Vision, Modeling, and Visualization (2011)
Peter Eisert, Konrad Polthier, and Joachim Hornegger (Eds.)

Bent Normals and Cones in Screen-space

O. Klehm[1,2] T. Ritschel[2,3] E. Eisemann[3] H.-P. Seidel[1]

[1]MPI Informatik [2]Intel Visual Computing Institute [3]Télécom ParisTech / CNRS-LTCI

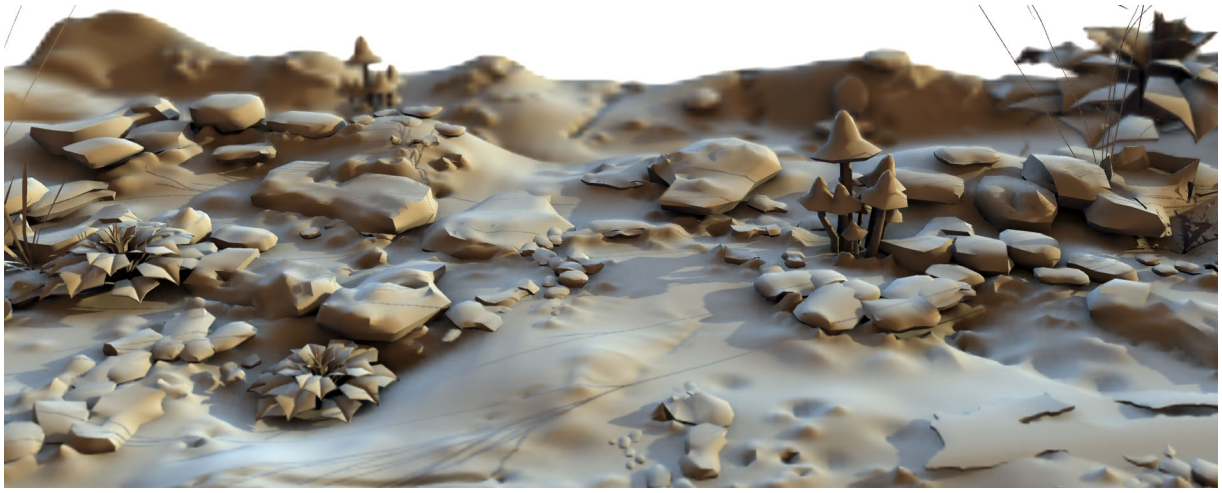

Figure 1: *Possible game scenario with lighting using bent normals:* 2048×1024 *pixels, 60.0 fps, including direct light and DOF on an Nvidia GF 560Ti. Environment mapping produces natural illumination, while bent normals cause colored shadows.*

Abstract

Ambient occlusion (AO) is a popular technique for real-time as well as offline rendering. One of its benefits is a gain in efficiency due to the fact that occlusion and shading are decoupled which results in an average occlusion that modulates the surface shading. Its main drawback is a loss of realism due to the lack of directional occlusion and lighting. As a solution, the use of bent normals was proposed for offline rendering. This work describes how to compute bent normals and bent cones in combination with screen-space ambient occlusion. These extensions combine the speed and simplicity of AO with physically more plausible lighting.

The definitive version is available at diglib.eg.org.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—I.337 [Computer Graphics]: Color, Shading, Shadowing and Texture—

1. Introduction

Ambient occlusion (AO) is a nonphysically-based approximation of environmental lighting and gives an impression of global illumination. It achieves high performance by computing an average occlusion that is used to modulate surface shading without respecting the directionality of light. Consequently, the approximation is not always acceptable, for example when used in conjunction with environment mapping.

Landis addressed the directionality issue by introducing so-called bent normals [Lan02]. While AO stores an average occlusion, bent normals are modified normals bent according to an estimate of the direction that is most unobstructed, i. e., the average unblocked direction. Bent normals can then be used to compute an illumination response that is closer to an actual sampling of the environment map. A useful property of bent normals is that they can usually be easily integrated in rendering engines. Basically, they can directly be used for

shading just like standard normals, but result in an improved accuracy.

One popular incarnation of AO is its implementation in screen-space (SSAO) [Mit07, SA07, BSD08, RGS09, LS10, HBR*11]. In this paper, we will describe a technique to extend SSAO. Our idea is to keep the simplicity of SSAO by relying on a screen-space solution, but to add the advantages of bent normals. Furthermore, we describe a new entity: bent cones that capture the distribution of unoccluded directions by storing the average direction and their variance. Hereby, rendering quality and precision are improved.

2. Related Work

The remarkable importance of occlusion as a visual cue was first described by Langer and Zucker [LZ94]. They found that the human ability to understand shape from shading combines common directional lighting and shadows, as well as soft ambient lighting and *ambient occlusion* (AO) found on a cloudy day.

Miller [Mil94] introduced accessibility shading that was modified by Zhukov et al. [ZIKS98] to AO. It is an approximation to the rendering equation [Kaj86] for the outgoing light L_o at a location $\mathbf{x}$ in direction ω_o and assumes environmental illumination and diffuse surfaces:

$$L_o(\mathbf{x},\omega_o) := \frac{1}{\pi}\int_{\Omega^+} L_i(\mathbf{x},\omega_i)\, V(\mathbf{x},\omega_i)\,(\mathbf{n}\cdot\omega_i)\,\mathrm{d}\omega_i,$$

where Ω^+ is the upper hemisphere, $\mathbf{n}$ is the surface normal, L_i the incoming light and V the visibility function that is zero when a ray is blocked and one otherwise. AO assumes that V can be moved outside the integral, as

$$L_o(\mathbf{x}) \approx AO(\mathbf{x})\,\frac{1}{\pi}\int_{\Omega^+} L_i(\mathbf{x},\omega)\,(\mathbf{n}\cdot\omega)\,\mathrm{d}\omega,$$

where

$$AO(\mathbf{x}) := \frac{1}{2\pi}\int_{\Omega^+} V(\mathbf{x},\omega)\,\mathrm{d}\omega. \tag{1}$$

Basically, AO decouples shading and the directional dependency of visibility: light from all directions is equally attenuated by the average blocking over all directions. Whether the AO assumptions apply depends on the lighting and materials [YCK*09].

In the production setting considered by Landis [Lan02], ray-tracing was used to compute the hemispherical integral of AO using Monte-Carlo integration: a set of rays R is cast into the hemisphere to evaluate V, and the result is averaged:

$$AO_{MC}(\mathbf{x}) := \frac{1}{|R|}\sum_{\omega\in R} V(\mathbf{x},\omega) \approx AO(\mathbf{x}).$$

The idea of *bent normals* dates back to Landis [Lan02] where it was proposed as a generalization of AO. Bent normals are the mean free direction scaled by the mean occlusion and are used for shading instead of the surface normals. Different from AO, their definition includes the direction ω inside the integral:

$$N(\mathbf{x}) := \frac{1}{\pi}\int_{\Omega^+} V(\mathbf{x},\omega)\,\omega\,\mathrm{d}\omega. \tag{2}$$

For lighting computations, bent normals simply replace the surface normal and the visibility term:

$$L_o(\mathbf{x}) \approx \frac{1}{\pi}\int_{\Omega^+} L_i(\mathbf{x},\omega)\,(N(\mathbf{x})\cdot\omega)\,\mathrm{d}\omega.$$

The Monte-Carlo computation of bent normals $N_{MC}(\mathbf{x})$ requires to multiply visibility with the direction, which is as computationally simple and efficient as AO_{MC} alone. Different to common normals, bent normals are not normalized and include AO in their length.

Due to its success in offline production, considerable effort was made to support AO in interactive rendering. Pharr and Green [PG04] used several shadow maps to compute ambient occlusion for a single static object and mention that their method generalizes to bent normals as well. Bunnell [Bun05] compute AO in dynamic scenes using a finite-element approach that approximates the surfaces as a hierarchy of discs. AO occlusion is gathered at vertices or pixels in a fast GPU multipole approach. The incorrect linear summation of visibility of different occluders can be improved with a multi-pass approach. The method can compute bent normals and even full indirect illumination. Rigidly transformed static objects can benefit from a pre-computation that stores AO in a discrete 3D grid [KL05, MMAH07]. In contrast, SSAO works for dynamic scenes and received much interest [Mit07, SA07, BSD08, LS10]. It has many desirable properties, including output-sensitivity (AO is computed for visible pixels only) and it avoids intermediate data structures. Further, computations are simplified, no assumptions are made about the geometry besides the requirement to render the scene into a deferred shading buffer [ST90]. While screen space allows for directional effects as well [RGS09], computing bent normals in screen space was not proposed to our knowledge. Nonetheless, working in screen space is inherently approximate because of incomplete information, but its efficiency made SSAO a core component in most modern real-time rendering engines.

In its most basic form [Mit07], SSAO (Fig. 2) is computed for a pixel i by comparing its camera space location $\mathbf{x}_i$ with other pixel's camera space position $\mathbf{x}_j$ in a pixel neighborhood $P_i \subset \mathbb{N}$:

$$AO_{ss}(i) := \frac{1}{|P_i|}\sum_{j\in P_i} d(\Delta_{ij}) \approx AO_{MC}(\mathbf{x}_i), \tag{3}$$

where $\Delta_{ij} := \mathbf{x}_j - \mathbf{x}_i$. One possible implementation of $d(\Delta)$ is $d_x(\Delta)$, defined as:

$$d_x(\Delta) := \begin{cases} 0 & \text{if } \Delta.z > 0 \\ 1 & \text{else} \end{cases}.$$

The pixels P_i to compute occlusion at pixel i are usually

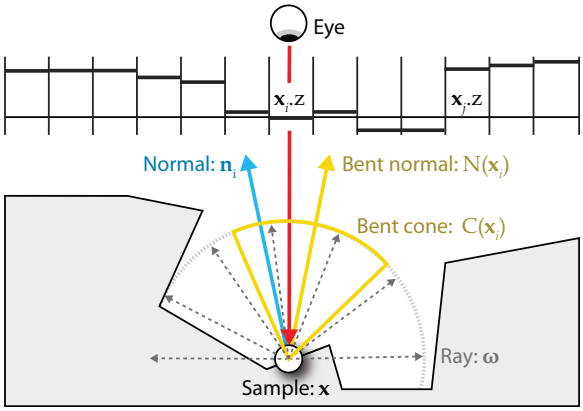

Figure 2: *Illustration of our main variables*

chosen to be neighboring pixels in screen space [Mit07], or the projection of a set of sample points near to $\mathbf{x}_i$ in world space [SA07]. Improvements are possible by accounting for outliers that should not cast shadows [SA07, RGS09, LS10], and by including the normal at the i-th pixel [RGS09]:

$$d_{\mathrm{xn}}(\Delta,\mathbf{n}) := \begin{cases} 0 & \text{if } z_{\max} > \Delta.\mathrm{z} > 0 \text{ and } (\Delta\cdot\mathbf{n}) > 0 \\ 1 & \text{else} \end{cases}.$$

Instead of gathering occlusion from a neighborhood P, Shanmugam and Arikan [SA07], as well as McGuire [McG10] use a splatting approach to distribute AO from surfaces to pixels.

The underlying assumption of SSAO is, that summing occlusion of "nearby" occluders approximates true visibility. However, visibility is a nonlinear effect: two occluders behind each other in one direction do not cast a shadow twice. Hence, other approaches [OS07] find the correct occlusion for a set of directions in screen space via ray marching in the depth buffer. Alternatively, one can also compute a horizon angle [Max88, BSD08]. As an in-between solution, others [SKUT*09, LS10] considered the free volume over a height field of depth values inside a sphere around a pixel as a better approximation.

3. Our Technique

In this section, we will describe our approach for interactive AO and GI based on screen-space bent normals. We will first introduce their computation (Sec. 3.1), and then generalize them to bent cones (Sec. 3.2).

3.1. Bent normals

We will here describe the basic implementation along the lines of the original Crytek SSAO [Mit07]. There, the original continuous AO defined in world space (Eq. 1), was solved in a discrete way in screen space (Eq. 3). We will apply this idea to the continuous world-space bent normals (Eq. 2), and compute them in discrete screen space. To this end, we aggregate unoccluded directions Δ_{ij}:

$$N_{\mathrm{ss}}(i) := \Big(\sum_{j\in P_i} d(\Delta_{ij})\Big)^{-1} \sum_{j\in P_i} \frac{\Delta_{ij}}{|\Delta_{ij}|}\, d(\Delta_{ij}) \approx N(\mathbf{x}_i).$$

The basic principle behind our approach is that, when computing AO in screen space, the direction Δ_{ij} and its visibility d are known and can be used to accumulate a mean direction that defines the bent normal. Our technique is mostly orthogonal to the type of SSAO used, allowing for different implementations of d. The bent normal can then be used for lighting (Sec. 3.3) in place of the original normal.

3.2. Bent cones

Bent cones are bent normals augmented by an *angle*. Inspired by the estimation of the variance of a von Mises-Fisher distribution on spheres [MJ00], we compute the angle directly from the mean direction, which is the non-normalized bent normal:

$$C(i) := (1 - \max(0, 2\,|N_{\mathrm{ss}}(i)| - 1))\,\frac{\pi}{2}.$$

Hereby, the *variance* of the unoccluded direction is directly reflected by the length of the bent normal. The max-function ensures that the bent cones of unoccluded points exactly cover the hemisphere.

Bent normal and bent cone define a spherical cap of visibility, which allows us to use illumination methods that compute the incoming light inside a spherical cap (Sec. 3.3). Most importantly, instead of using the cone as a visibility approximation, we only restrict the directions from which we gather light. We still rely on AO to estimate the average visibility, which can be more complicated than a single cone. Thus, the cone might cover incorrect directions, but AO accounts for a plausible darkening.

3.3. Pre-convolved lighting

Bent normals and cones are most useful in combination with a shading model that computes the illumination coming from a set of directions, such as pre-convolved environment maps [HS99] or irradiance volumes [GSHG98].

Environment maps assume *distant* illumination, i. e., $L_{\mathrm{i}}(\mathbf{x},\omega_{\mathrm{i}}) \approx L_{\mathrm{i}}(\mathbf{x}',\omega_{\mathrm{i}})\ \forall \mathbf{x},\mathbf{x}' \in \mathbb{R}^3$. Dropping the spatial dependency on $\mathbf{x}$, we simply write: $L'_{\mathrm{i}}(\omega_{\mathrm{i}})$.

Pre-convolution of distant illumination computes a directional function $L_{\mathrm{p}}(\omega_{\mathrm{o}})$. For every outgoing direction ω_{o}, it convolves the distant illumination $L'_{\mathrm{i}}(\omega_{\mathrm{i}})$ and the geometric term (cosine of incoming light and normal):

$$L_{\mathrm{p}}(\omega_{\mathrm{o}}) := \frac{1}{\pi}\int_{\Omega^+} L'_{\mathrm{i}}(\omega_{\mathrm{i}})\,(\omega_{\mathrm{o}}\cdot\omega_{\mathrm{i}})\,\mathrm{d}\omega_{\mathrm{i}}.$$

To query this environment map, the bent normal is used in place of ω_{o}.

However, to be correct, pre-convolution has to assume no shadows or at least, that visibility can approximately be moved outside the integral. In practice, this means the lighting result is just multiplied with a shadow term (AO). We propose to include visibility inside the pre-convolution leading to a tri-variate function

$$L_c(\omega_o, \alpha) := t \frac{1}{\pi} \int_{\Omega^+} L_i'(\omega_i) \bar{V}(\omega_i, \omega_o, \alpha) (\omega_o \cdot \omega_i) \, d\omega_i,$$
$$t := (1 - \cos(\alpha))^{-1}$$

that stores the outgoing radiance for a bent cone in direction ω_o with angle α (Fig. 3). The function $\bar{V}$ returns one if ω_i and ω_o form an angle smaller than α and zero otherwise. Note, that by doing so, with increasing α the pre-convolved values get larger. As we do not use the bent cone as visibility approximation (cf. Sec. 3.2), we introduce the *normalization* term t. Thus, we get equal results L_c independent of α if L_i' is constant. The mean direction $N_{ss}(i)$ and angle $C(i)$ of the bent cone replace ω_o and α to look-up a convolved environment map.

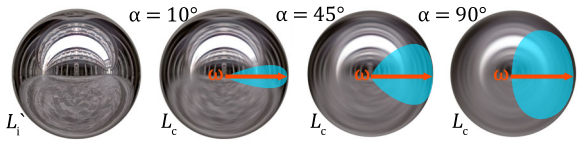

Figure 3: *Pre-convolution of distant lighting (left) into a series of triple products of light, BRDF, and visibility of cones with varying angle α (left to right: 10, 45, and 90 degrees).*

3.4. Geometric Term

We use a heuristic to apply the geometric term in our bent cones. It needs to be a part of the pre-convolution because the incoming light per direction is only known at this time. Correctly integrating the geometric term would be five-dimensional: 2D for the mean direction (bent normal), 1D for the angle of the cone, and 2D for the surface normal. We can approximate it by:

$$(\omega \cdot \mathbf{n}) \approx (\omega \cdot N(\mathbf{x}))(N(\mathbf{x}) \cdot \mathbf{n}).$$

If $(N(\mathbf{x}) \cdot \mathbf{n}) \approx 1$, $(\omega \cdot N(\mathbf{x}))$ approximates the correct geometric term very well, while $(N(\mathbf{x}) \cdot \mathbf{n})$ vanishes by definition. If $N(\mathbf{x})$ and $\mathbf{n}$ diverge, the heuristic returns near correct results for light directions close to the bent normal ($(\omega \cdot N(\mathbf{x})) \approx 1$), as $(\omega \cdot \mathbf{n}) \approx (\mathbf{n} \cdot N(\mathbf{x}))$. Additionally, the angle of the bent cone becomes smaller, such that the error for light directions within the cone remains small for all possible cones (cf. Fig. 4). The heuristic regards the original normal and should be preferred to only using $(\omega \cdot N(\mathbf{x}))$. Further, it avoids a 5D pre-convolution and gives correct results for unoccluded points (where $\mathbf{n} = N(\mathbf{x})$).

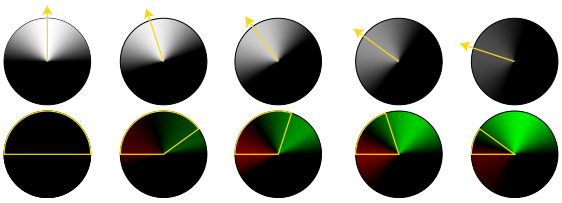

Figure 4: *2D examples of our approximation of the geometric term. The top rows shows the evaluated term, the bottom shows the error (compared to a normal with direction upwards; left). Directions with overestimation are red, while underestimation is green compared to the correct geometric term. The bent normal is increasingly rotated in the images to the right. The heuristic ensures correct weighting for unoccluded directions (bent normal and normal are equal; left). The more normal and bent normal diverge (images to the right), the greater the error becomes. However, at the same time, the angle of the bent cone becomes smaller and the error within the cone remains low. Note, that the bent cone can have a different size than shown.*

4. Implementation

Variants For our experiments, we combined the approach with several methods: `Crytek2D` [Mit07] uses a 2D sampling pattern to distribute samples in screen space. For higher quality, we perform jittering and reject samples outside the unit disc. `Crytek3D` [Mit09] uses a 3D sampling pattern to distribute samples in space that are projected to screen space. We generate samples on the unit hemisphere, optionally add ray-marching steps to increase the visibility test quality, and apply randomized offsets in each direction to turn aliasing into more eye-pleasing noise. `HBAO` [BSD08] uses random 3D directions orthogonal to $\mathbf{n}$ that are marched to find the highest horizon angle. The samples are distributed on the unit disc and transformed according to $\mathbf{n}$. Again, randomized offsets are used. We rely on OpenGL 3 and store all sample patterns in uniform variables.

Interleaved sampling [KH01] allows for using only a low number of samples per pixel, which is important for performance. Here, using a randomization pattern of size $m \times m$ applies different sample sets for neighboring pixels, which prevents banding artifacts and turns under-sampling in noise. To blur the noisy AO and bent normals a geometry-aware blur [LSK*07] regarding position and normal discontinuities is used. When applying an 16×16 interleaved sampling pattern and using 2×2 super-sampling for anti-aliasing, a single sample is enough for AO. Bent cones require at least 4 samples. Instead of interpolating the bent normal directly, we compute the difference between normal and bent normal and blur this difference. The result is added to the original high-frequency normals. Hereby, the details in the normal field itself are preserved and the bent information is propagated.

Pre-convolved lighting For pre-convolved environment maps, we store L_p as a floating point-texel cube texture. Pre-convolution including visibility is tri-variate and can be stored most efficiently using the recent cube map array extension of OpenGL. This extensions stores an array of cube maps, that can be accessed with a direction and an index. We discretize α to 8 levels and apply linear filtering between the levels, which has shown to be sufficient. More efficiently, in the context of glossy reflections Kautz and McCool [KM00] propose to store the third dimension in cube map MIP levels.

5. Results

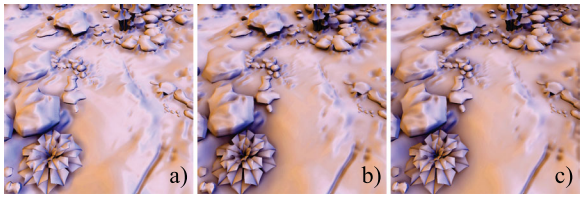

Figure 5: *Our approach is largely orthogonal to the particular SSAO implementation used: two-dimensional sampling (`Crytek2D`,* (a)*), three-dimensional sampling (`Crytek3D`,* (b)*), horizon-based ray-marching (`HBAO`,* (c)*).*

In this section we present our results that increase accuracy while adding only a small performance penalty compared to SSAO. For the following images, we used the 3D sampling pattern with three ray-marching steps, which best approximates the integrals Eq. 1 and 2.

A performance breakdown of Fig. 1 at resolution 2048×1024 on an Nvidia Geforce 560Ti is as follows: 4.2 ms for AO and bent normals, 4.7 ms for geometry-aware blurring, 1.5 ms for the deferred shading buffer, and 1.6 ms for direct light with PCF shadows. The overhead for bent normals compared to AO only using the same number of samples is 7 % for the computation and 25 % for the blur, in total less than 11 %.

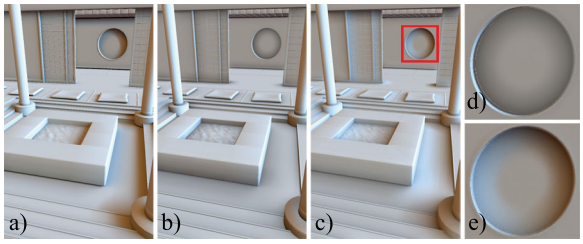

Figure 6: *Lighting using ray-tracing* (a)*, SSAO* (b)*, and our screen-space bent normals* (c)*. Details have more articulated lighting when AO* (d) *is enhanced by our bent normals* (e)*.*

Lighting using our bent normals is closer to a reference solution than AO alone (Fig. 6). It can be seen how AO decouples lighting and visibility, which leads to grey shadows that are perceived as a change of reflectance, rather than an effect of lighting. Our screen-space bent normals are similar to real bent normals (Fig. 7) which leads to only a small difference between lighting using accurate bent normals and our bent normals. The reference solutions were created using ray-tracing, using several hundred samples.

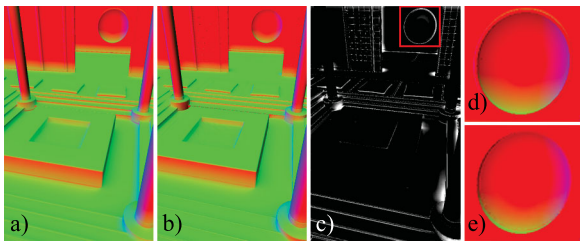

Figure 7: *Ray-traced bent-normals* (a)*, our screen-space bent normals* (b)*, and the 8× angle difference* (c)*. Ray-traced bent normal details* (d) *are similar to our approximation* (e)*.*

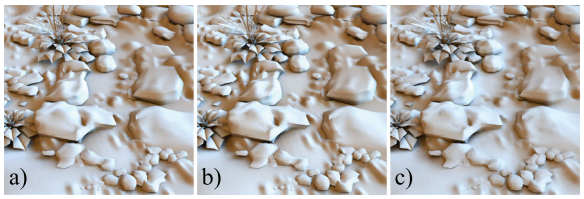

Figure 8: *Environment map importance sampling (IS)* (a)*, SSDO [RGS09]* (b)*, and SS bent cones* (c)*. IS requires 32 samples per pixel (18.8 ms), SSDO 16 samples (12.5 ms), and bent cones only 8 samples (6.0 ms) to achieve the results (1024×1024 resolution; 2×2 super-sampling). All techniques use three ray-marching steps per sample for the SS visibility tests, a 8×8 interleaved sampling pattern, and an according geometry-aware blur.*

Bent cones improve on bent normals as they do not gather light from all directions in the upper hemisphere of a point. Ritschel et al. [RGS09] regard the directionality of light by applying a brute-force sampling for unblocked directions (SSDO). This implies an additional texture lookup, which results in an overhead of about 10 % in execution time. Even more important, noise appears if the sample count is not increased. In our tests, we required 16 to 24 samples per-pixel in combination with interleaved sampling to avoid visible artifacts. Our bent cones produce smooth results with 8 samples with similar quality. SSAO returns good results with such a low number of samples and the cone does not need to be very accurate (Fig. 8). Additionally, we compared our technique to importance sampling of the environment map (IS), which requires at least 32 samples to achieve equal quality. Note, in contrast to sampling-based techniques, bent cones are not able to handle high-frequency illumination.

Thus, we used low-frequency illumination changes only to allow for a fair comparison. Else, bent cones simply blur the high-frequencies, SSDO requires many more samples (up to hundreds), and IS at least 64 samples.

6. Discussion and Conclusion

Screen-space bent normals improve accuracy of shading without imposing much additional computation costs. Bent cones are easy to compute and regard the directionality of light by reducing the directions from which light is gathered. If the actual visibility configuration is not well approximated by a cone, they fall back to lighting with bent normals, possibly adding light from blocked directions (Fig. 9). However, our

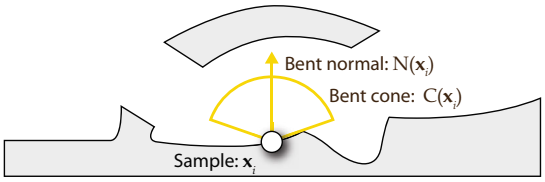

Figure 9: *Bent normals and cones assume blockers to form a simple horizon. In case of more complex occlusion such as depicted here, the direction can diverge.*

approach shares these limitations with previous SSAO techniques. Our approach can extend several different SSAO techniques and delivers higher speed at a similar quality as SSDO. In future work, we plan to investigate other representations of the occlusion function, specular BRDFs, new interpolation methods, a combination with irradiance volumes [GSHG98] for local pre-filtered directional occlusion, as well as the use of Gaussians to approximate visibility [GKMD06, GKD07].

Acknowledgments We thank the anonymous reviewers for their valuable comments and suggestions. This work was partly funded by the Intel Visual Computing Institute at Saarland University and the ANR iSpace & Time of the French government.

References

[BSD08] BAVOIL L., SAINZ M., DIMITROV R.: Image-space horizon-based ambient occlusion. In *SIGGRAPH Talk* (2008). 2, 3, 4

[Bun05] BUNNELL M.: *Dynamic ambient occlusion and indirect lighting*, vol. 2. Addison Wesley, 2005, pp. 223–233. 2

[GKD07] GREEN P., KAUTZ J., DURAND F.: Efficient reflectance and visibility approximations for environment map rendering. *Computer Graphics Forum 26*, 3 (2007), 495–502. 6

[GKMD06] GREEN P., KAUTZ J., MATUSIK W., DURAND F.: View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proc. I3D* (2006), pp. 7–14. 6

[GSHG98] GREGER G., SHIRLEY P., HUBBARD P., GREENBERG D.: The irradiance volume. *IEEE Computer Graphics and Applications* (1998), 32–43. 3, 6

[HBR*11] HUANG J., BOUBEKEUR T., RITSCHEL T., HOLLÄNDER M., EISEMANN E.: Separable approximation of ambient occlusion. In *Short paper at Eurographics* (2011). 2

[HS99] HEIDRICH W., SEIDEL H.-P.: Realistic, hardware-accelerated shading and lighting. In *Proc. SIGGRAPH* (1999), pp. 171–178. 3

[Kaj86] KAJIYA J.: The rendering equation. *ACM SIGGRAPH Computer Graphics 20*, 4 (1986), 143–150. 2

[KH01] KELLER A., HEIDRICH W.: Interleaved sampling. In *Proc. EGWR* (2001), p. 269. 4

[KL05] KONTKANEN J., LAINE S.: Ambient occlusion fields. In *Proc. I3D* (2005), pp. 41–48. 2

[KM00] KAUTZ J., MCCOOL M. D.: Approximation of glossy reflection with prefiltered environment maps. In *Proc. Graphics Interface* (2000), pp. 119–126. 5

[Lan02] LANDIS H.: Production-ready global illumination. In *SIGGRAPH Course* (2002), pp. 87–102. 1, 2

[LS10] LOOS B. J., SLOAN P.-P.: Volumetric obscurance. In *Proc. I3D* (2010), pp. 151–156. 2, 3

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. EGSR* (2007), pp. 277–286. 4

[LZ94] LANGER M., ZUCKER S.: Shape-from-shading on a cloudy day. *JOSA A 11*, 2 (1994), 467–478. 2

[Max88] MAX N.: Horizon mapping: shadows for bump-mapped surfaces. *The Visual Computer 4*, 2 (1988), 109–117. 3

[McG10] MCGUIRE M.: Ambient occlusion volumes. In *Proc. I3D* (2010), pp. 12:1–12:1. 3

[Mil94] MILLER G.: Efficient algorithms for local and global accessibility shading. In *Proc. SIGGRAPH* (1994), pp. 319–26. 2

[Mit07] MITTRING M.: Finding next gen: CryEngine 2. In *SIGGRAPH Courses* (2007), pp. 97–121. 2, 3, 4

[Mit09] MITTRING M.: A bit more deferred – CryEngine 3. In *Triangle Game Conference* (2009). 4

[MJ00] MARDIA K. V., JUPP P. E.: *Directional Statistics*. John Wiley and Sons, 2000. 3

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

[OS07] OAT C., SANDER P. V.: Ambient aperture lighting. In *Proc. I3D* (2007), I3D '07, pp. 61–64. 3

[PG04] PHARR M., GREEN S.: *Ambient occlusion*, vol. 1. Addison Wesley, 2004, pp. 279–292. 2

[RGS09] RITSCHEL T., GROSCH T., SEIDEL H.-P.: Approximating dynamic global illumination in image space. In *Proc. I3D* (2009), pp. 75–82. 2, 3, 5

[SA07] SHANMUGAM P., ARIKAN O.: Hardware accelerated ambient occlusion techniques on GPUs. In *Proc. I3D* (2007), pp. 73–80. 2, 3

[SKUT*09] SZIRMAY-KALOS L., UMENHOFFER T., TÓTH B., SZÉCSI L., SBERT M.: Volumetric ambient occlusion. *IEEE Computer Graphics and Applications* (2009). 3

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3D shapes. *Comp. Graph. (SIGGRAPH) 24*, 4 (1990), 197–206. 2

[YCK*09] YU I., COX A., KIM M. H., RITSCHEL T., GROSCH T., DACHSBACHER C., KAUTZ J.: Perceptual influence of approximate visibility in indirect illumination. *ACM Trans. Applied Perception 6*, 4 (2009). 2

[ZIKS98] ZHUKOV S., IONES A., KRONIN G., STUDIO G.: An ambient light illumination model. In *Proc. EGSR* (1998), p. 45. 2