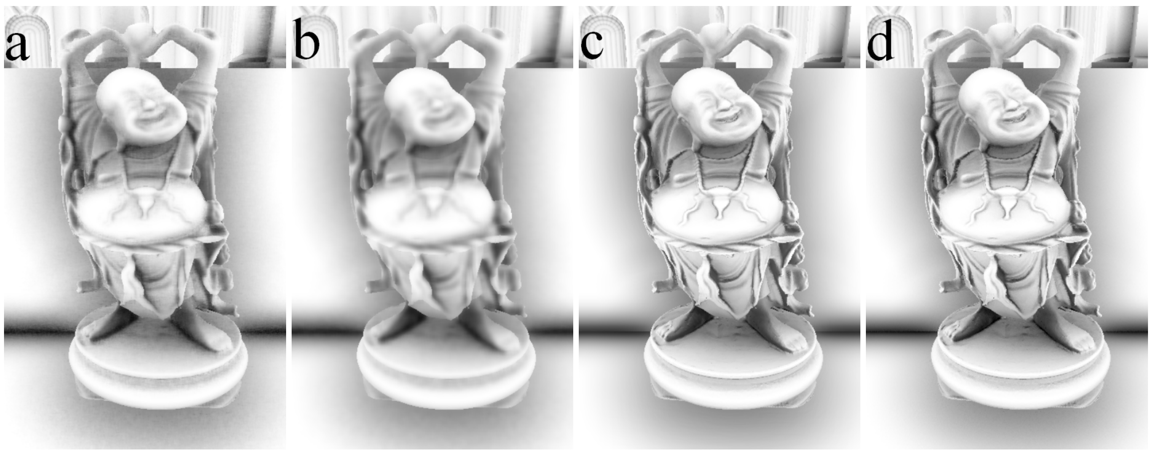

Figure 1: From left to right: *SSAO without temporal coherence (23 FPS) with 32 samples per pixel, with (a) a weak blur, (b) a strong blur. (c) TSSAO (45 FPS), using 8–32 samples per pixel (initially 32, 8 in a converged state). (d) Reference solution using 480 samples per frame (2.5 FPS). All images at 1024x768 resolution and using 32 bit precision render targets. The scene has 7M vertices and runs at 62 FPS without SSAO shading.*

combined with the image resulting from direct diffuse illumination in a separate step. See Figure 1 for a comparison of SSAO with and without employing temporal coherence.

Recently, reprojection techniques have been used for a number of useful applications, like antialiasing or shadow mapping. SSAO is an especially interesting case because it takes a pixel neighborhood into account, which leads to unique challenges, especially for dynamic scenes. First, since SSAO is a postprocessing operation, information about reprojection needs to be stored during the main rendering pass. Second, the validity of reprojected pixels does not depend only on the pixel itself, but also on its neighborhood, requiring a new validity test for reprojected pixels. Third, the convergence of the solution allows us to reduce or completely omit spatial filtering that is typically necessary in SSAO methods, and allows us to rely on a minimal number of new samples per frame. For pixels that have not converged, e.g., when cached samples have been invalidated, we can use information about convergence in order to reconstruct the value from nearby samples using an adaptive convergence-aware filter that gives more weight to already converged samples.

2. Related Work

2.1. Ambient Occlusion

Ambient occlusion is a shading technique that measures the reverse surface exposure and was introduced by Zhukov et al. [ZIK98]. It is extensively used in areas like production rendering [Lan02,PG04] because it is a relatively simple and cheap technique which greatly enhances the appearance of an image.

In recent years several conceptually different approaches were proposed that make online calculation of ambient occlusion in dynamic scenes feasible for real-time applications. Some *object-based* approaches distribute the *occlusion power* of each object on all the other objects in the range of interest [KL05, MMAH07]. These methods have problems handling overlapping occluders. A feasible way of testing blocker visibility using spherical harmonics exponentiation for soft shadows and low-frequency illumination was proposed by Ren et al. [RWS*06]. Bunnell [Bun05] proposed a per-vertex ambient occlusion method for dynamic scenes, which uses a hierarchical approach to break down the complexity, and can be extended to compute diffuse color bleeding.

Image-space methods, on the other hand, interpret the frame buffer as a discrete scene approximation and are often referred to as screen-space ambient occlusion [Mit07]. They are usually used in combination with deferred shading [ST90, Shi05]. The original implementation compared the depth of ambient occlusion samples against the depth buffer in order to estimate the number of samples that are occluded [Mit07]. Shanmugam and Arikan [SA07] proposed a two-stage image space method, one stage for the low frequency far field illumination, and one for high frequency near field ambient occlusion. The high frequency part treats each pixel of the depth buffer as a spherical occluder. Fox and Compton [FC08] proposed a similar SSAO shading technique that samples the depth buffer and weights the contribution by the cosine between surface normal and the vector to the sample as well as the sample distance. Recently, methods that use horizon mapping for computing the occluded part of the surface were proposed [BSD08]. Ritschel et al. [RGS09] extended an SSAO generation method to in-

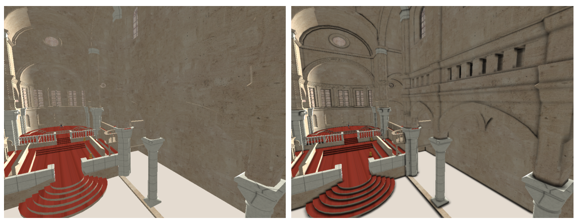

Figure 2: *This figure compares rendering without (left) and with (right) ambient occlusion, and shows that AO allows much better depth perception and feature recognition, without requiring any additional lighting.*

clude directional information and first-bounce global illumination. Their SSAO method is similar to the original method of Mittring, but also takes the angle to the sample point into account to accurately weight the samples that passed the depth test. Bavoil et al. [BS09] proposed some general techniques that aim at reducing the problems of SSAO algorithms due to information missing in the depth buffer representation. They also proposed a multi-resolution approach that computes half resolution SSAO in regions with low difference in the SSAO value range, and switches to high resolution SSAO only in regions with high difference. Our method also makes use of adaptive sampling, in our case with the different goal to reduce the computational effort put into sampling in regions that are already sufficiently converged due to temporal coherence.

Ambient occlusion for character animation can be achieved by precomputing ambient occlusion for a couple of key poses and then interpolating between them [KA06, KA07], at considerable storage cost.

Figure 2 demonstrates the visual impact of SSAO for the depth perception of a scene.

2.2. Reprojection Techniques

In order to use temporal coherence for GPU rendering, we employ a technique called reverse reprojection, which was independently proposed by Scherzer et al. [SJW07] (referred to as temporal reprojection) and Nehab et al. [NSL*07]. This allows the reuse of pixel content from previous frames for the current frame by associating image pixels from the current frame with the pixels from the previous frame that represent the same world space position. The technique was shown to be useful for a variety of applications, like shadow mapping, anti-aliasing, or even motion blur [Ros07].

Smedberg et al. [SW09] independently proposed using reprojection to enhance the quality of SSAO in a GDC talk. However, exact details about the method are not available, and they do not specifically address the proper invalidation of incorrect cache values.

3. Our algorithm

3.1. SSAO generation

SSAO methods aim to approximate the original AO integral in screen space. Several versions of SSAO with different assumptions and trade-offs have been described [BSD08, Mit07, RGS09]. While we demonstrate our technique with only two different ambient occlusion methods, it works with many more, and could be used for several other shading method that depend on a screen-space sampling kernel. We assume that such a shading method can be written as an average over contributions C which depend on a series of samples s_i:

$$AO_n(p) = \frac{1}{n}\sum_{i=1}^{n} C(p,s_i) \tag{2}$$

A typical example contribution function for SSAO would be

$$C(p,s_i) = V(p,s_i)\cos(s_i - p, n_p)D(|s_i - p|).$$

$V(p,s_i)$ is a binary visibility function that gives 0 if s_i is visible from p and 1 otherwise. Visibility is for example determined by checking whether s_i is visible in the z-buffer. We assume that the samples s_i have been precomputed and stored in a texture, for example a set of 3D points uniformly distributed in the hemisphere, which are transformed into the tangent space of p for the evaluation of C [Mit07]. V can be attenuated with a function D of the distance from p to s_i (e.g., an $exp()$ function). In order to achieve faster convergence we use a Halton sequence for sample generation, which is known for its low discrepancy [WH00].

3.2. Reprojection

Reprojection techniques use two render targets in ping-pong fashion, one for the current frame and one representing the cached information from the previous frames (denoted as *real-time reprojection cache* [NSL*07] or *history buffer [SJW07]*). In our context we cache ambient occlusion values and therefore denote this buffer as *ambient occlusion buffer*.

For static geometry, reprojection is constant for the whole frame, and can be carried out in the pixel shader or in a separate shading pass (in the case of deferred shading) using current and previous view (V) and projection (P) matrices, where t denotes the post-perspective position of a pixel [SJW07]:

$$t_{old_{x',y',z'}} = P_{old}V_{old}V_{new}^{-1}P_{new}^{-1}t_{new_{x,y,z}} \tag{3}$$

In our deferred shading pipeline, we store eye linear depth

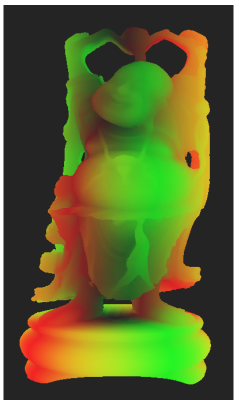

Figure 3: *Frame-to-frame 3D optical flow of a Happy Buddha model rotating around its base. The fixpoint of the rotation is visible as a dark spot.*

values for the current frame and the previous frame, and use them to reconstruct the world space positions p. Note that p_{old} can be obtained from the above formula by applying the inverse view-projection matrix to t. For dynamic scenes, this simple formula does not work because reprojection depends on the transformations of moving objects. Nehab et al. [NSL*07] therefore propose to do the reprojection in the vertex shader by applying the complete vertex transformation twice, once using the current transformation parameters (modeling matrix, skinning etc.) and once using the parameters of the previous frame.

However, in a deferred shading pipeline, p_{old} needs to be accessed in a separate shading pass, where information about transformation parameters is already lost. Therefore, we store the *3D optical flow* $p_{old} - p_{new}$ in the frame buffer as another shading parameter (alongside normal, material etc.), using a lower precision than for the absolute depth values (16 bit instead of 32 bit). See Figure 3 for a depiction of optical flow induced by a rotating Happy Buddha model.

During reprojection, we have to check for pixels that became *invalid* (e.g. due to a disocclusion). This will be described in Section 3.4, where we also show that SSAO imposes some additional constraints for a pixel to stay valid.

3.3. Temporal refinement

The main idea of our algorithm is to spread the computation of ambient occlusion (Equation 2) over several frames by using reprojection. Whenever possible we take the solution from the previous frame that corresponds to an image pixel and refine it with the contribution of new samples computed in the current frame. In frame $t+1$, we calculate a new contribution C_{t+1} from k new samples:

$$C_{t+1}(p) = \frac{1}{k} \sum_{i=n_t(p)+1}^{n_t(p)+k} C(p, s_i) \quad (4)$$

and combine them with the previously computed solution from frame t:

$$AO_{t+1}(p) = \frac{n_t(p)AO_t(p) + kC_{t+1}(p)}{n_t(p)+k} \quad (5)$$

$$n_{t+1}(p) = min(n_t(p)+k, n_{max}), \quad (6)$$

where n_t keeps track of the number of samples that have already been accumulated in the solution. If the cached AO value is invalid, then the previous solution can be discarded by setting n_t to 0. Otherwise, the solution will reach a stable state after a few frames.

Theoretically, this approach can use arbitrarily many samples. In practice, however, this is not advisable: since reprojection is not exact and requires bilinear filtering for reconstruction, each reprojection step introduces an error which accumulates over time. This error is noticeable as an increasing amount of blur [YNS*09]. Furthermore, the influence of newly computed samples becomes close to zero, and previously computed samples never get replaced. Therefore we clamp n_t to a user-defined threshold n_{max}, which causes the influence of older contributions to decay over time. Thus, $conv(p) = \min(n_t(p), n_{max})/n_{max}$ is an indicator of the state of convergence. Note that for $n_{max} \to \infty$, Equation 6 would converge to the correct solution – unlike the exponential smoothing used in previous approaches [SJW07, NSL*07], which acts as a temporal filter kernel.

In our experience, a threshold n_{max} in the range [500..1500] provides a sufficiently fast update frequency to avoid major blurring artifacts while avoiding undersampling artifacts like temporal flickering. Figure 4 (left) depicts how the influence of a new sample changes from the moment it is introduced, and (right) the exponential decay of the influence of a previously computed AO solution after the threshold n_{max} was reached.

3.3.1. Implementation notes

The value n_t is stored in a separate channel in the ambient occlusion buffer. We also store the starting index into the set of precomputed samples, which is used to take the new sampling positions for the current frame from a precomputed low-discrepancy Halton sequence. Hence the current index position is propagated to the next frame by means of reverse reprojection like the SSAO values. In order to prevent the index position to be interpolated by the hardware and introduce a bias into the sequence, it is important to always fetch the index value from the nearest pixel center.

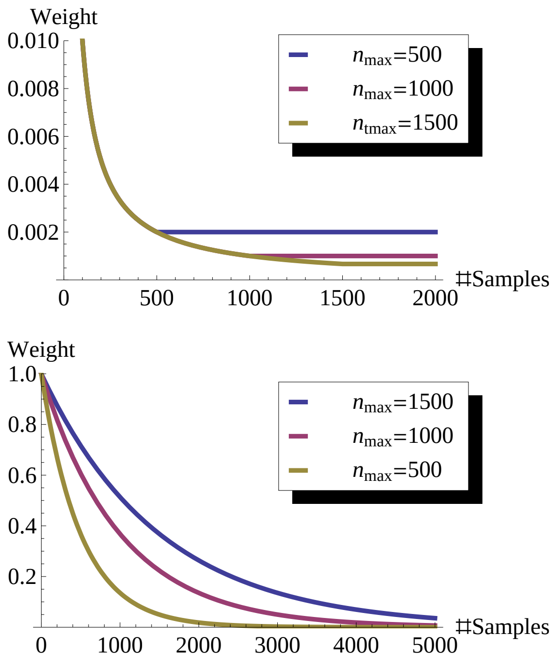

Figure 4: *The weight ($1/n_t$) of a new sample (top) and of a previously computed AO solution (bottom) after given number of samples using thresholds $n_{max} = 500, 1000, 1500$.*

3.4. Detecting and dealing with invalid pixels

When reprojecting a fragment, we need to check whether the cached AO value is valid. If it is not, n_t is reset to 0 and a new AO solution is computed. In order to detect such invalid pixels, we check if either one of the following two conditions has occured: 1) a disocclusion of the current fragment [SJW07, NSL*07], and 2) changes in the *sample neighborhood* of the fragment. In the following we discuss both conditions.

3.4.1. Detecting disocclusions

Previous approaches [SJW07, NSL*07] check for disocclusions by comparing the depth of the reprojected fragment position d_{new} to the depth of the cached value d_{old}:

$$|d_{new} - d_{old}| < \epsilon$$

However, we found that neither screen-space depth values nor eye-linear depth values gave reliable results. Screen-space depth values are only accurate in a small region around the near plane, while eye-linear depth comparisons are overly sensitive in regions near the far plane and not sensitive enough near the near plane. The solution is to store eye-linear depth values, but to consider *relative* depth differences instead of absolute ones, by checking for

$$|1 - \frac{d_{new}}{d_{old}}| < \epsilon, \tag{7}$$

which gives good results even for large scenes with a wide depth range. Note that pixels which were outside the frame buffer in the previous frame are also marked as invalid.

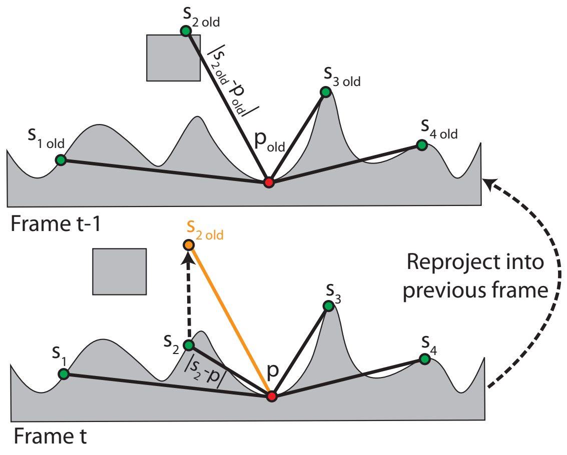

Figure 5: *The distance of p to sample point s_2 in the current frame differs significantly from the distance of p_{old} to s_{2old} in the previous frame, hence we assume that a local change of geometry occurred, which affects the shading of P.*

3.4.2. Detecting changes in the neighborhood

Testing for disocclusion is a sufficient condition for validity in a purely static scene, or for pure antialiasing. However, for shading kernels that access neighboring pixels in dynamic scenes, like SSAO, we have to take into account that the shading of the current pixel can be affected by nearby moving objects, even if there is no disocclusion. Consider for example a scene where a box is lifted from the floor. The SSAO values of pixels in the contact shadow area surrounding the box change even if there is no disocclusion of the pixel itself.

The size of the neighborhood to check is equivalent to the size of the sampling kernel used for SSAO. Checking the complete neighborhood of a pixel would be prohibitively expensive, therefore we use sampling. Actually, it turns out that we already have a set of samples, namely the ones used for AO generation. That means that we effectively use our AO sampling kernel for two purposes: for computing the current contribution $C_t(p)$, and to test for validity. A pixel p is considered to be valid for a sample s_i if the relative positions of sample and pixel have not changed by more than ϵ (see Figure 5):

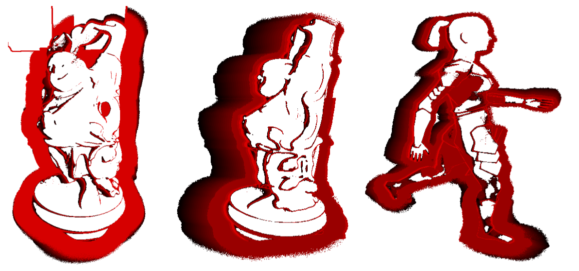

Figure 6: *This figure shows pixels recently invalidated by our algorithm. The left image depicts a rotation, the middle image a translation, and the right image an animated (walking) character. Red pixels were invalidated more recently than black ones while white pixels are sufficiently converged.*

$$\left||s_i - p| - |s_{iold} - p_{old}|\right| < \varepsilon, \tag{8}$$

where the reprojected position s_{iold} is computed from the offset vector stored for s_i (recall that the first rendering pass stores the offset vectors for all pixels in the frame buffer for later access by the SSAO shading pass). Note that we therefore use only those samples for the neighborhood test that lie in front of the tangent plane of p, since only those samples actually modify the shadow term. If this test fails for one of the samples evaluated for pixel p, then p is marked invalid and n_t is set to 0.

Theoretically we could also check if the angle between surface normal and vector to the sample point has changed by a significant amount from one frame to the next, and there could be practical cases where the vector length is not enough. However, this would require more information to be stored (i.e., the surface normal of the pixel in the previous frame), and in all our tests we found it sufficient to test for condition 8.

Note that in order to avoid one costly texture look up when fetching p_{old}, the required values for this test and for the ambient occlusion computation should be stored in a single render target.

In the case of dynamic objects, a large portion of the pixel information from the previous frame is often reusable. See Figure 6 for a visualization of pixels recently invalidated by our algorithm. E.g., for rotational movements, disocclusions occur on the silhouette, or at pixels that were hidden by creases or folds on the surface in previous frames. The amount of disocclusion depends on the speed of rotation, and is usually reasonably low for up to some degrees per frame.

3.5. Dealing with undersampled regions

Our algorithm ensures high-quality AO for sufficiently converged pixels. However, in screen regions that have been invalidated recently, the undersampling may result in temporal flickering. Disoccluded regions are often coherent and lead to distracting correlated noise patterns over several frames. We solve this by a new convergence-aware spatial filter.

3.5.1. Adaptive Convergence-Aware Spatial Filter

SSAO methods usually apply a spatial filtering pass after shading computations in order to prevent noise artifacts caused by insufficient sampling rates. We also apply spatial filtering, but only as long as the temporal coherence is not sufficient. Variants of the cross bilateral filter [ED04, BSD08] are typically used, where filtering over edges is avoided by taking the depth differences into account. Although this filter is not formally separable, in a real-time setting it is usually applied separately in x and y directions to make evaluation feasible. We follow this approach, too.

In contrast to previous approaches, we have additional information for the filter which can greatly reduce noise: the *convergence* $conv(p)$ of our AO values. Recently disoccluded pixels (e.g., in a thin silhouette region) can gather more information from nearby converged pixels than from other unreliable pixels. Furthermore, we apply the filter kernel directly to world-space distances like Laine et al. [LSK*07], which automatically takes depth differences into account, and can detect discontinuities in cases of high depth differences.

$$AO_{filt}(p) = \frac{1}{k(x,p)} \sum_{x \in F} g(|p-x|) conv(x) AO(x), \tag{9}$$

where x are the individual filter samples in the screen-space support F of the filter (e.g., a 9x9 pixel region), $k(x,p)$ is the normalization $\sum_{x \in F} g(|p-x|) conv(x)$, and g is a spatial filter kernel (e.g., a Gaussian). As a pixel gets more converged, we shrink the screen-space filter support smoothly using the shrinking factor s:

$$s(p) = \frac{\max(c_{adaptive} - conv(p), 0)}{c_{adaptive}}, \tag{10}$$

so that when convergence has reached $c_{adaptive}$, we turn off spatial filtering completely. We found the setting of $c_{adaptive}$ to be perceptually uncritical, e.g., setting $c_{adaptive}$ to 0.2 leads to unnoticeable transitions.

3.6. Optimizations

In this section we describe some optimizations of the core algorithm that allow for faster frame rates or better image quality.

3.6.1. Smooth invalidation

For moderately fast moving objects, the ambient occlusion is perceptually valid for more than one frame. If only the neighborhood changes (i.e., the second condition for invalidation from Section 3.4), a full invalidation of the current pixel is a waste of useful information, and noise and flickering artifacts may occur. Hence we rather clamp n_t to a low value instead of fully discarding the current solution by resetting n_t to 0. Note that in practice, we achieve this by reducing n_{max} for a single frame. We found an n_{max} value in the range of 32 and 64 to be a good tradeoff between full invalidation and no invalidation at all in many cases. Using this small optimization, the AO will appear smoother over time. Hence we denote it as *smooth invalidation*. From our experiments we can state that the resulting slight motion blur effect is less distracting than temporal flickering.

3.6.2. Adaptive sampling

While spatial filtering can reduce noise, it is even better to provide more input samples in undersampled regions. Or, to put it differently, once the AO has reached sufficient convergence, we can just reuse this solution and do not have to use as many samples as before. We adapt the number k of new AO samples per frame depending on convergence (note that these samples are completely unrelated to the screen-space samples used for spatial filtering in the previous section, where the kernel size is adapted instead of changing the number of samples). Since disoccluded regions are often spatially coherent (as can be seen in Figure 6), the required dynamic branching operations in the shader are quite efficient on today's graphics hardware.

Note that it is necessary to generate at least a minimum amount of samples for the same reasons that we clamp $n_t(p)$ in Equation 5, i.e., to avoid blurring artifacts introduced by bilinear filtering. Furthermore, a certain number of samples is required for detecting invalid pixels due to changing neighborhoods (Section 3.4.2). In order to introduce a minimum amount of branching, we chose a simple two-stage scheme, with k_1 samples if $conv(p) < c_{spatial}$ and k_2 samples otherwise (refer to Table 1 for a list of parameters actually used in our implementation).

parameter name	value
Initial samples k_1	32
Converged samples k_2	8–16
Threshold $c_{adaptive}$	0.2
Threshold $c_{spatial}$	0.3
Threshold c_{rot}	0.5
n_{max}	500–1500
Filter width F	5x5

Table 1: *Recommended parameters for the TSSAO algorithm.*

3.6.3. Frame buffer borders

A problem inherent in SSAO is the handling of samples that extend beyond the frame buffer borders. As there is no best method, we settle for reusing the values that appeared on the border by using clamp-to-edge. To avoid artifacts on the edges of the screen due to the missing depth information, we can optionally compute a slightly larger image than we finally display on the screen – it is sufficient to extend about 5–10% on each side of the screen depending on the size of the SSAO kernel and the near plane [BS09]. Note that this is a general problem of SSAO and not a restriction caused by our algorithm. Because these border samples carry incorrect information that should not be propagated, we detect those samples that were outside the frame buffer in the previous frame using our invalidation scheme (see Section 3.4).

3.6.4. Noise patterns

As in most AO approaches, we rotate the sampling pattern by a different random vector for each input pixel. However, this leads to a surprisingly large performance hit, supposedly due to texture cache thrashing [SW09]. Therefore we turn off the rotation once convergence has reached a certain threshold c_{rot}.

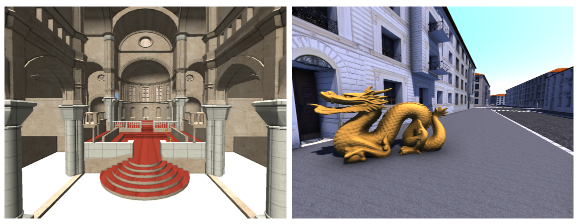

Figure 7: *Used test scenes: Sibenik cathedral (7,013,932 vertices) and Vienna (21,934,980 vertices).*

4. Results and implementation

We implemented the proposed algorithm in OpenGL using the Cg shading language, and tested it on two scenes of different characteristics (shown in Figure 7), the Sibenik cathedral and the Vienna city model. Both scenes were populated with several dynamic objects. The walkthrough sequences taken for the performance experiments are shown in the accompanying videos. For all our tests we used an Intel Core 2 processor at 2.66 GHZ (using 1 core) and an NVIDIA GeForce 280 GTX graphics board. The resolution of our render targets is either 1600x1200, 1024x768, or 800x600. To achieve sufficient accuracy in such large-scale scenes like Vienna, we use 32 bit depth precision. Both the ambient occlusion buffer and the SSAO texture are 32 bit RGBA render targets. In practical applications, SSAO is often computed on a half-resolution render target, and then upsampled for the final image using a spatial filter [FM08]. Hence we also tested the performance of the algorithm when using this common acceleration method. Note that the result images were all computed using full resolution render targets.

To prove that our algorithm is mostly independent of the SSAO generation method, we implemented both the method of Ritschel et al. [RGS09], and the algorithm introduced by Fox and Compton [FC08]. The latter algorithm uses screen-space sampling like Shanmugam and Arikan [SA07], and is more of an artistic approach and less physically motivated. In our opinion it preserves more small details when using large kernels for capturing low-frequency AO. In order to ensure that the sampling kernel will always cover the same size in world space, we implemented the following extension to the method: we multiply the initial kernel radius with the w coordinate from the perspective projection of the current pixel, i.e., the perspective foreshortening factor. As a result we obtain AO values that are invariant to zooming. However, this method is prone to reveal the underlying tesselation, therefore we do not count samples at gracing angles of the hemisphere (i.e., where the cosine is smaller than an ϵ)) as a remedy. For both methods, we do not count samples where the distance from the pixel center to the intersection point with the depth buffer is more than 2 times the length of the maximum sample radius. This heuristics has the purpose of avoiding incorrect shadowing of distant and disconnected surfaces caused by objects in the foreground, while still correctly accounting for occlusion in the vicinity of the currently shaded point.

The TSSAO algorithm has a couple of parameters used to control temporal refinement, which we list in Table 1 together with some recommended values. In all results we applied the optimizations discussed in Section 3.6. TSSAO uses 8 samples for Vienna respectively 16 samples for Sibenik when converged, and 32 samples otherwise. SSAO always uses 32 samples.

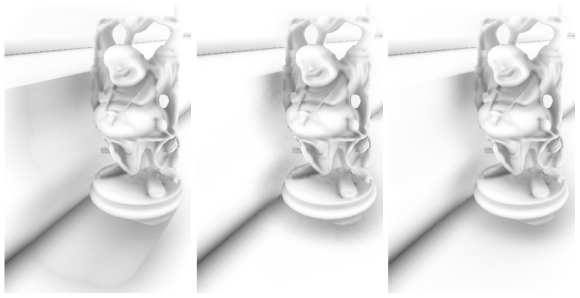

Figure 8: *Translational motion of the Happy Buddha model. (left) If we check for disocclusions of the current fragment only, we get strong artifacts due to motion. (middle) A full invalidation removes these artifacts, but there is some noise in the shadow regions. (right) Smooth invalidation: Assigning a small weight to the invalidated pixels allows to make better use of the temporal coherence.*

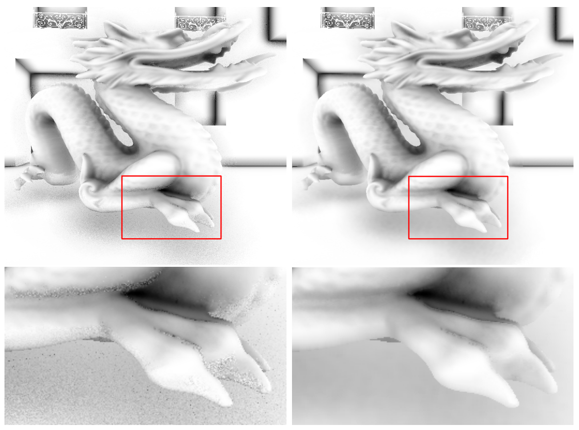

Figure 9: *Rotational movement using TSSAO. (Left) Without filter. (Right) Using our filter. Note how the filter is only applied in the noisy regions, while the rest stays crisp. Closeups of the marked regions are shown in the bottom row.*

In the two walkthrough sequences for Vienna and Sibenik, we aimed to consider the major cases of motion that occur in real-world applications, as can be seen in the accompanying videos. We included rigid dynamic objects as well as animated characters with deforming parts. All videos were shot with vertical refresh synchronization on, and encoded in 30 FPS. Note that the convergence shown in the video does not fully correspond to the convergence rate in real time: In sequences where more than 30 FPS are reached (the absolute minimum for games) the TSSAO method would converge faster. The video compression posed major problems. Quantization artifacts appear in the videos which cannot be seen in the live application or the uncompressed video, resulting in dark shades from contact shadows which are dragged behind, and slightly frayed shadow boundaries. This happened even using the high-quality H.264 codec.

Figure 8 demonstrates the importance of our novel invalidation scheme on a scene with a translational movement. Only checking disocclusions but not the pixel neighborhood for invalidation causes artifacts visible as wrong contact shadows (left image). Pixels that are not longer in shadow due to the movement can be detected by checking the pixel neighborhood (middle image). A smooth invalidation further improves the quality of animated sequences, effectively reducing the noise at the transitions between the silhouettes of moving objects and the background (right image).

Figure 9 depicts the Stanford Dragon rotating above a floor plane in the Vienna scene. It shows the influence of the adaptive convergence-aware filter on the quality of the TSSAO solution. Note that the filter blurs only the contact shadow region where temporal coherence is low, while the rest stays crisp.

In terms of visual image quality, TSSAO achieves better results than SSAO in all our tests. It corresponds to at least

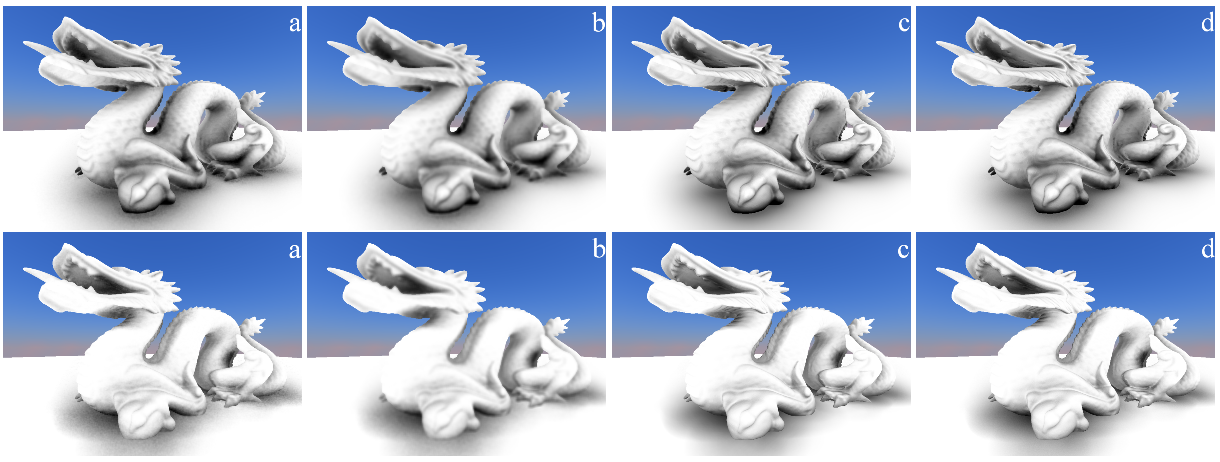

Figure 10: *From left to right: SSAO without temporal coherence using 32 samples, with a (a) weak, (b) strong blur. (c) TSSAO using 8 - 32 samples per frame. (d) Reference solution using 480 samples. The results are shown for the methods of Fox and Compton (top row) and Ritschel et al. (bottom row). All at 1024x768 resolution and using 32 bit precision render targets.*

a 32 sample SSAO solution (since 32 samples are used for disocclusions), while the converged state takes up to several hundred (n_{max}) samples into account. However, we have to keep in mind that using the smooth invalidation optimization causes the algorithm to deviate from a correct solution for the benefit of a better visual impression. Note that a similar quality SSAO solution would be prohibitively slow. As can be seen in Figure 1 (using the method of Fox and Compton) and Figure 10 (using either the method of Fox and Compton or the method of Ritschel et al.), and also in the accompanying videos, TSSAO provides finer details and less noise artifacts, while at the same time being faster. We compare TSSAO to SSAO with a weak and a strong blur filter, which gives a high respectively low weight to discontinuities. Furthermore, we compare TSSAO to a reference solution using 480 samples per frame – which was the highest number of samples our shader could compute in a single frame. Notice that the TSSAO method is very close to the reference solution, to which it converges after a short time.

Figure 11 shows the Sibenik cathedral populated with the Bunny model at resolution 1600x1200. While SSAO without temporal coherence works quite well, the TSSAO algorithm provides good quality even for fine details in the background. Also, the TSSAO algorithm maintains a better performance at such high resolutions.

The videos show that objects with deforming parts like the Paladin character in Figure 6 can be handled quite well by our algorithm. Also, the algorithm works sufficiently well even for fast moving objects, as can be seen in the video that shows all sorts of movement with a static camera. Although there is increased amount of blur for dynamic motions compared to the reference solution, using temporal coherence improves the image quality compared to plain SSAO. It can be seen that TSSAO effectively reduces flickering artifacts in animated sequences. The visual refinement of the SSAO solution over time is almost unnoticeable due to the adaptive filter.

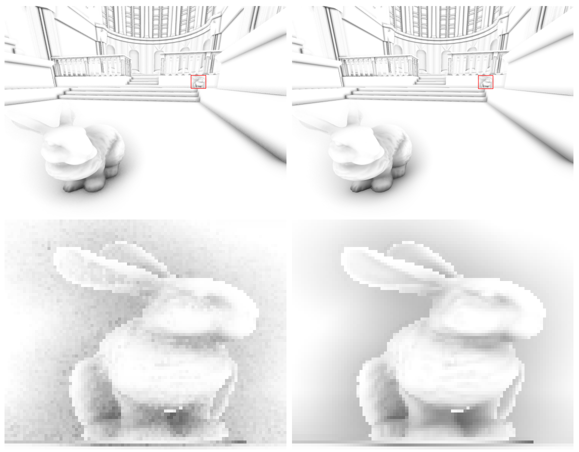

Figure 11: *Sibnik cathedral with the Bunny model at resolution 1600x1200, and extreme closeup at the distant Bunny (using the method of Fox and Compton). (left) SSAO using 32 samples without temporal coherence at 12 FPS, (right) TSSAO using 8 - 32 samples per frame at 26 FPS.*

Figure 12 shows a comparison to a ray-traced reference solution. The reference solution (middle) uses the Mental Ray ambient occlusion shader with 256 hemispherical samples and a linear falloff function. TSSAO is shown using the method of Ritschel et al. (left) and the method of Fox and Compton (right), computing only 8 new samples per frame

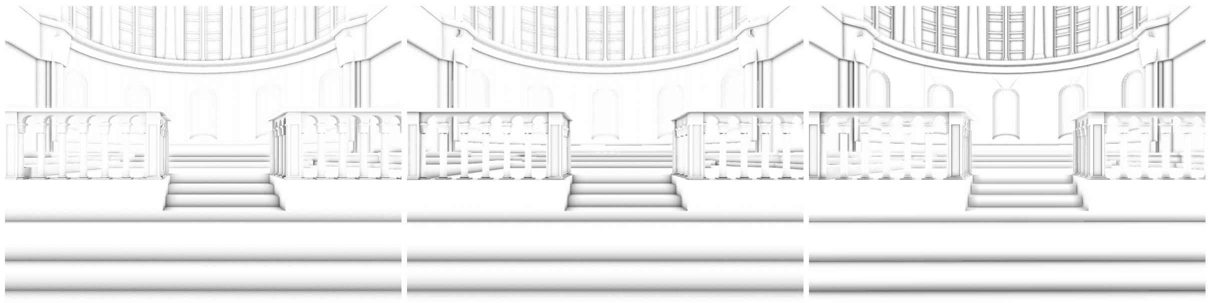

Figure 12: *Comparison of a ray traced solution using 256 samples (middle) with TSSAO using 8 samples per frame (initially 32) to the method of Ritschel (left), and the method of Fox and Compton (right), in the Sibenik cathedral.*

(32 initially). Note that for Ritschel et al., we weight samples equally, which corresponds to a linear falloff equivalent to the Mental Ray solution, and use the same maximum radius as for the ray-traced solution. The SSAO solutions exhibit some minor differences to the reference solution (e.g., at the complex geometry of the balustrade), and the method of Fox and Compton overemphasizes the underlying geometric structure. Otherwise the visual quality of both solutions is mostly comparable to the reference solution.

Table 2 shows average timings of our walkthroughs, comparing our method (TSSAO) with SSAO without temporal coherence and the performance-baseline method: deferred shading without SSAO. In all results we applied the optimizations discussed in Section 3.6. TSSAO uses 8 respectively 16 samples when converged and 32 otherwise for Vienna and for Sibenik, whereas SSAO always uses 32 samples. The cost of the SSAO/TSSAO algorithms (difference in frame times to baseline) is relatively independent of the scene complexity, and scales with the number of pixels in the frame buffer. TSSAO is always faster than SSAO when using the same number of samples, for full and for half resolution ambient occlusion buffers. Note that TSSAO does not apply spatial filtering or rotate the sampling filter kernel with the noise patterns after convergence has been reached.

Figure 13 shows the frame time variations for both walkthroughs. Note that online occlusion culling [MBW08] is enabled for the large-scale Vienna model, and thus the frame rate for the baseline deferred shading is quite high for such a complex model. The framerate variations for TSSAO stem from the fact that the method generates adaptively more samples for recently disoccluded regions, which can be quite a lot if we have dynamic objects that are large in screen space. For closeups of dynamic objects, the frame times of TSSAO are almost the same as the frame times of SSAO. For more static parts of the walkthroughss, TSSAO is significantly faster.

Scene	Vienna (21,934,980 vertices)		
resolution	SSAO	TSSAO	Deferred
1600x1200	14 FPS	29 FPS	73 FPS
1024x768	30 FPS	51 FPS	97 FPS
800x600	42 FPS	63 FPS	102 FPS
800x600 halfres	65 FPS	77 FPS	
Scene	Sibenik (7,013,932 vertices)		
resolution	SSAO	TSSAO	Deferred
1600x1200	10 FPS	12 FPS	38 FPS
1024x768	21 FPS	25 FPS	65 FPS
800x600	29 FPS	34 FPS	67 FPS
800x600 halfres	47 FPS	49 FPS	

Table 2: *Average timings for the two walkthrough sequences shown in the videos. We compare standard SSAO, our method (TSSAO), and deferred shading without SSAO as a baseline. For SSAO we used 32 samples in all scenes. For TSSAO we used 8 – 32 samples in the Vienna scene and 16 – 32 samples in the Sibenik cathedral. We used 32 bit depth precision for SSAO and TSSAO. Note that occlusion culling is used in the Vienna scene.*

4.1. Discussion and limitations

In the case of deforming objects, using a full invalidation on the deforming parts in each frame would cause temporal coherence to be constantly broken. Hence most of the visual improvements compared to conventional SSAO come from the smooth invalidation optimization (refer to the closeup of the cloak in Figure 14). Setting the smooth invalidation to a good value is quite important here – using this optimization too loosely can result in artifacts like a noticeable dark trail following moving objects.

There is definitely a limit to exploiting temporal coherence once the objects are moving or deforming too quickly. Also, invalidation may fail in cases of thin (with respect to the SSAO kernel size), quickly moving structures, which are missed by the invalidation algorithm. The legs of the Skeleton character in the Sibenik walkthrough are an example for

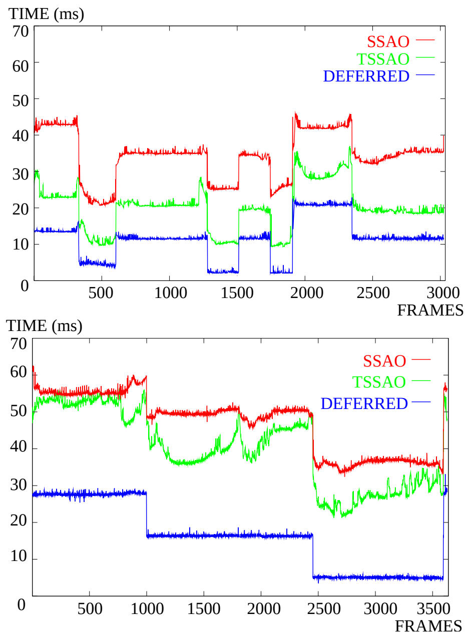

Figure 13: *Frame times of the Vienna walkthrough (with occlusion culling, top) and the Sibenik walkthrough (bottom) at resolution 1024x768 (using 32 bit precision render targets).*

a difficult case. Likewise, a very large kernel size can also cause problems with the invalidation, because the sampling could miss disocclusion events.

The main targets of this algorithm are real-time visualization and games. The adaptive sample generation cannot guarantee constantly better frame times than conventional SSAO, and there are definitely fluctuations in the frame time which could be disturbing in games. However, we have to keep in mind that games usually undergo extensive playtesting in order to avoid annoying frame rate drops. Faster SSAO in most frames is useful, because more time for other effects is available.

Also, note that the adaptive sample generation is just an optimization feature of the algorithm – without it we would fall back to the speed of conventional SSAO, but still have the improvements in image quality. Furthermore, the major purpose of this algorithm is to speed up the standard case, i.e., a moderate amount of dynamic movement. In these cases it has been shown to work well, although it may break down at extreme cases, which is probably true for most methods that depend on temporal coherence.

5. Conclusion and future work

We presented a screen-space ambient occlusion algorithm that utilizes reprojection and temporal coherence to produce high-quality ambient occlusion for dynamic scenes. Our algorithm reuses available sample information from previous frames if available, while adaptively generating more samples and applying spatial filtering only in the regions where not enough samples have been accumulated yet. We have shown an efficient new pixel validity test for shading algorithms that access only the affected pixel neighborhood. Using our method, such shading methods can benefit from temporal reprojection also in dynamic scenes with animated objects. While we restricted our results to show ambient occlusion, our algorithm can be seen as a cookbook recipe that is applicable to many other screen-space sampling algorithms with a finite kernel size.

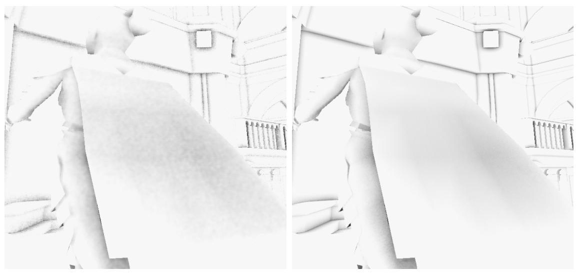

Figure 14: *Closeup of the deforming cloak of the Paladin character. (left) SSAO using 32 samples and (right) TSSAO using 8 – 32 samples. Although deforming objects are a difficult task for a temporal coherence algorithm, there is still some benefit from TSSAO in form of reduced surface noise (and reduced flickering artifacts when animated).*

In the future, we want to improve the smooth invalidation and put it on a more theoretical basis. In essence, smooth invalidation avoids the binary decision of fully discarding the previous AO stored in the ambient occlusion buffer. However, instead of the binary definition of invalidation, we should rather use a continous one. In particular, we estimate a *confidence value* between 0 and 1 to weight the confidence into the validity of the previous SSAO solution. This confidence can be steered by the length differences between the validation sample distances from Equation 8, which gives an indication of the magnitude of the change.

Furthermore, we want to explore other useful sampling-based techniques and how they can be enhanced with temporal coherence, and aim to derive general rules regarding invalidation that are applicable to different classes of problems. Also we plan to further investigate the problems with the blurriness that occurs because of bilinear filtering in combination with fast zooms. As these artifacts mainly occur on surfaces that are nearly perpendicular to the view direction, we want to investigate how they are related to the *projection error* known from shadow mapping.

References

[BS09] BAVOIL L., SAINZ M.: Multi-layer dual-resolution screen-space ambient occlusion. In *SIGGRAPH '09: SIGGRAPH 2009: Talks* (New York, NY, USA, 2009), ACM. 3, 7

[BSD08] BAVOIL L., SAINZ M., DIMITROV R.: Image-space horizon-based ambient occlusion. In *SIGGRAPH '08: ACM SIGGRAPH 2008 talks* (New York, NY, USA, 2008), ACM, pp. 1–1. 2, 3, 6

[Bun05] BUNNELL M.: *Dynamic Ambient Occlusion and Indirect Lighting*. Addison-Wesley Professional, 2005, ch. 14, pp. 223–233. 2

[CT81] COOK R. L., TORRANCE K. E.: A reflectance model for computer graphics. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM, pp. 307–316. 1

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. In *ACM Transactions on Graphics (Proceedings of Siggraph Conference)* (2004), vol. 23, ACM Press. 6

[FC08] FOX M., COMPTON S.: Ambient occlusive crease shading. *Game Developer Magazine* (March 2008). 2, 8

[FM08] FILION D., MCNAUGHTON R.: Starcraft ii: Effects & techniques. In *Proceedings of the conference on SIGGRAPH 2008 course notes, Advanced Real-Time Rendering in 3D Graphics and Games, Chapter 5* (2008), ACM Press, pp. 133–164. 1, 7

[KA06] KONTKANEN J., AILA T.: Ambient occlusion for animated characters. In *Rendering Techniques 2006 (Eurographics Symposium on Rendering)* (jun 2006), Eurographics. 3

[KA07] KIRK A. G., ARIKAN O.: Real-time ambient occlusion for dynamic character skins. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2007), ACM, pp. 47–52. 3

[KL05] KONTKANEN J., LAINE S.: Ambient occlusion fields. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM, pp. 41–48. 2

[Lan02] LANDIS H.: Production-ready global illumination. In *Proceedings of the conference on SIGGRAPH 2002 course notes 16* (2002). 1, 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proceedings of Eurographics Symposium on Rendering 2007* (2007), Eurographics Association, pp. 277–286. 6

[MBW08] MATTAUSCH O., BITTNER J., WIMMER M.: Chc++: Coherent hierarchical culling revisited. *Computer Graphics Forum (Proceedings Eurographics 2008) 27*, 2 (Apr. 2008), 221–230. 10

[Mit07] MITTRING M.: Finding next gen - cryengine 2. In *Proceedings of the conference on SIGGRAPH 2007 course notes, course 28, Advanced Real-Time Rendering in 3D Graphics and Games* (2007), ACM Press, pp. 97–121. 1, 2, 3

[MMAH07] MALMER M., MALMER F., ASSARSSON U., HOLZSCHUCH N.: Fast precomputed ambient occlusion for proximity shadows. *journal of graphics tools 12*, 2 (2007), 59–71. 2

[NSL*07] NEHAB D., SANDER P. V., LAWRENCE J., TATARCHUK N., ISIDORO J. R.: Accelerating real-time shading with reverse reprojection caching. In *Proceedings of the 22nd ACM SIGGRAPH/EUROGRAPHICS symposium on Graphics Hardware 2007* (Aire-la-Ville, Switzerland, Switzerland, 2007), Eurographics Association, pp. 25–35. 3, 4, 5

[PG04] PHARR M., GREEN S.: *Ambient Occlusion*. Addison-Wesley Professional, 2004, ch. 14, pp. 279–292. 2

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2009), ACM, pp. 75–82. 2, 3, 8

[Ros07] ROSADO G.: *Motion Blur as a Post-Processing Effect*. Addison-Wesley Professional, 2007, ch. 27, pp. 575–576. 3

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 977–986. 2

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on gpus. In *Symbosium of Interactive 3D graphics and games* (2007), pp. 73–80. 2, 8

[Shi05] SHISHKOVTSOV O.: *Deferred Shading in S.T.A.L.K.E.R.* Addison-Wesley Professional, 2005, ch. 2, pp. 143–166. 2

[SJW07] SCHERZER D., JESCHKE S., WIMMER M.: Pixel-correct shadow maps with temporal reprojection and shadow test confidence. In *Rendering Techniques 2007 (Proceedings Eurographics Symposium on Rendering)* (June 2007), Eurographics, Eurographics Association, pp. 45–50. 3, 4, 5

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. *SIGGRAPH Computer Graphics 24*, 4 (1990), 197–206. 2

[SW09] SMEDBERG N., WRIGHT D.: Rendering techniques in gears of war 2, 2009. 3, 7

[WH00] WANG X., HICKERNELL F. J.: Randomized halton sequences. *Mathematical and Computer Modelling 32* (2000), 2000. 3

[YNS*09] YANG L., NEHAB D., SANDER P. V., SITTHI-AMORN P., LAWRENCE J., HOPPE H.: Amortized supersampling. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia 2009) 28*, 5 (2009), 135. 4

[ZIK98] ZHUKOV S., IONES A., KRONIN G.: An ambient light illumination model. In *Rendering Techniques* (1998), pp. 45–56. 2