An Unbiased Hybrid Rendering Approach to Path Guiding

Jia-Wun Jhang and Chun-Fa Chang

National Taiwan Normal University, Taiwan

Abstract

When we think of hybrid rendering of rasterization and ray tracing, we often consider rasterization as a mean to solve the primary rays and then consider ray tracing as a mean to add secondary effects. We take a different approach to combine ray tracing and rasterization, in which our final output images are still produced with a path tracer. We leverage the GPU rasterization to build the necessary data that are required for path guidng, thus improves the convergence of our path tracer. We borrow the ideas of Voxel Cone Tracing and implement it in GPU shaders to build the path guiding data. The advantage of our proposed hybrid approach is that it maintains the unbiased results of a Monte Carlo path tracer while incurring relatively small performance hit in path guiding, as shown in our preliminary results.

1. Introduction

We present a hybrid rendering approach that combines ray tracing with rasterization. Typically a hybrid renderer produces the initial frame with rasterization and GPU shaders, then adds the missing global illumination effects (such as the shadow, reflection, and refraction) to some objects by ray tracing selected regions of the initial frame. However the other regions that are not enhanced by ray tracing may still miss the subtle global illumination effects such as indirect lighting, especially when the results are compared to the ground truth images of an unbiased Monte Carlo path tracer.

We take a different approach to combine ray tracing and rasterization, in which our final output images are still produced with a path tracer. We leverage the GPU rasterization to build the necessary data that are required for path guidng [MGN17] [DHD20], thus improves the convergence of our path tracer. We borrow the ideas of Voxel Cone Tracing [CNS*11] and implement it in GPU shaders. However we do not use Voxel Cone Tracing for rendering the output frame. Instead we use it to build the path guiding data. The advantage of our proposed hybrid approach is that it maintains the unbiased results of a Monte Carlo path tracer while incurring relatively small performance hit in path guiding.

2. Unbiased Hybrid Rendering

Unlike the typical hybrid renders that produce the initial frame with rasterization, our proposed unbiased hybrid renderer consists of a Monte Carlo path tracer which samples the reflection directions based on path guiding. The path guiding improves the convergence of the path tracer by providing the information about the surrounding lighting conditions at the path vertices. There are many choices to store the surrounding lighting information for path guiding, such as the SD-trees in [MGN17]. We use a rather simple irradiance record for path tracing, which divides the scenes into uniform-sized voxels and stores the lighting information from all incoming directions in each voxel. The simplicity of our irradiance record allows us to build its content with GPU rasterization methods such as the voxel cone tracing.

2.1. Irradiance Record for Path Guiding

We first divide the scene space into uniform sized voxels. Each voxel contains an irradiance record that stores the lighting information from all incoming directions as depicted in Figure 1. Before the path tracer starts, we fill in the irradiance records in all voxels using Voxel Cone Tracing which will be explained in the next subsection.

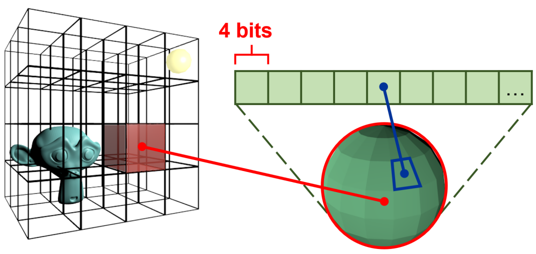


Figure 1: *Each voxel contains an irradiance record that stores the lighting information from all incoming directions.*

2.2. Voxel Cone Tracing

We leverage the GPU rasterization in our hybrid renderer to speed up the processing of irradiance records that are used for path guid-

DOI: 10.2312/egp.20211032

ing. Our approach is inspired by Voxel Cone Tracing [CNS*11]. However we do not use Voxel Cone Tracing for rendering the output frame. Instead we use it to build the path guiding data.

First we divide the scene space into uniform sized voxels. Each voxel contains a tag that is first initialized to either X or O: X for empty voxels, O for voxels that contain objects. This initialization could be implemented using any GPU assisted voxelization method. We also set the tag to L for voxels that contain a light source, then use the reflective shadow map [DS05] to find the voxels that could be directly illuminated and set their tags to P.

Finally we process the indirect lighting by finding the voxels with the O tag and perform voxel cone tracing to build the irradiance records that are used for path guiding. Figure 2 summarizes the above mentioned steps. Note that we also attempt to process the caustics as a special case by setting the tags of the illuminated voxels to P. This approach is still an experiment and could change in our future implementation.

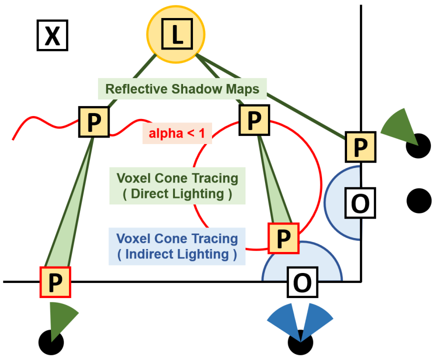


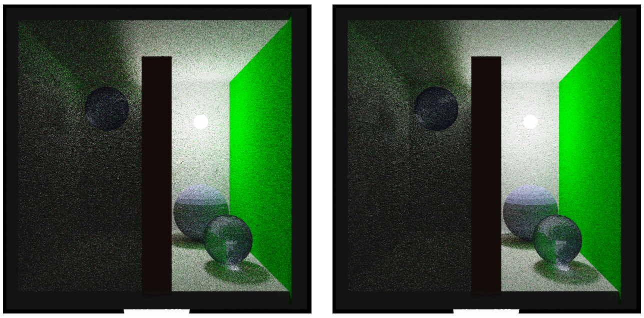
Figure 2: *Voxel Cone Tracing*

3. Results

We show the results of our hybrid renderer using two scenes as shown in Figures 3 and 4. They are rendered on a notebook PC with an NVIDIA GeForce GTX 1070 GPU. The first is a Cornell box with a center wall to emphasize the indirect lighting in the left side. It contains 5,306 triangles and $13 \times 18 \times 13$ voxels. The second is a under-water scene to emphasize the caustics. It contains 102,974 triangles and $11 \times 11 \times 11$ voxels. All figures are rendered at the size of 1024×1024 and show the convergence after accumulating 30 frames. Please see the figure captions for the frame rates and the mean squared errors when compared to the converged results. The results show our hybrid renderer produces smaller errors while reducing the frame rates by about 2% to 3%.

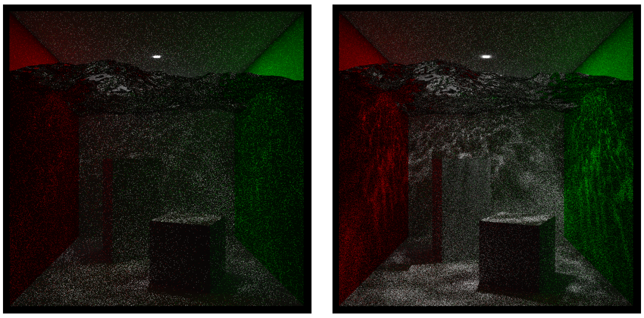
4. Conclusion and Future Work

We have shown an alternative approach to hybrid rendering, in which the GPU rasterization is leveraged to help the path guiding of a Monte Carlo path tracer. In order to save the memory space, we currently use a 4-bit counter to store the incoming lighting intensity in each direction of the irradiance record. All incoming directions in an irradiance record are divided into 16 horizontal directions and



(a) *Results without path guiding: FPS=26.3, MSE=0.044* **(b)** *Results with path guiding: FPS=25.8, MSE=0.039*

Figure 3: *Results of the Cornell Box scene*



(a) *Results without path guiding: FPS=15.4, MSE=0.045* **(b)** *Results with path guiding: FPS=15.0, MSE=0.036*

Figure 4: *Results of the under water scene*

8 vertical directions. This memory space constraint inevitably limits the effectiveness of the path guiding, which is certainly an area to improve in the future.

This work is supported in part by MediaTek Inc.

References

[CNS*11] Crassin C., Neyret F., Sainz M., Green S., Eisemann E.: Interactive Indirect Illumination Using Voxel Cone Tracing. *Computer Graphics Forum* (2011). doi:10.1111/j.1467-8659.2011.02063.x. 1, 2

[DHD20] Dittebrandt A., Hanika J., Dachsbacher C.: Temporal Sample Reuse for Next Event Estimation and Path Guiding for Real-Time Path Tracing. In *Eurographics Symposium on Rendering - DL-only Track* (2020), Dachsbacher C., Pharr M., (Eds.), The Eurographics Association. doi:10.2312/sr.20201135. 1

[DS05] Dachsbacher C., Stamminger M.: Reflective shadow maps. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (New York, NY, USA, 2005), I3D '05, Association for Computing Machinery, p. 203–231. URL: https://doi.org/10.1145/1053427.1053460, doi:10.1145/1053427.1053460. 2

[MGN17] Müller T., Gross M., Novák J.: Practical path guiding for efficient light-transport simulation. *Comput. Graph. Forum 36*, 4 (July 2017), 91–100. URL: https://doi.org/10.1111/cgf.13227, doi:10.1111/cgf.13227. 1