Hybrid Progressive Photon Mapping

GROUP 25

HANSUNG KIM, University of California, Berkeley, United States

LINYUE SONG, University of California, Berkeley, United States

RAHUL ARYA, University of California, Berkeley, United States

SEAN MURPHY, University of California, Berkeley, United States

ACM Reference Format:

Group 25, Hansung Kim, Linyue Song, Rahul Arya, and Sean Murphy. 2018. Hybrid Progressive Photon Mapping. In . ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

While the path tracing algorithm [Kajiya 1986] is effective at producing photorealistic renders for non-reflective and transmissive materials, they fall short in rendering more complex light transportation effects such as caustics in terms of sample efficiency.

A well-known technique that mitigates this issue is photon mapping. It traces the light path in an opposite way to path tracing, by tracing a photon from the light source to the surface and accumulating the energy of photons to estimate irradiance. However, from our experiments, photon mapping in turn has disadvantages in rendering diffusive materials, posing a tradeoff between photon mapping and path tracing.

In this paper, we solve this problem by introducing a hybrid approach that combines advantages of both path tracing and progressive photon mapping. We achieve a higher rendering quality for caustics with less number of samples than a render that purely relies on path tracing or photon mapping. Our contributions are as follows:

- We introduce **hybrid progressive photon mapping**, which uses progressive photon mapping to only render the caustics of the scene and uses pure path tracing to render the rest.
- We construct a separate BVH structure to accelerate updating of hit points in the photon mapping pass, which is an optimization not mentioned in the paper.
- We implement all components of our renderer from scratch on GPU, using CUDA. This brings performance improvement compared to our CPU path tracer implementation from the previous assignments.

2 BACKGROUNDS

2.1 Traditional and progressive photon mapping

Computing caustics accurately is crucial in producing highly realistic renders that closely mimic the way that light interacts with reflective and refractive surfaces in the real world, yet it's a complex and computationally intensive

Manuscript submitted to ACM

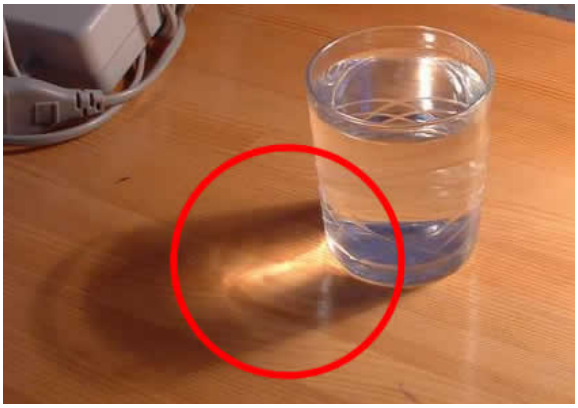


Fig. 1. Light being refracted through a glass of water. Image source: `https://www.interstation3d.com`.

task for pure Monte Carlo based path tracers to perform. Jensen introduces Photon Mapping rendering [Jensen 1996], which shoots photons from the light sources and stores the photons in a photon map data structure as they hit surfaces. During the rendering phase, the flux value accumulated at each hit point are used to compute the final radiance for each pixel. This method can efficiently simulate caustics. Because photon mapping traces photons in its physically-based direction of travel, we get a better rendering of caustics which are essentially concentrated patches of light refracting through a surface. An example of such a caustic is shown above in Figure 1.

However, traditional photon mapping has a significant disadvantage in large memory footprint, as all photons have to be saved in a photon map until the rendering phase. This becomes a significant issue for complex scenes with large numbers of refractive objects, where billions of photons are required. To remedy this, Hachisuka [Hachisuka et al. 2008] proposes a technique called progressive photon mapping. It first generates hitpoints from a traditional ray-tracing pass to store the accumulated output color, and then performs multiple passes photon tracing and updates the hitpoint structures at the end of each pass. By improving the rendering iteratively, the program is able to free the photon structures after each pass and thus reducing the memory footprint of the program.

Nevertheless, we find photon mapping by itself to be inefficient to render non-caustic light effects and surfaces, as we will show in Section 4. We outline our approach that potentially solves this problem in the following Section 3.

3 HYBRID PATH TRACING AND PHOTON MAPPING

To overcome the aforementioned limitations of path tracing-only or photon mapping-only approach, we take a hybrid design that combines both mechanisms to achieve high quality renders for both diffuse materials and caustics. The idea is to render different portions of the final image using path tracing and progressive photon mapping, and add them together to produce the final result. Figure 3a and Figure 3b shows an example of different render contributions coming from the path tracing pass and the photon mapping pass, respectively.

Our rendering pipeline consists of **four different passes**: (1) path tracing, (2) BVH construction, (3) photon mapping, and (4) radiance update. The passes are implemented as multithreaded CUDA kernels, and are launched from the CPU after another sequentially. The following paragraphs as well as Figure 2 illustrates each of the passes in detail.

The path tracing pass serves two purposes. First, it renders the contribution in the image that comes from the diffusive or non-specular materials, e.g. the walls and floors of the Cornell box. In addition, it records the metadata

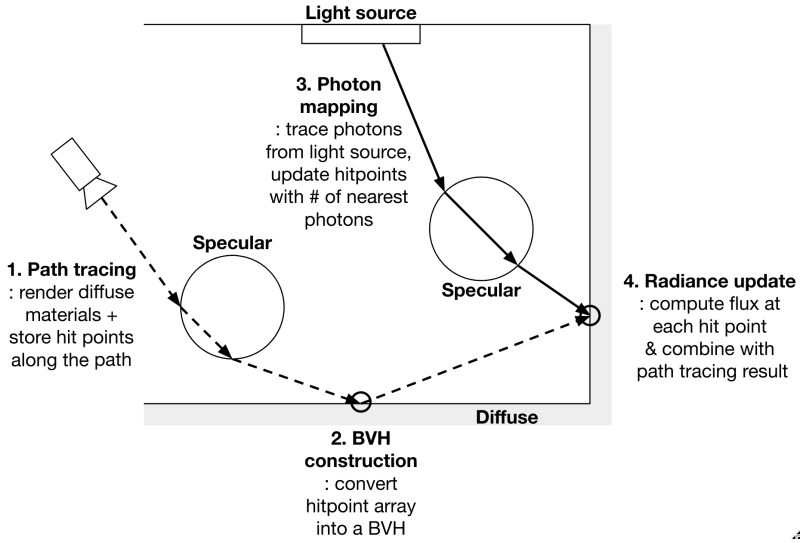


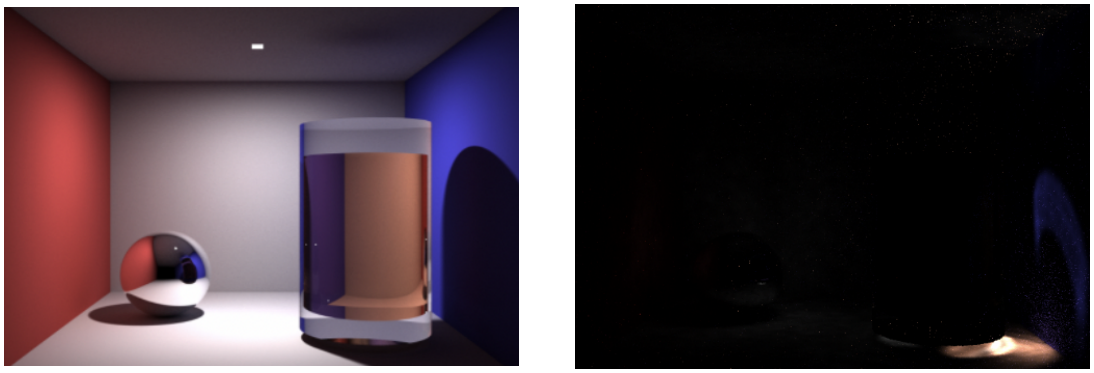
Fig. 2. Illustration of hybrid path tracing and photon mapping, with path segmentation and different rendering passes.

of the hit points at which the path intersects with the scene at each segment into a global data structure. This data structure is for use in the photon mapping pass, where for each traced photons, the hit point data structure should be updated with the newly accumulated photon counts and flux values. Since the hit point structure is a global memory that is shared between multiple threads, new hit point additions are handled with thread-safe atomic operations such as `atomicAdd` and `atomicExch`.

The BVH construction pass converts the simple array structure of hit points obtained from the path tracing pass into a bounding volume hierarchy. This step is necessary because the later photon mapping pass involves searching through the global hit point structure to locate the nearest hit point to a photon. This operation can be computationally expensive when done on a simple array structure, especially if the number of hit points is very high (up to 10 million for our Cornell box scene). Spatial trees such as k-d trees and BVH trees are example acceleration structures that are better suited for such nearest-neighbor search operations. We outline our BVH construction algorithm in Section 3.2.

The photon mapping pass implements progressive photon mapping [Hachisuka et al. 2008] to trace photons shot from the light source to randomly sampled outgoing directions. The photons are traced along the path to any specular transmissive materials, e.g. refraction through glass, until they terminate by hitting a non-specular diffusive material. At termination, the hit point BVH tree is queried with the location of the photon's intersection point, and the hit point that is located nearest to the photon is found. The hit point is then updated with new metadata, such as incremented photon hit count and shrinked radius value.

One advantage of the progressive photon mapping algorithm is that once a photon is accumulated into the hitpoint metadata, it is no longer needed to be stored in the memory. Therefore, we can repeat this photon mapping pass as many times as needed to achieve desired visual quality. However, since the GPU supports parallel execution of very high number of threads, a single photon mapping pass with a higher number of photon proves to be sufficient for our implementation.



(a) Path tracing, for diffusive material.

(b) Photon mapping, for reflective and transmissive material.

Fig. 3. Render contribution from different passes of our renderer. 512 samples per pixel for path tracing (3a) and 200k photon (3b) are used.

Lastly, **the radiance update pass** goes through the hitpoint data and computes the final radiance value for each of the screen pixels that corresponds to the hitpoint. The computation of the radiance follows the original paper. However, one important difference is that this pass adds the computed radiance to the radiance value coming from the path tracing pass to produce a final render that combines the two.

3.1 Path segmentation for determination of rendering method

In order to figure out which part of the render has to be contributed from photon mapping and vice versa, we implement a simple segmenting algorithm of a path. We divide the light path that connects the camera and the light source at the **last hit point on a diffusive surface, after which the path must contain traversal of at least one reflective or transmissive material.** This is illustrated in Figure 2. This is because as long as a path contains a diffusive hit point, path tracing must be used to render the effect of Lambertian distribution and interreflection. Furthermore, if the path traverses through a glass material before reaching the light source, it means the path renders part of a caustics pattern, where photon mapping should be used.

3.2 BVH construction algorithm for hit points

Our BVH construction follows the SAH algorithm [MacDonald and Booth 1990]. At every tree level, we find the best split axes and and position according to the SAH cost model. Then, we reorder the original array structure that stores hit points according to which side of the split each hitpoint is. We recurse this process down the tree until the SAH deems further splitting is unnecessary.

Usage of an acceleration structure is not specified in our reference paper [Hachisuka et al. 2008]. Therefore, we consider using BVH as part of our improvement.

BVH construction is currently done on the CPU with the resulting structure copied to the GPU using `cudaMemcpy`. Offloading construction to the GPU using a parallelizable builder such as LBVH [Lauterbach et al. 2009] is left as future work.

4 RESULTS

We use a simple modified Cornell box scene for evaluation that features a glass material with a higher refractive index than 1, with interesting shapes such as solid filled glass or empty ring cylinder.

Figure 4 shows the rendered images from our renderer. Each row shows different scenes, and each column shows configurations of **(1) path tracing only, (2) photon mapping only and (3) our hybrid approach**. As we can see, the path tracing only renders do a good job at rendering diffusive materials, but produce very noisy caustics. This is because rendering caustics relies on a low-probability events, where light reflected from the surface has to re Photon mapping only renders on the other hand produce low-noise caustics but high variance diffusive renders. This is because much fewer number of photons arrive at the diffusive surfaces compared to caustics, increasing variance of the flux estimation.

The renders roughly take (1) 2 seconds for path tracing only, (2) 11 seconds for photon mapping only, and (3) 6 to 7 seconds for our hybrid approach, with little variance over different scenes. While our hybrid render takes more time than path tracing only because it contains a separate photon mapping pass, it significantly improves the overall quality of the image.

To better compare the sampling efficiency between path tracing and our hybrid approach, we render the same scene with greatly increased 16384 samples per pixel, shown in Figure 5. While the visual noise decreases, there is still a considerable noise around the caustics and we find the caustics pattern looks less realistic. More importantly, the render takes a significantly longer time of 76 seconds, giving it a significant disadvantage in terms of quality to performance ratio.

We note that our GPU implementation of the path tracing pass has a significantly higher performance than the CPU implementation that we did in previous class assignments, where similar renders took roughly 50 times longer time.

5 CONCLUSIONS AND FUTURE WORK

By segmenting the light path into two parts where path tracing and photon mapping should be applied, we introduce a novel way of combining two different rendering techniques and achieve higher sampling efficiency and less noise for rendering scenes with complex caustics. We also achieve high efficiency in managing hit point structure for photon mapping by employing a BVH acceleration structure.

Finally, by implementing every component of our renderer from scratch on a multithreaded hardware, we could have a better understanding of the rendering algorithms and their performance implications.

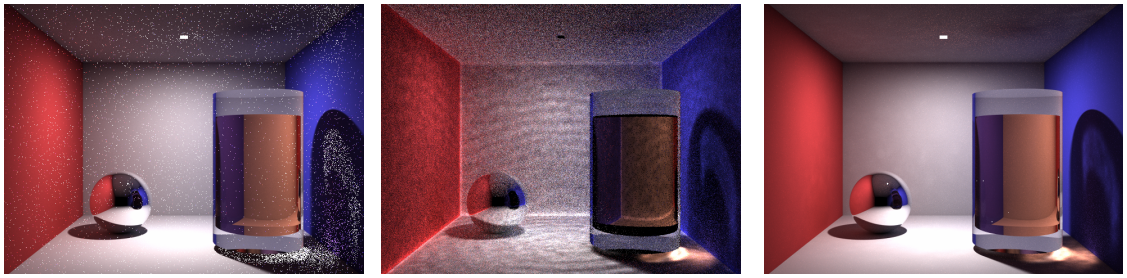
Currently, our renderer does not include a file loader for scene geometry such as the OBJ format, and relies on hard-coded geometry for scene creation. Implementing a scene file loader would allow us to evaluate the rendering algorithm with more sophisticated transmissive geometries. Furthermore, due to time constraints, we could not get to implementing deep learning-based optimization of representing the photon map as we originally planned. We still think this is a promising idea that leverages machine learning in graphics rendering and hope to see it in future work.

REFERENCES

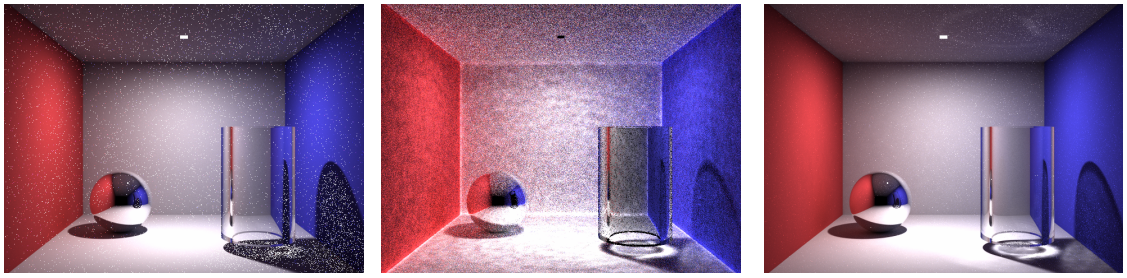
Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. 2008. Progressive Photon Mapping. *ACM Trans. Graph.* 27, 5, Article 130 (dec 2008), 8 pages. https://doi.org/10.1145/1409060.1409083

Henrik Wann Jensen. 1996. Global Illumination Using Photon Maps. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96* (Porto, Portugal). Springer-Verlag, 21–30.

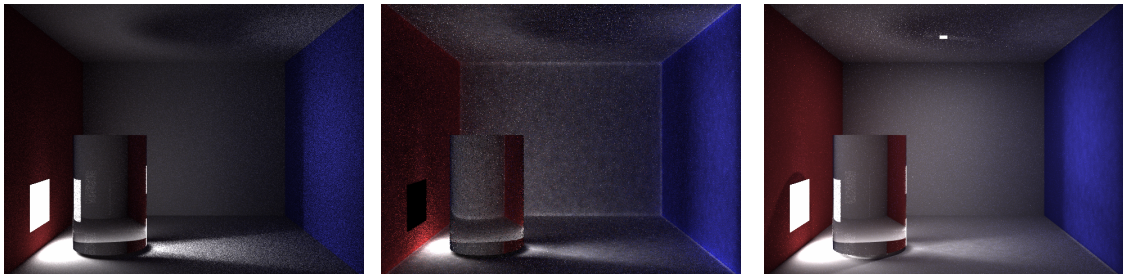
James T Kajiya. 1986. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 143–150.



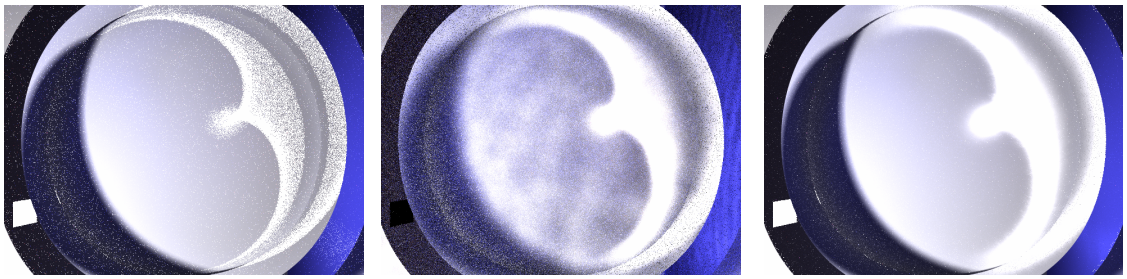
(a) Liquid glass scene, path-tracing only (b) Liquid glass scene, photon mapping only. (c) Liquid glass scene, our hybrid approach.



(d) Empty glass scene, path tracing only. (e) Empty glass scene, photon mapping only. (f) Empty glass scene, our hybrid approach.



(g) Solid glass scene, path tracing only. (h) Solid glass scene, photon mapping only. (i) Solid glass scene, our hybrid approach.



(j) Solid glass scene, path tracing only. (k) Solid glass scene, photon mapping only. (l) Solid glass scene, our hybrid approach.

Fig. 4. Renders produced with our renderer. For the path tracing pass, 512 samples per pixel are used; for photon mapping pass, 2 million photons are used except for the liquid glass scene with 200k photons.

Christian Lauterbach, Michael Garland, Shubhabrata Sengupta, David Luebke, and Dinesh Manocha. 2009. Fast BVH construction on GPUs. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 375–384.

J David MacDonald and Kellogg S Booth. 1990. Heuristics for ray tracing using space subdivision. *The Visual Computer* 6, 3 (1990), 153–166.

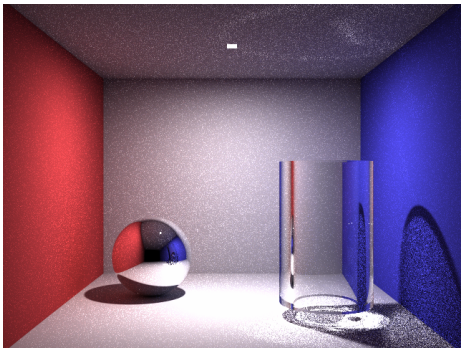


Fig. 5. Empty glass scene, rendered with 16384 samples per pixel using path tracing only. Even with a significantly higher number of samples the render quality is inferior to our hybrid render, shown in Figure 4f.