Final Project:
Photon Mapping

GROUP NUMBER: 5
MEMBER 1: CHUNHAO BI
MEMBER 2: HONGDI YANG
MEMBER 3: MUTIAN LI

1 INTRODUCTION

This is the final project of CS171.01. In this project, the photon mapping method for rendering is introduced, and some comparisons are carried out.

2 ALGORITHM

2.1 Photon mapping

The standard photon mapping includes two passes. The first is to generate photons from all light sources, and the photons will bounce between objects and stops when meeting terminate conditions. The photon tracing is similar to path tracing, and the photons are stored in a tree for later use. The second pass is ray tracing, in which rays are traced until they meet a diffuse surface. Then the radiance is calculated by summation of photon energy nearby, which is searched from the photon tree.

Photon mapping is good for dealing with small light sources, as it seems to sample light directly. Meanwhile, it is significantly good at capturing caustics lighting through occlusion or refraction.

2.2 Progressive Photon Mapping

In PPM, the two passes are converted, which reduces the memory cost due to that ray tracing hit points are much fewer than photons. It also introduced a method to change the radius of photons (or viewpoints) by carrying out rounds of photon tracing part rather than one. This makes it possible to use fewer photons with bigger radii to get a satisfying result rather than much more photons with small radii.

2.3 Stochastic Progressive Photon Mapping

SPPM is an extension of PPM for special ray tracing and camera properties, which includes glossy reflection, motion blur and depth-of-field. It uses multiple ray tracing between photon tracings to handle these situations.

3 IMPLEMENTATION

3.1 Rendering pipeline

In this project, we adopt SPPM algorithm for photon mapping.

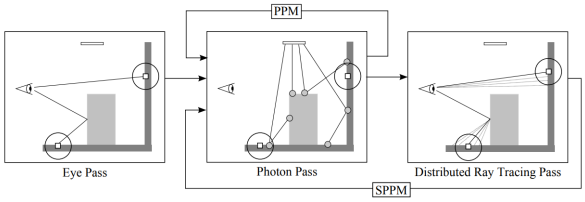


Fig. 1. SPPM pipeline

As the above figure shows, we do one distributed ray tracing pass and one photon pass each round. The basic pipeline is shown as below.

1. Prepare for rendering. Set sample per pixel spp, number of photons ptn_num, initial search radius $init_r$, attenuation coefficient α, total render round rnd_num and initial photon energy $e = \frac{1-\alpha}{1-\alpha^{rnd\_num}}$
2. Reset viewpoints and temporary pixel energy, make sure they are clear before ray tracing pass.
3. In every pixel, we shoot number of spp rays randomly, and then do Ray tracing pass to get the viewpoints.
4. Use the viewpoints we get from all pixels to construct a viewpoint KD-tree.
5. We randomly generate ptn_num rays from light source, and do Photon tracing pass with current search radius r.
6. We update the pixels color using temporary pixel energy calculated in photon tracing pass.
7. Update search radius $r = r * \alpha$, update photon energy $e = e * \alpha$.
8. If current render round > rnd_num, stop rendering and output the image. Otherwise, go back to step2 to start next round.

3.2 Ray Tracing

Ray tracing is the first pass in both PPM and SPPM. This part is not like path tracing but simpler. Each ray generated from the camera will stop right after hitting a diffuse surface. Only if the hit surface is not a diffuse one does it sample another ray to trace.

After hitting a diffuse surface, the position will be stored as a viewpoint. Later in photon tracing part, the radiance of the viewpoint will be calculated, and the color will be set back to the corresponding pixel.

Here the viewpoint are stored in a balanced KD-Tree for speed reason. The intersection part uses BVH for that triangles may intersect with different bounding boxes in KD-Tree which results in inefficiency.

3.3 Photon Tracing

In standard photon mapping method, photons are stored and each viewpoint needs to find all photons within a radius. Here we implemented SPPM, where we stored the viewpoints and let the photons find the nearby viewpoints instead. This modification costs less memory when the photon number is large.

In each render iteration, we shoot a certain number of rays from the light, each ray sampled according to the light distribution representing the path of a photon. If a photon hits an ideal diffuse surface, the view points within a certain radius from the hitting point will be illuminated, and the color of the corresponding pixels will be updated according to the viewpoint color from the object and the photon radiance. Whenever a photon hits a geometry surface, a new ray will be sampled according to the brdf of the surface, and the photon radiance will update by multiplying the brdf value. Here's the rendering equation:

$$L(S,\omega) = \lim_{i\to\infty} \frac{\tau_i(S,\omega)}{N_e(i)\pi R_i(S)^2} \tag{1}$$

where $N_e(i)$ is the photon number, $R_i(S)$ is the radius and $\tau_i(S,\omega)$ represents the total radiance, where

$$\tau_i(S,\omega) = \sum_{p=1}^{N(x)} f_r(x,\omega,\omega_p)\Phi_p(x_p,\omega_p) \tag{2}$$

And the new photon energy is combined with the brdf:

$$\Phi_p^{'} = \Phi_p f_r \tag{3}$$

For every photon, we apply the algorithm above.

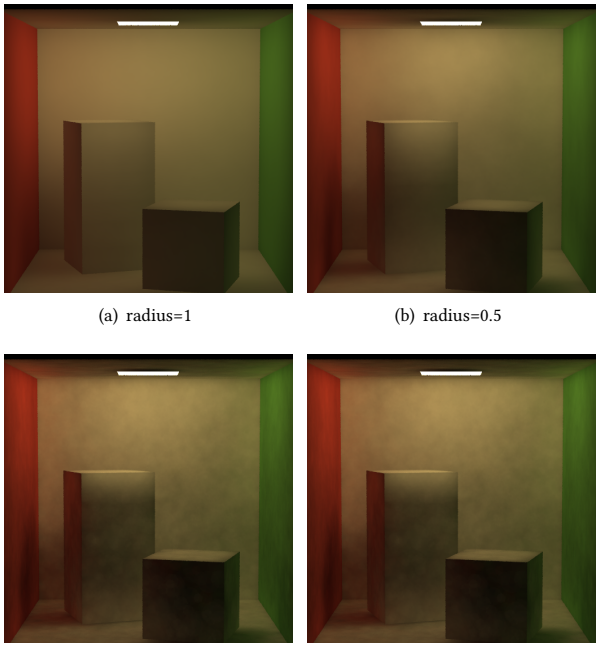
3.4 Contribution

Member 1. Chunhao Bi : Ray tracing and Path tracing, Materials and Texture, Algorithm Corrections.
Member 2. Hongdi Yang : Rendering pipeline, Light settings, Scene construction.
Member 3. Mutian Li : Photon tracing, Color calculation

4 EXPERIMENTS

4.1 Radius

This algorithm can render realistic pictures only if the initial radius is within an appropriate range. If the radius is too large, some occluded area will be illuminated, and some shadow will disappear. If the radius is too small and the number of photons is insufficient, the output image will be noisy on account of bright areas around the photons, and this problem can be solved by increasing the number of photons. Generally, decreasing the initial radius and increasing the number of photons simultaneously will render a more realistic image. Fig.2. shows the render results with different initial radius, when the number of photons is 10000, with $spp = 1$, total render round = 10, attenuation coefficient $\alpha = 0.8$, search depth $depth = 8$.



(a) radius=1 (b) radius=0.5

(c) radius=0.25 (d) radius=0.125

Fig. 2. Radius

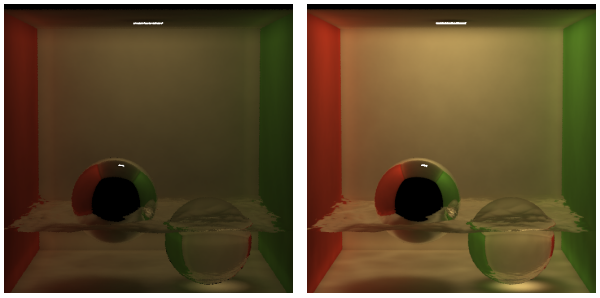
A smaller update coefficients will accelerate the convergence, and the image will be more realistic under the same number of total render rounds, but might become noisy during the last several rounds, and the reason is the same as a small radius. Therefore, increasing the number of photon will overcome the unexpected noise. The following figure shows the results with different attenuation coefficients, when the number of photons is 10000, with $spp = 1$, total render round = 10, initial search radius $r = 0.5$, search depth $depth = 8$.

(a) coefficient=0.8 (b) coefficient=0.7 (c) coefficient=0.5

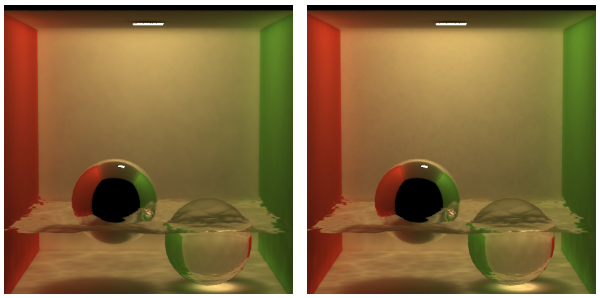
Fig. 3. Attenuation coefficient

4.2 Round number

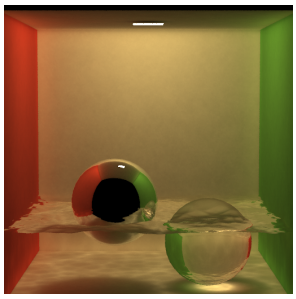
Here we show how our render result progress with render round increases.



(a) Render round 0 (b) Render round 4



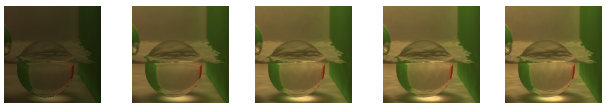
(c) Render round 9 (d) Render round 14



(e) Render round 19

Fig. 4. Render results

As Figure2 shows, as render round increases, our render result gets brighter and more realistic. As we decrease the search radius every round, the calculated radiance will converge to the real radiance. And this will be quite efficient for caustics effect as there will be many rays hitting a relatively small area through translucent objects.



(a) Round 0 (b) Round 4 (c) Round 9 (d) Round 14 (e) Round 19

Fig. 5. caustics effects

As Figure3 shows, the caustics effect become much better as the render round increases.

Also, as search radius decreases, there will be fewer photons found around each viewpoint, thus making the render time faster. Under the parameter of : $spp = 1$, photon number = 200000, initial search radius $r = 0.15$, total render round = 20, attenuation coefficient $\alpha = 0.8$, search depth $depth = 32$, we record the time for some render rounds.

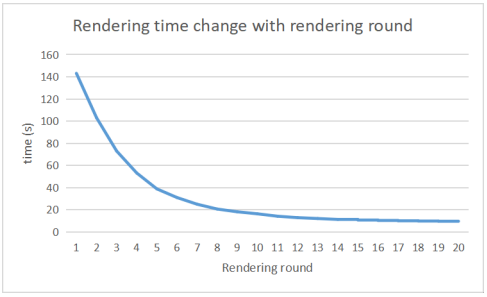
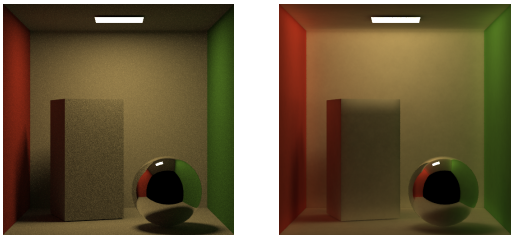


Fig. 6. Rendering time change with rendering round

4.3 Comparison

Comparison between different render methods is important. For photon mapping method, if the parameters are set well, the converging speed will be much higher comparing to basic path tracing method.

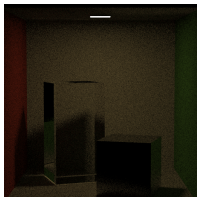
We tried rendering the same scene with Monte-carlo ray tracing and our method. In ray tracing, the parameters are : $spp = 256$, search depth = 8. In photon mapping, the parameters are : $spp = 1$, photon number = 200000, initial search radius $r = 0.15$, total render round = 20, attenuation coefficient $\alpha = 0.8$, search depth $depth = 8$. Here are the results.



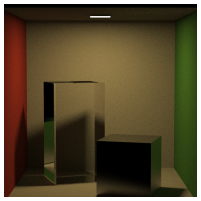
(a) Monte-carlo (b) SPPM

For Monte-Carlo ray tracing, it takes 1313s to render. For photon mapping, it only takes 502s.

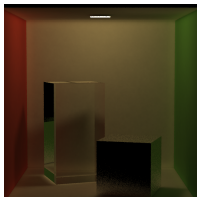
There are some metrics in path tracing, which use different method to make it converge faster, such as direct light sampling, bidirectional sampling and metropolis sampling. However, those methods are still quite slow for that they are still sampling pixels. Photon mapping changes the rule, which makes it possible for some scenes with severe light condition. Here we compared the results of naive path tracing, path tracing with direct light sampling, and SPPM with the same rendering time of about 550000ms.



(c) Monte-carlo



(d) Direct light sampling



(e) SPPM

Fig. 7. Same scene rendered by different methods

From the figure above, the naive method converges really slow because of the small light source to reach. The direct light sampling method does quite good on the walls, but because the translucent boxes are considered light-tight, the shadow area converges as slow as using naive method. This converging speed difference results in misleading understanding of the scene. The SPPM method gets a good result for the scene with the translucent part. The shadow of the long box is quite soft which representing a far more converged result. However, the result seem not so good at glossy materials telling from the noise on the short box compared to direct light sampling.

5 RESULTS

Here are some of our final render results.

Parameters: $spp = 1$, photon number = 200000, initial search radius $r = 0.15$, total render round = 20, attenuation coefficient $\alpha = 0.8$, search depth $depth = 32$

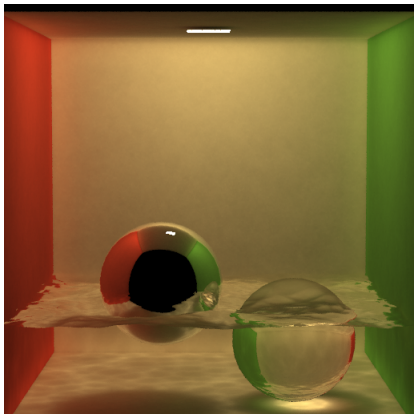


Fig. 8. Scene with Water and spheres

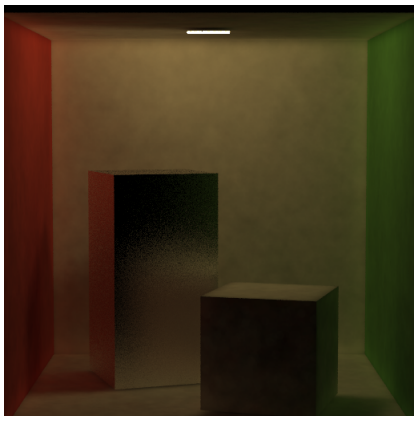


Fig. 9. Cube with glossy surface

Fig. 10. An ice-cream bowl made of glass

Fig. 11. Dragon made of glass

Fig. 12. A flower pot made of glass

Fig. 15. Ice dragon

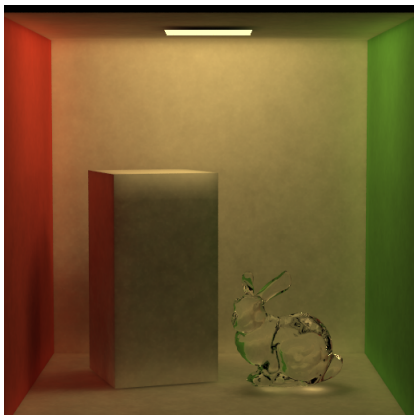


Fig. 13. diffuse cube and translucent bunny

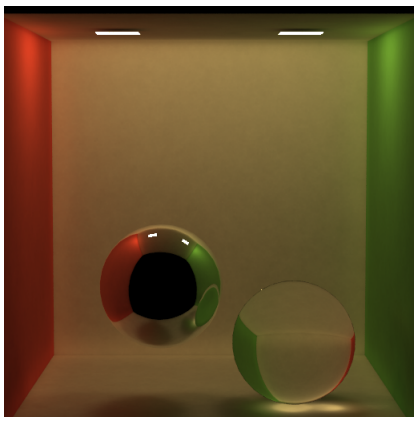


Fig. 14. Scene under multiple lights