Combining Cone and Ray Tracing for Ambient Occlusion Computation

Namo Podee[1,a)] Yoshinori Dobashi[1,b)] Tsuyoshi Yamamoto[1,c)]

Abstract: We present a method to compute ambient occlusion efficiently by combining two methods: Voxel cone tracing and Ray tracing together. Global illumination or indirect illumination is a key factor to the realism of the computer-generated image. Although it can be accurately solved by Ray tracing method, it is too complicated and difficult to be calculated in an interactive time with our current-generation-hardware. There are many methods that tried to solve it accurately within such a limited time and one of the most promising method is Voxel cone tracing. Unfortunately, the method has a low and rough accuracy. Thus, our idea is to combine the fast and rough method with the slow and accurate method together to efficiently generate global illumination effects. Ambient occlusion is the simplest global illumination effect, which we decided to solve it before anything else. Our method estimates an error from voxel cone tracing and uses this error to blend in the result of the two methods together. By doing this, a rough result from Voxel cone tracing can handle large area integration while an accurate result from Ray tracing can fill in as a detail. Our result shows that our method can achieve more accurate result than the two methods individually.

1. Introduction

Since the beginning of computer graphics, photorealistic rendering is one of the most important subjects of the research field. It can be used in many applications such as medical visualization, architecture visualization, product design, movie's visual effect rendering, and game's interactive rendering. One of the keys of photorealistic rendering is global illumination.

Ambient occlusion (AO) [4] is a simple and important effect . It reduces light intensity of every pixel by an amount relative to how much the other surfaces are visible to each pixel. In another word, a group of surfaces that is close and facing to each other will have less light on them. This aims to reproduce the effect of geometries block or occlude other geometries from ambient light.

AO effect enhances the realism of the generated result. However, just like any other global illumination effect, it is usually computed offline by using ray tracing [8] If the quality and accuracy are not important there are many methods that could compute AO in real-time and voxel cone tracing [2] is one such a method. Voxel cone tracing can generate almost any global illumination effects of a dynamic scene in real-time in exchange for the quality and accuracy. The result from voxel cone tracing is very rough, which is the problem if the detail is important.

To improve a quality of AO voxel cone tracing, we propose a method that uses voxel cone tracing to produce AO and estimate its error then reduce it by using ray tracing. Our result shows that our method can produce more accurate results than those obtained by voxel cone tracing while having less noise than ray tracing.

[1] Hokkaido University
[a)] namo@ime.ist.hokudai.ac.jp
[b)] doba@ime.ist.hokudai.ac.jp
[c)] yamamoto@ist.hokudai.ac.jp

Our main contributions are:

- We propose an AO voxel cone tracing error estimation method.
- We propose ambient occlusion computation method that combines voxel cone tracing and ray tracing.
- We demonstrate that our method has higher performance than voxel cone tracing and less noise than ray tracing.

2. Related Work

Ambient occlusion (AO) [4] is a shading technique of a surface is partially or fully occluded from an ambient light by surrounding geometries as shown in Figure 1. Thus, AO tends to make a surface darker as it gets closer to other surfaces. AO is often used to approximately simulate the global illumination effects and widely used in both offline and online rendering. It can significantly improve the realism of a result with just a simple calculation.

Ray tracing [8] can be used to compute AO by shooting a number of short distance rays around the target point to detect any nearby surfaces and calculate the visibility from them as shown in Figure 2b. Although the result of this method is very high quality, it needs many rays and it takes long computational time to get a noiseless result.

Screen space ambient occlusion (SSAO) [1], [3], [5], [6] is a very popular technique for AO computation in game and interactive application. It can compute AO in real-time by using depth information on the resulting image to determine how close each pixel is to each other, then add occlusion to them. The downside is that it can only calculate AO from the information visible through the screen so the result will far less accurate than the result from ray tracing.

(a) Conference Room Scene

(b) AO Result

Fig. 1: AO effect makes floor and wall, which is occluded by the chairs, appeared to be darker.

Voxel cone tracing [2] is an advanced AO computation method for games and interactive applications. Voxel cone tracing simplifies a whole scene into a voxel structure then it can perform a visibility tracing efficiently by using the whole scene information, unlike SSAO. Voxel cone tracing can trace a cone of visibility by sampling opacity data of voxels inside the cone from voxel structure as shown in Figure 2c. The opacity of a voxel is an averaged opacity of geometries inside the voxel. This way voxel cone tracing can accumulate visibility cones, which is much more efficient than lines like ray tracing does. Although voxel cone tracing can compute AO in real-time, it simplifies geometries into the voxel structure so the detail of these geometries will be lost. The result from voxel cone tracing is thus inaccurate.

3. Cone and Ray Tracing For AO

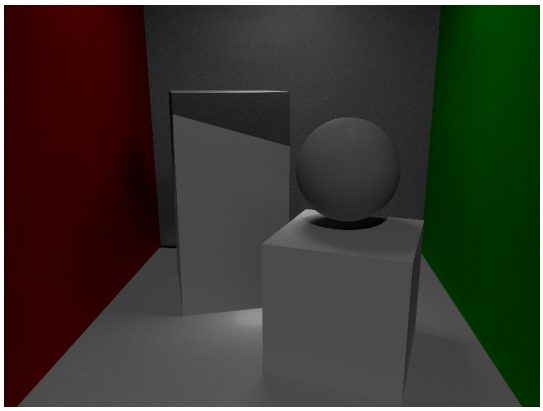
We present our method that combines voxel cone tracing and ray tracing together to be more accurate and efficient. To fix voxel cone tracing inaccuracy problem, our method uses ray tracing to find a more accurate result for any inaccurate cone tracing. So, we need to estimate how accurate voxel cone tracing for each cone is.

Voxel cone tracing computes ambient occlusion by using only opacity value of voxel thus an accuracy is directly related to this value.

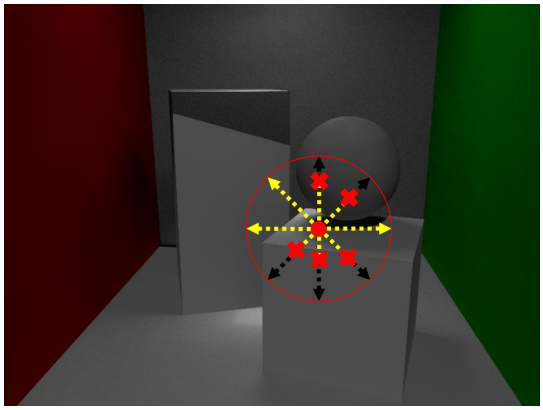
3.1 Voxel cone tracing error

The error of cone tracing mainly comes from voxelization, which means a difference between calculation with a real geometry and geometry approximated by voxels. This difference happens because each voxel parameter is an averaged parameter from multiple geometries inside the voxel. A single parameter value cannot represent multiple objects correctly. If we know an error then we can control an accuracy of our result. According to a paper by Walter [7], an error can hardly be visually perceived if it is under 2%. In other words, if we replace some of the results from cone tracing with ray tracing to keep its error under 2%, then we should be fine. Then how do we calculate its error?

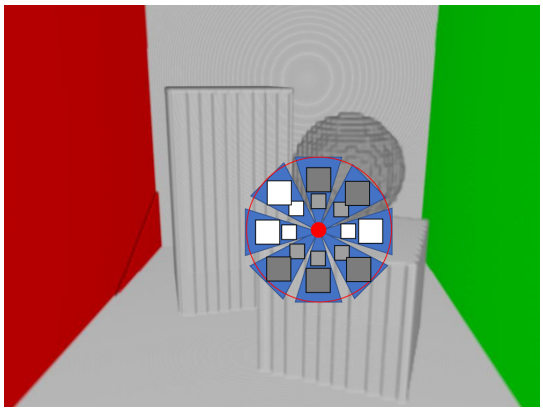
Error from voxelization can be calculated easily by finding the error of opacity parameter, which is the only parameter that is used to compute AO. Voxel structure is an octree, which has many levels in a parent-child relationship. The lowest level voxels are calculated directly from a geometry, while upper-level voxels are averaged of their children, i.e., lower level voxels. An error of



(a) Cornell Box Scene



(b) AO with Ray Tracing



(c) AO with Voxel Cone Tracing

Fig. 2: Two methods of AO computation on Cornell box scene. Ray tracing method is done by generating many rays around the target point while voxel cone tracing generating cones and accumulate opacity sample of voxels that is inside each cone.

opacity of each voxel happens when representing its opacity as its lowest children opacity. So, we can find it by computing the difference of its opacity (mean) and its lowest children opacity (sample). This is just a standard deviation (SD). So, we use SD of opacity as an error of voxelization directly.

Thus, for ambient occlusion computation. We estimate an error of voxel cone tracing of a single cone by accumulating only SD of an opacity of each collected voxel.

3.2 Combination of the two methods

Now that we know an estimated error of voxel cone tracing then, we can use it for blending two methods together.

Our method starts with the same method as the original voxel cone tracing by tracing cones in fixed directions. For each cone that has an error value more than a threshold, we want to use ray tracing to reduce its error. But to blend the results computed by the two different methods, we need to find a blending value, which tells how much a result from each method should be used. This blending value must smoothly change from 0 to 1 relative to the error value, to combine the results of the two methods without any noticeable artifact. Hence, we use an exponential function for this blending value.

$$\alpha = \exp(-\beta(\epsilon - \gamma)) \tag{1}$$

, where α is a ratio of voxel cone tracing, β is a transition parameter, which decides how smooth the transition will be, ϵ is an error value, and γ is an error threshold, which decides minimum error value to start the transition. α must also be within 0 to 1. After we calculated the blending value, we also use it to find sample number for ray tracing inside each cone. Then, for each ray, we randomly choose a direction inside the cone and trace a ray. This method prone to noise but to correctly approximate visibility cone, we must use Monte Carlo method for this. After traced, we use the results to approximate the integral of visibility function of that cone with Monte Carlo method. Our method is summarized as followed:

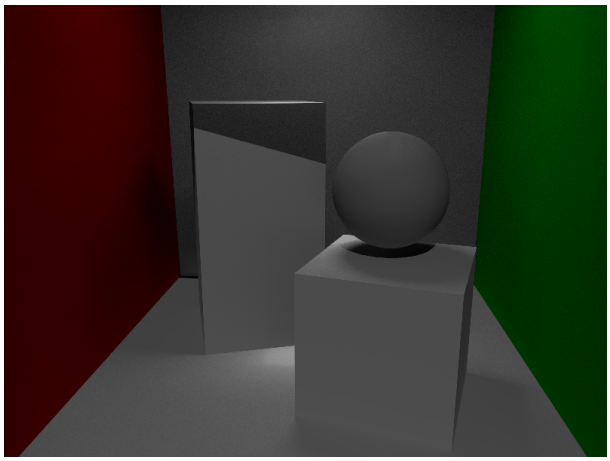
(1) Do voxel cone tracing for fixed directions around a hemisphere.
(2) For each cone whose error from voxel cone tracing is over a threshold.
  (a) Calculate the blending value by using Equation 1.
  (b) Do ray tracing inside the cone to integral the cones visibility. Its number of sample is according to the followed formula:

$$N = (1 - \alpha)N_{max} \tag{2}$$

  , where N is a number of rays, N_{max} is a maximum number of rays.
  (c) Blend the result of two methods, by using the blending value:

$$I = \alpha C + (1 - \alpha)P \tag{3}$$

  , where I is the final result, C is the voxel cone tracing result, and P is the ray tracing result.
(3) Accumulate all visibility result from all cones and accumulate occluded space from them then, use it to shade the pixel.



(a) Cornell Box

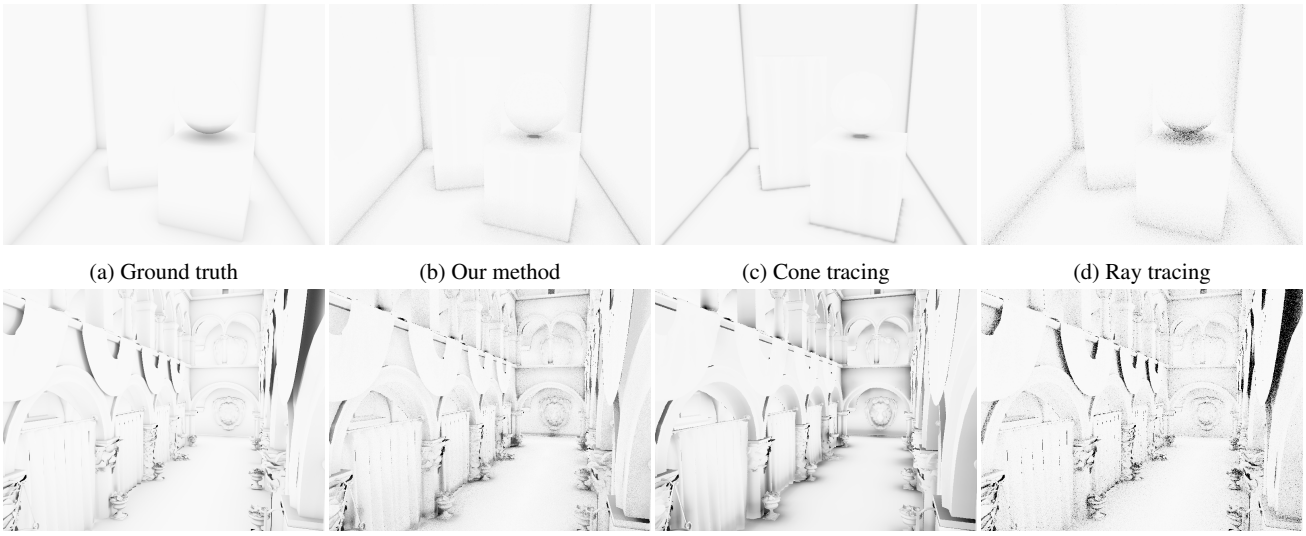
(b) Crytek Sponza

Fig. 3: Two experiment scenes. Cornell box is simple while Crytek Sponza scene is complex and has a lot of detail.

4. Result and Discussion

We implement the algorithm with CUDA and OpenGL. The program is executed on the machine with Intel Core i7-4790K CPU @ 4.00 GHz, Memory 32 GB, and one GeForce GTX TITAN X. We experiment and compare three methods, which are our method, ray tracing, and cone tracing, with our ground truth, which is rendered by ray tracing method, on two scenes, which are Cornell box (Figure 3a), and Crytek Sponza (Figure 3b).

The results are shown in Figure 4. Our method can produce a higher quality result than cone tracing while having less noise than ray tracing in both scenes. However, our method still suffers from dark artifacts in a corner and under the ball in Cornell box scene. This happened because our error estimator mistakes them as accurate cone tracings. There is much error estimated mistakes since we ignore to calculate an error from tracing itself. Tracing in voxel cone tracing is similar to tracing a participating media. This need to be considered in our error estimator as well. Also, from the stats breakdown at Table 1, our method produces the less accurate result than ray tracing. Additionally, Figure 5 shows that ray tracing will win our method in root mean square error comparison if we progress the rendering more than a single frame. This outcome is expected since we will accumulate an error from



(a) Ground truth (b) Our method (c) Cone tracing (d) Ray tracing

(e) Ground truth (f) Our method (g) Cone tracing (h) Ray tracing

Fig. 4: The single frame results of ambient occlusion computation. Our method can produce much less noise result than ray tracing's and more accurate then cone tracing's.

Table 1: First Frame Stats Breakdown

Scene	Method	Root Square Mean Error	Time(ms)
Cornell Box	Our Method	0.22815	60
	Cone Tracing	0.34017	30
	Ray Tracing	0.28343	66
Crytek Sponza	Our Method	0.52953	285
	Cone Tracing	0.79895	63
	Ray Tracing	0.47633	316

cone tracing from several frames.

Nonetheless, our method can produce much less noise than ray tracing does, while being able to express a nuance of detail, which is impossible in cone tracing, as you can see at occlusion under the clothes hanged on the second floor in Figure 4f. Figure 6 shows only the result of cone tracing in our method thus the darker area means ray tracing will be used instead. As you can see, ray tracing will be used mostly in the complicated area while leaving non-occluded surfaces for cone tracing, which is efficient.

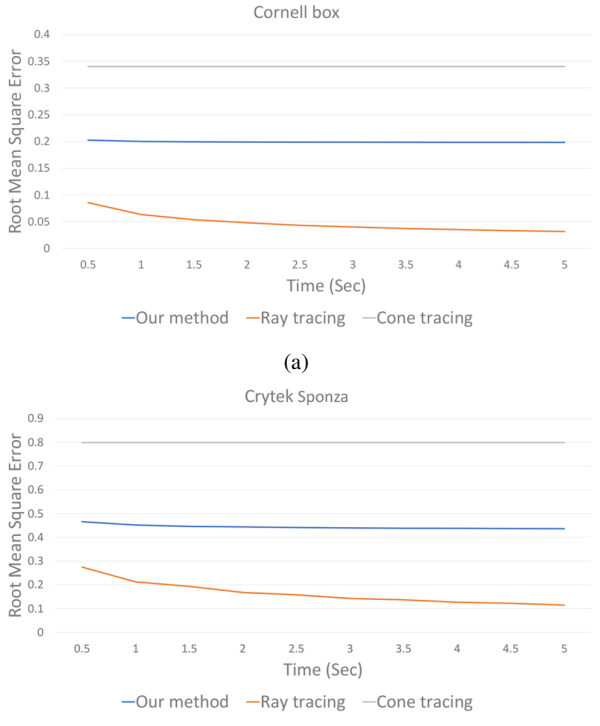
5. Conclusion and Future Work

Our method can produce a result that has more accurate than cone tracing and has less noise than ray tracing. However, our error estimation technique isn't perfected yet. There is a lot of areas our method fails to use ray tracing on, which make our method less accurate than it should be.

We plan to improve our voxel cone tracing error estimation by including an error from tracing and continue our study of cone and ray tracing by using it to compute indirect illumination, which is much harder.

6. Acknowledgments

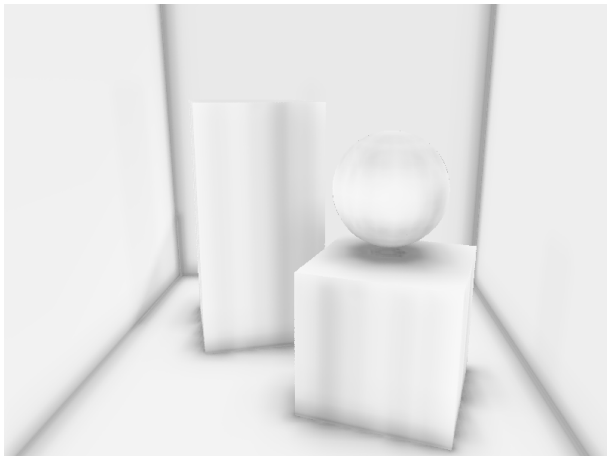
Cornell Box model courtesy of Cornell University, Crytek Sponza model courtesy of Frank Meinl at Crytek and Marko Dabrovic, Conference room courtesy of Anat Grynberg and Greg Ward.



(a)

(b)

Fig. 5: A comparison of root mean square error between our method, ray tracing, and cone tracing in two scenes. Our method unable to win against ray tracing by using root mean square error but our method does much better than cone tracing.



(a) Cornell box

(b) Crytek Sponza

Fig. 6: The result from our method without ray tracing. This means the darker parts show an area that our method decides to choose ray tracing over cone tracing.

References

[1] Bavoil, L., Sainz, M. and Dimitrov, R.: Image-space Horizon-based Ambient Occlusion, *ACM SIGGRAPH 2008 Talks*, SIGGRAPH '08, New York, NY, USA, ACM, pp. 22:1–22:1 (online), DOI: 10.1145/1401032.1401061 (2008).

[2] Crassin, C., Neyret, F., Sainz, M., Green, S. and Eisemann, E.: Interactive Indirect Illumination Using Voxel-based Cone Tracing: An Insight, *ACM SIGGRAPH 2011 Talks*, SIGGRAPH '11, New York, NY, USA, ACM, pp. 20:1–20:1 (online), DOI: 10.1145/2037826.2037853 (2011).

[3] Filion, D. and McNaughton, R.: Effects & Techniques, *ACM SIGGRAPH 2008 Games*, SIGGRAPH '08, New York, NY, USA, ACM, pp. 133–164 (online), DOI: 10.1145/1404435.1404441 (2008).

[4] Miller, G.: Efficient Algorithms for Local and Global Accessibility Shading, *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, New York, NY, USA, ACM, pp. 319–326 (online), DOI: 10.1145/192161.192244 (1994).

[5] Mittring, M.: Finding Next Gen: CryEngine 2, *ACM SIGGRAPH 2007 Courses*, SIGGRAPH '07, New York, NY, USA, ACM, pp. 97–121 (online), DOI: 10.1145/1281500.1281671 (2007).

[6] Shanmugam, P. and Arikan, O.: Hardware Accelerated Ambient Occlusion Techniques on GPUs, *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, I3D '07, New York, NY, USA, ACM, pp. 73–80 (online), DOI: 10.1145/1230100.1230113 (2007).

[7] Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M. and Greenberg, D. P.: Lightcuts: A Scalable Approach to Illumination, *ACM Trans. Graph.*, Vol. 24, No. 3, pp. 1098–1107 (online), DOI: 10.1145/1073204.1073318 (2005).

[8] Whitted, T.: An Improved Illumination Model for Shaded Display, *Commun. ACM*, Vol. 23, No. 6, pp. 343–349 (online), DOI: 10.1145/358876.358882 (1980).