 https://doi.org/10.3311/PPee.20209

Periodica Polytechnica Electrical Engineering and Computer Science, 66(3), pp. 248–252, 2022

Accounting for Non-Uniform Ambient Lighting in Ray Traced Ambient Occlusion

András Fridvalszky[1*], Balázs Tóth[1]

[1] Department of Control Engineering and Information Technology, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
[*] Corresponding author, e-mail: fridvalszky@iit.bme.hu

Received: 22 March 2022, Accepted: 30 June 2022, Published online: 11 July 2022

Abstract

Ambient occlusion is a popular method to enhance the visuals of real-time applications. The generally accepted equation assumes that incoming ambient lighting is uniform in the enclosing hemisphere around the surface point. This assumption does not always hold and can cause artifacts. With screen-space techniques the artifact may not be present, because of the lack of information, but it becomes visible with more accurate methods. This is especially true for ray tracing based implementations which are widely used to create the ground truth image to measure other techniques. With recent advancements of the graphics hardware, they also found their way into real-time applications where these shortcomings were neither discussed nor addressed. In this paper we analyze the problem and demonstrate its presence in popular game engines. We also propose a ray tracing based extension to eliminate these artifact in a physically plausible way.

Keywords

ray tracing, ambient occlusion

1 Introduction

An important component to generate realistically looking images is to consider the effect of the ambient illumination. As the physically correct approach would require prohibitively expensive calculations, we need a flexible compromise. Such a compromise is the *local indirect lighting* approach that considers only a finite neighborhood of a shaded surface point during the illumination calculation. The *obscurances* method, which is also called the *ambient occlusion*, computes just how "open" the scene is in the neighborhood of a point and scales the ambient light accordingly. The openness of the point is proportional to the solid angle of directions where no nearby occluder exists.

Implementations of the *ambient occlusion* method are generally using multiple simplifications and approximations to calculate it more efficiently. Until recently, for real-time purposes various screen space techniques were the common choice. Ray traced ambient occlusion was only used to evaluate other techniques by comparing how they fared against it. Recently real-time ray tracing became a possibility and was used in hybrid renderers to enhance the visual quality with ray traced effects.

The fundamental assumption of these methods is that the incoming ambient light is uniform from all directions. If this assumption does not hold then it can result in visible artifacts. A problematic case is when a thin object is near a surface, roughly parallel to its normal, as shown by Fig. 1. In this case, sampling based methods usually produce a false lightening of the area directly under the object compared to the surrounding surface. This happens because very few, possibly none of the sampled rays will hit the object. In the case of screen space techniques this artifact is not present because rays that would sample the area behind the curtain will have depth values higher than the ones stored in the depth buffer (i.e., the curtain's depth). Basically, the curtain will act as a wall compared to ray tracing where those samples will not hit the geometry.

Although this false lightening is correct in the already established definition of ambient occlusion, it contradicts our intuition about the real world. In this paper we formulate the problem, demonstrate its presence in the Unreal Engine 4 and Unity game engines and propose a physically based solution as an extension to the basic algorithm.

Cite this article as: "Accounting for Non-Uniform Ambient Lighting in Ray Traced Ambient Occlusion", Periodica Polytechnica Electrical Engineering and Computer Science, 66(3), pp. 248–252, 2022. https://doi.org/10.3311/PPee.20209

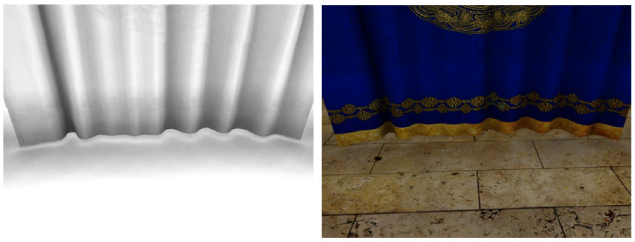


Fig. 1 The area directly under the curtain incorrectly receives more ambient light than its surroundings.

2 Related work

Nearby occlusions, accessibility, or openness of points on a surface were first examined with the objective of ambient lighting calculations in [1, 2], where the *obscurances method* was introduced. It was also called ambient occlusion in [3, 4]. Obtaining their radiance, occluders can be treated as indirect light sources, so even color bleeding effects can be simulated [5–8].

Until recently, a physically correct approach would have been too expensive computationally when dynamic scenes need to be rendered in real-time, therefore the mentioned models simplify the problem by using sampled representation of the visible geometry [9].

For a better approximation of the ambient lighting, Bavoil et al. [10] proposed a hybrid technique to combine the classic screen space approach with the new hardware accelerated ray tracing to improve the robustness while maintaining a low sample count. Gautron [11] proposed a fully ray tracing based technique where the sampling noise is filtered in world space with various methods to reach real-time frame rates with complex scenes. Another promising direction could be to use neural networks to predict the ambient occlusion. Zhang et al. [12] recently proposed one such technique with a detailed comparison to existing methods.

3 Mathematical formulation

To simplify the formulation of ambient occlusion or obscurances models, we can assume that the surfaces are diffuse and have albedo $a(\vec{s})$. Consequently, their BRDF is $f_r(\vec{s}) = a(\vec{s})/\pi$.

According to the rendering equation [13], the reflected radiance L^r in *shaded point* $\vec{s}$ can be obtained as:

$$L^r(\vec{s}) = \int_\Omega L^{in}(\vec{s},\vec{\omega})\frac{a(\vec{s})}{\pi}\cos\theta d\omega, \tag{1}$$

where Ω is the hemisphere, $L^{in}(\vec{s},\vec{\omega})$ is the incident radiance from direction $\vec{\omega}$ and θ is the angle between illumination direction $\vec{\omega}$ and the surface normal.

If no surface is seen from $\vec{s}$ at direction $\vec{\omega}$, then the shaded point $\vec{s}$ is said to be *open* in this direction, and incident radiance L^{in} is equal to ambient radiance L^a. If there is a nearby occluder, then this point is called *closed*. The incident radiance at this direction is the radiance of the *occluder point* $\vec{o}$.

This means that the radiance of arbitrary far occluder points may influence the shaded point, but this does not meet our intuition. Our everyday experience is that the effect of very far surfaces becomes negligible, due to that the real space is not empty but is filled by participating media. Its effect can be expressed as:

$$\mu(d) = \left(1-\exp\left(-\sigma_t d\right)\right), \tag{2}$$

where σ_t is the extinction coefficient and d is the distance from the shaded point. As σ_t increases only a narrower neighborhood needs to be considered. Then the incoming radiance can be calculated as:

$$L^{in}(\vec{s},\vec{\omega}) = \left(1-\mu(d)\right)L^r(\vec{o}) + \mu(d)L^a. \tag{3}$$

In the classical ambient occlusion $\mu(d)$ is replaced by a step function with a value of 1 if d is greater than R and 0 otherwise, where R is a user defined parameter to control the range of ambient occlusion.

Because $a(\vec{s})$ is independent from the view direction, using $\mu(d)$, the reflected radiance of the shaded point can be expressed in the following way:

$$L^r(\vec{s}) = a(\vec{s})\left(L^a O(\vec{s}) + I(\vec{s})\right), \tag{4}$$

where $I(\vec{s})$ is the incident radiance from nearby occluders and $O(\vec{s})$ is the ambient occlusion of the shaded point [1, 2]:

$$O(\vec{s}) = \frac{1}{\pi}\int_\Omega \mu(d)\cos\theta d\omega. \tag{5}$$

At this point, we have reached the final equation to calculate ambient occlusion. Now for the sake of simplicity, let us examine the flat-land case, where we can calculate the exact function. We will observe a fixed point on the ground and calculate the visible part of the semicircle with radius R, which is not covered by the box above the surface. This correlates with the ambient occlusion as discussed previously. Schematic illustration of the scene can be seen in Fig. 2 and the results, as a function of the horizontal distance (t), are depicted in Fig. 3 with multiple parameter configurations.

In this scene, the function calculates Eq. 5 apart from the $1/\pi$ factor. If we use a uniform L^a then it will be

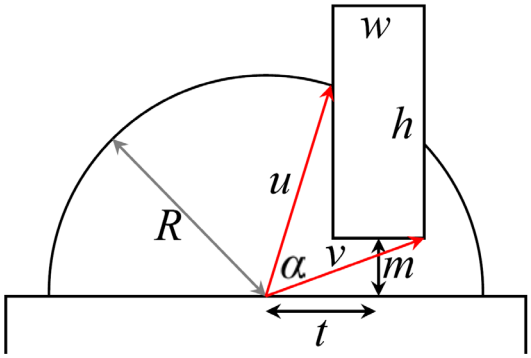


Fig. 2 The observed point is at the center of the semicircle with radius R. We are interested in the visible angle which is $\pi-\alpha$.

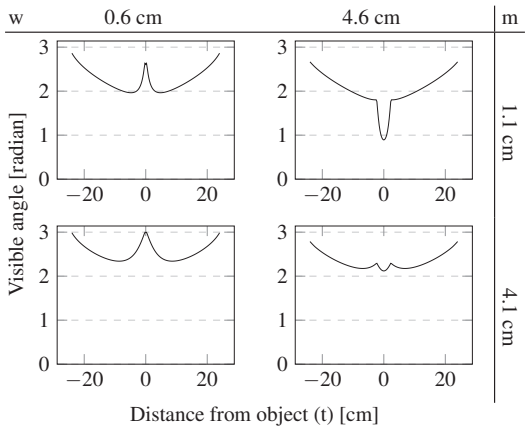


Fig. 3 Results showing the artifact in 2D. Parameters: $R = 25$ cm, $h = 100$ cm.

responsible for the characteristics of the ambient occlusion effect. We can see that the effect will change based on how large or small the width and the vertical height is compared to each other and to the radius of the semicircle. If the width is small relative to R and m (left), then the function has a local apex when the shaded point is directly under the object. It has a higher maximum value if the vertical distance is larger. The function can be extended to 3D if we assume that the other extents of the thin occluder are large. This is the case in Fig. 1, and it results in the visible highlight under the curtain.

If the width is the larger, then with smaller m (top right) we have a correct darkening effect under the object. If m and the w are close to each other (bottom right) then the artifact is unnoticeable and there is no prominent darkening effect which is in line with what we expect from the real world.

So far we have assumed that L^a is uniform in the far field and the light will not make any other interactions besides extinction on solid objects (occluders). In reality though, scattering also happens in the near field and the assumption of receiving L^a radiance from a given direction is only true, when every point along the ray receives the same ambient radiance L^a. If a given direction is close to a solid object, then this assumption will not hold because that object will block some part of the ambient radiance. The idea is depicted in Fig. 4. These local differences cause an error in the ambient occlusion function which becomes evident in some scenarios (e.g., Fig. 1).

To calculate a better approximation of L^a, we propose to use secondary rays to further examine the open primary directions for nearby occluders. This way we can scale down contribution of the primary rays in a physically plausible way.

4 Evaluating popular RTAO implementations

The artifact can be easily observed with other popular RTAO implementations, for example those found in game engines like Unity or Unreal Engine 4. In Fig. 5 we can see that while RTAO provides softer images compared to the

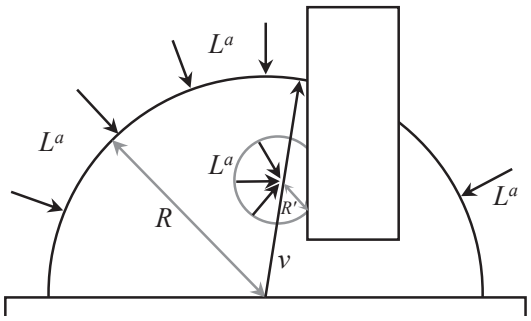


Fig. 4 Even though incoming L^a is uniform in the far field, points along ray will receive less ambient radiance than rays that are farther from the object.

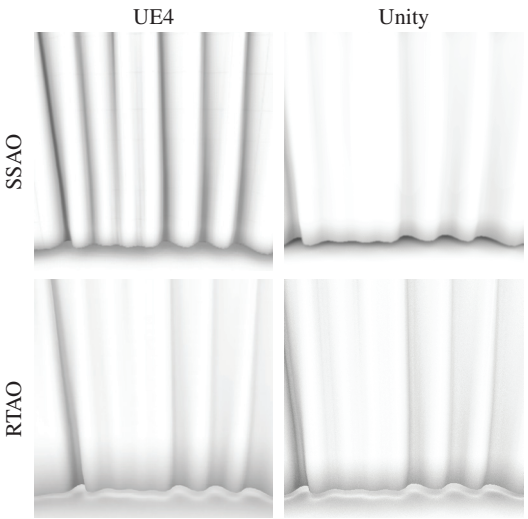


Fig. 5 Visualization of the artifact as it appears in Unreal Engine and in Unity.

traditional solution, the artifact appears exactly under the curtain, in both engines.

For reference, we generated an image of the Sponza scene using the Cycles renderer from Blender, which can be seen in Fig. 6. A path tracer does not need to use an approximation for ambient occlusion, so the artifact is not present which is in line with what we expect about the real world.

5 Details of the proposed method

Our goal is to measure L^a for every sampled open direction. To this end we introduce secondary rays for every primary ray. These will be checked for occlusion, and we will scale down L^a according to the number of closed secondary directions. The next question is how to select these additional rays. Our goal was to use the least number of them that are already able to capture enough detail about L^a. We ended up using the following construction.

We used two secondary rays per primary ray. The first was perpendicular to the primary ray and to the surface normal, while the second was perpendicular to the primary ray and to the first secondary ray (as depicted in Fig. 7). The rays were cast around the middle point of the primary ray with radius R'. This parameter describes the volume of each ray and should be set based on the original R and the number of primary rays to reduce overlapping.

Fig. 6 Path traced render of the Sponza scene.

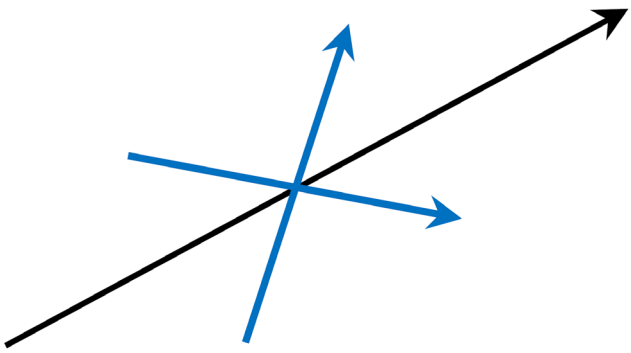


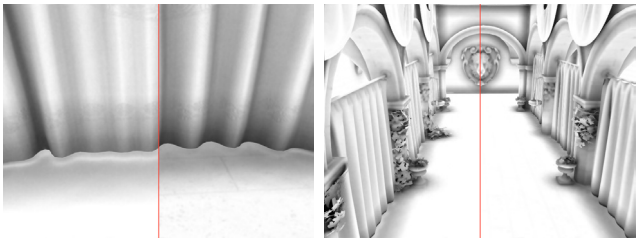
Fig. 7 Schematic illustration of the secondary rays (blue) relative to the primary ray (black).

We did not use temporal filtering in our implementation, but it would work well with our technique. The scaling factor of L^a can be thought of as scaling the weight of the current ray and thus would naturally incorporate into the history. Even the origin of the secondary rays could be varied randomly in subsequent frames.

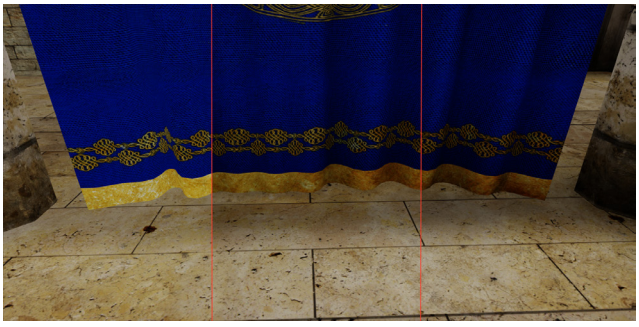
6 Results

We implemented the described algorithm in Vulkan using a hybrid rendering technique. We calculated the direct illumination according to the GLTF specification on the Sponza scene and combined it with the aforementioned ambient occlusion factor. Fig. 8 demonstrates the results and compares our algorithms with the original. Our extension compensates for the lightening effect and can be observed under the curtains and on the pillars. On other parts of the scene where ambient occlusion occurs a slight darkening can be observed.

Performances naturally takes a hit for tracing more rays. On a 1080p image an additional 2 ms is needed with 4 primary samples in our implementation. With 32 samples this increases to 15ms which makes it unusable in real time scenarios. We found that 4 samples are enough to produce images where the noise is unnoticeable in the



(a)



(b)

Fig. 8 Comparison of the original and the proposed algorithms; (a) visualization of ambient occlusion. Left parts of the images were rendered by the original method, while the right parts were calculated with our extension. (b) left part has no AO, the middle was rendered by the original method and the right by our extension. 32 rays per pixel were used in both cases.

composed picture and temporal filtering would further reduce the performance cost.

7 Conclusion

In this paper we discussed a problem of ambient occlusion that becomes evident around thin objects with large surfaces. We analyzed the issue in 2D and showed that it is inherent to the commonly accepted definition of ambient occlusion. We also demonstrated that it is present in the RTAO implementation of popular game engines. We presented an extension to the standard ray tracing based method that is able to detect the geometry causing the artifact and reduces the unnatural highlight.

Acknowledgement

The research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program, and András Fridvalszky was supported by the ÚNKP-21-3 New National Excellence Program of the Ministry for Innovation and Technology from the source of the National Research, Development and Innovation Fund.

References

[1] Zhukov, S., Iones, A., Kronin, G. "An ambient light illumination model", In: Rendering Techniques '98: Proceedings of the Eurographics Workshop, Vienna, Austria, 1998, pp. 45–56. ISBN: 978-3-7091-6453-2 https://doi.org/10.1007/978-3-7091-6453-2_5

[2] Iones, A., Krupkin, A., Sbert, M., Zhukov, S. "Fast, realistic lighting for video games", IEEE Computer Graphics and Applications, 23(3), pp. 54–64, 2003. https://doi.org/10.1109/MCG.2003.1198263

[3] Landis, H. "Production-Ready Global Illumination", In: SIGGRAPH 2002 course note #16, 2002, pp. 87–102. [online] Available at: https://asset-pdf.scinapse.io/prod/65018654/65018654.pdf [Accessed: 21 March 2022]

[4] Pharr, M., Green, S. "Chapter 17. Ambient Occlusion", In: Fernando, R. (ed.) GPU Gems: Programming Techniques, Tips and Tricks for Real-Time Graphics, Addison-Wesley Professional, 2004, pp. 279–292. ISBN-13978-0321228321 [online] Available at: https://developer.nvidia.com/gpugems/gpugems/part-iii-materials/chapter-17-ambient-occlusion [Accessed: 21 March 2022]

[5] Méndez, A., Sbert, M., Catá, J. "Real-time obscurances with color bleeding", In: SCCG '03: Proceedings of the 19th Spring Conference on Computer Graphics, Budmerice, Slovakia, 2003, pp. 171–176. ISBN 978-1-58113-861-0 https://doi.org/10.1145/984952.984980

[6] Umenhoffer, T., Tóth, B., Szirmay-Kalos, L. "Efficient methods for ambient lighting", In: SCCG '09: Proceedings of the 25th Spring Conference on Computer Graphics, Budmerice, Slovakia, 2009, pp. 87–94. ISBN 978-1-4503-0769-7 https://doi.org/10.1145/1980462.1980482

[7] Szirmay-Kalos, L., Umenhoffer, T., Tóth, B., Szécsi, L., Sbert, M. "Volumetric Ambient Occlusion for Real-Time Rendering and Games", IEEE Computer Graphics and Applications, 30(1), pp. 70–79, 2010. https://doi.org/10.1109/MCG.2010.19

[8] 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, Boston, MA, USA, 2009, pp. 75-82. ISBN 978-1-60558-429-4 https://doi.org/10.1145/1507149.1507161

[9] Bavoil, L., Sainz, M., Dimitrov, R. "Image-space horizon-based ambient occlusion", In: SIGGRAPH '08: ACM SIGGRAPH 2008 talks, Los Angeles, CA, USA, 22, 2008. ISBN: 978-1-60558-343-3 https://doi.org/10.1145/1401032.1401061

[10] Bavoil, L., Liu, E., Shirley, P., McGuire, M. "Hybrid Ray-Traced Ambient Occlusion", Poster at High-Performance Graphics, pp. 1–4, 2018. [online] Available at: https://casual-effects.com/research/Bavoil2018AO [Accessed: 21 March 2022]

[11] Gautron, P. "Real-Time Ray-Traced Ambient Occlusion of Complex Scenes using Spatial Hashing", presented at SIGGRAPH '20: ACM SIGGRAPH 2020 Talks, [online conference] 2020. Aug. 17., 5. ISBN: 978-1-4503-7971-7 https://doi.org/10.1145/3388767.3407375

[12] Zhang, D., Xian, C., Luo, G., Xiong, Y., Han, C. "DeepAO: Efficient Screen Space Ambient Occlusion Generation via Deep Network", IEEE Access, 8, pp. 64434–64441. https://doi.org/10.1109/ACCESS.2020.2984771

[13] Kajiya, J. T. "The rendering equation", ACM SIGGRAPH Computer Graphics, 20(4), pp. 143–150, 1986. https://doi.org/10.1145/15886.15902