Multithreaded Bidirectional Path Tracing

Anthony Handwerker and Nicholas Colclasure

Abstract:

Our paper presents a method for capturing realistic global illumination effects quickly through path tracing. Typical ray tracing gives struggles to take into account global illumination effects such as color bleeding and caustics. Additionally, raytracing is a relatively slow rendering method, especially when not done in parallel. While there are methods to introduce more global illumination effects to ray tracing, generally through multiple passes, this methods can be complicated and cumbersome. Bidirectional path tracing solves these issues by creating a unified framework for global illumination effects which is easily parallelizable.

Related Works:

Kajiya's "The Rendering Equation" covers integrating global illumination effects along with direct lighting effects into a unified path tracing algorithm, and was the basis of our initial pass at making a single directional path tracer. "Bi-Directional Ray Tracing" by Lafortune et al. was then used as a reference for bidirectional path tracing. "Parallel Path Tracing using Incoherent Path-Atom Binning" by Coulthurst et al. provided a framework for our parallelization model and pipeline.

Deficiencies of Basic Ray Tracing:

Basic Ray Tracing lacks the ability to take into account many global illumination effects. Ray tracing cannot account for reflected or refracted light from any sources, including diffuse surfaces. This is problematic in many types of scenes, as indirect illumination is important to most every-day scenes. In buildings, lighting is often specifically indirect, which cannot be accurately modeled by ray tracing, but even when the lighting is direct it looks very unnatural for unlight areas to be completely black. While this can be partially corrected by an ambient lighting term, this can cause the scene to look flat. Ambient occlusion serves to highlight details on surfaces which would otherwise be missed. Additionally, the variability in number of rays cast based on the reflective and refractive objects in the scene can cause significant variability in render times. Traditional raytracing requires multiple rays be cast upon each reflection or refraction, both simple shadow rays and fully traced rays, and the numbers of these additional rays can vary greatly from scene to scene.

The Advantages of Path Tracing:

Path tracing is a simple way to include global illumination effects. A path is traced, starting at the eye and going into the scene, which then bounces from surface to surface until it is terminated. Each bounce can be reflective, refractive, or diffuse, and the path they form represents a potential path for light to follow from the lightsource to the eye. This allows the algorithm to take into account global illumination effects such as color bleeding, as pixels will query many other points in the scene which can then contribute lighting to go to the eye.

The key to making the global illumination effects visible despite the probabilistic scattering of the rays is the number of rays cast. Similarly to how distributed ray tracing can account for effects otherwise hard to mathematically represent, sending a larger number of initial rays makes up for not being able to follow all the possible paths the ray can take, effectively utilizing Monte Carlo integration. Unfortunately, path tracing still biases the render to the eye. In scenes where indirect lighting is important path tracing can fail to deliver an accurate image.

The Advantages of Bidirectional Path Tracing:

Bidirectional path tracing utilizes paths from both the light and the eye in order to render the image. This is possible because of the Helmholtz reciprocity, which allows us to calculate both paths using the same algorithm as well as easily calculating the contribution of the light path to the final result. In addition to paths traced from the eye, paths are traced from the light through the scene, and these points are queried for their light contribution by the eye path. Since a path can only contribute light to a pixel if it reaches a light, single path tracing runs into the problem of not reaching the light. Even when shadow rays are used to check paths to the light at each bounce, many paths of light are not taken into account because they are a large number of bounces from the eye. Bidirectional path tracing ensures that the paths investigated are representative of both paths with high contribution from the light and high contribution to the final image. This provides a good compromise between bias towards the eye and bias towards the light.

The Pipeline:

Our path tracing pipeline is split into five parts; the producer, the caster, the bouncer, the shadow caster, and the draw function. Each of these five parts is given its own thread, and the caster, bouncer, and shadow caster can be given multiple threads. Producer/Consumer queues are used to pass information between the threads without the threat of concurrency errors.

The producer thread is also our main thread; it both initiates the other threads, and creates all of the initial rays to cast. It produces a set number of rays from the camera that pass through each pixel, and produces a random diffuse ray from the light source for each ray created from the camera. Each of the rays produced here gets passed into the caster thread.

The caster thread takes each ray and finds its first intersection point with some other object, and calculates the intersection position. From here, it passes the ray to one of 3 places; if the ray intersected with something, it goes to the bouncer. If the ray didn't intersect with anything, and came from the camera, it goes to the shadow caster. If the ray didn't intersect with anything, and came from the camera, it gets passed to the draw function.

The bouncer thread takes the input ray and collision and randomly picks which way to send a new ray, based on the different colors of the material. From here, the rays are sent to one of three places. There is a chance, based on the colors of the material, that we may terminate the path and send it to the shadow caster. If the ray came from the shadow caster, it sends the appropriate color to the draw function. Otherwise, we send the newly produced ray back to the caster.

The shadow caster thread pairs up camera paths and light paths. For each pair, it makes new rays that start at each point along the camera path that points in the direction of each point

along the light path. It labels these rays as having been through the shadow caster, and sends them back to the caster thread. When these rays reach the bouncer thread, it will check to see if the thread collided with the expected point; if it did, it sends the colors that have been gathered by the paths; otherwise, it sends black ({0, 0, 0}).

Finally, the drawing thread accepts color-pixel pairs and records them in a matrix that represents the pixels of the screen. Once all the rays have been recollected, the drawing thread dumps the matrix to the screen.

Conclusion/Results:

We managed to get fairly pleasing results from our path tracer, though there is clearly some tweaking and optimization that could be done. It produced results much faster than a raytracer would; and, in addition, they had the added global illumination effects. These effects were particularly subtle in the bidirectional version of the code, but still present. Our bidirectional version makes caustics visible, but that and other effects are muted. Their lack of visibility is likely due to the limitations of our Monte Carlo sampling, as we don't have any form of importance sampling implemented.

Figures 1-6 show our unidirectional renders of a cornell box with a diffuse sphere, a reflective sphere, and a hollow glass sphere inside, with various amounts of sampling. The color bleeding effects are very pronounced, and the image is fairly converged by 200 samples. Figures 7-12 show the same scene, but rendered with our bidirectional path tracer. One can see that the images are much more muted, but converge in about the same amount of time as the unidirectional render. While it is hard to see the color bleeding in these images, it is most heavily pronounced on the diffuse sphere. Figures 13 and 14 show a unidirectional rendering of a metal ring; though a proper rendering would show a caustic, these do not. This is where figures 15 and 16, and the bidirectional renderer, shine. In these images, though it is faint, one can see the cardioid caustic.

The project as a whole took around 30 hours to complete. Anthony worked on the pipeline framework, which controlled how rays were passed between the different parts of the system and how the results were collected and displayed. Nicholas created the lighting function, which determined where rays were to be cast and and how the results of the ray casts would be used.

Below, we give a table of render times of our unidirectional a bidirectional path tracers, based on the number of samples per pixel, when rendering a scene in a cornell box with a diffuse sphere, a reflective sphere, and a hollow glass sphere.

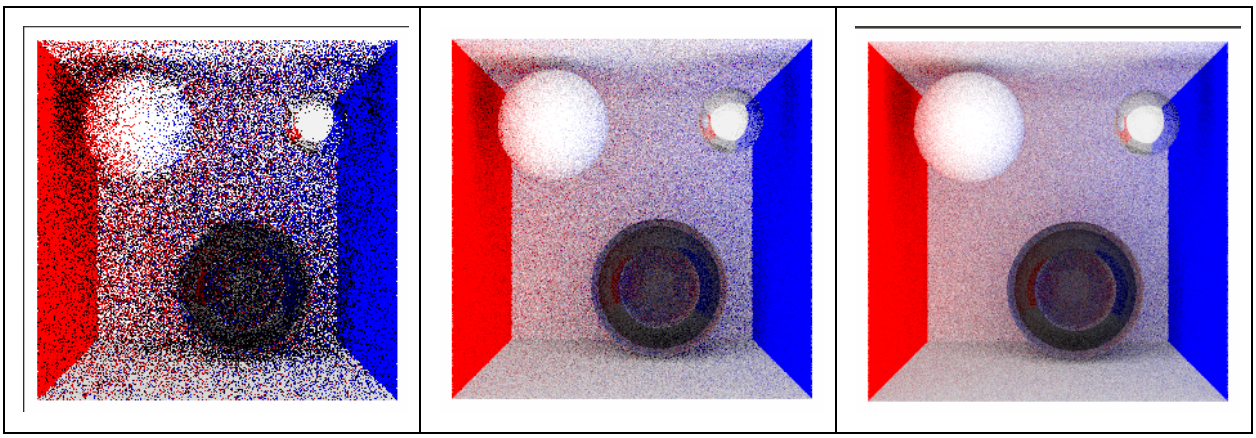
Samples per pixel	Unidirectional Render Time (mm:ss)	Bidirectional Render Time (mm:ss)
1	00:04	00:09
10	00:25	01:44
25	01:25	06:24
50	02:51	08:57
100	05:38	17:20
200	11:20	22:25

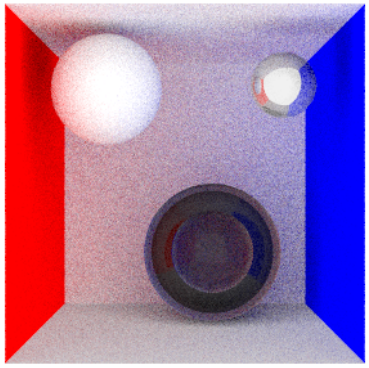
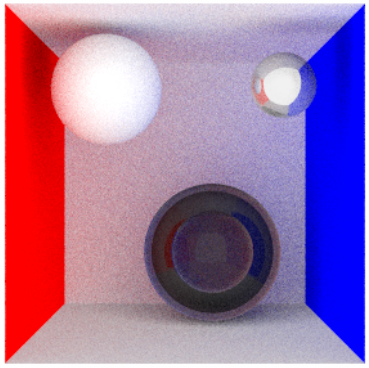
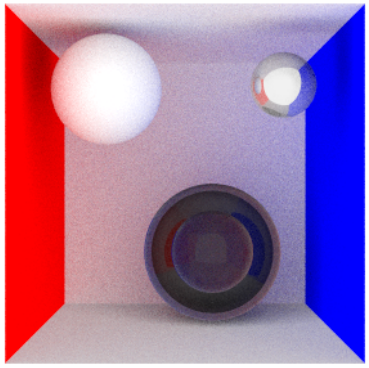
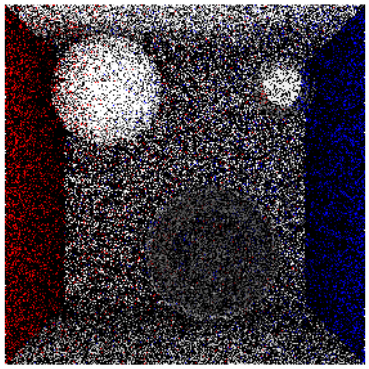
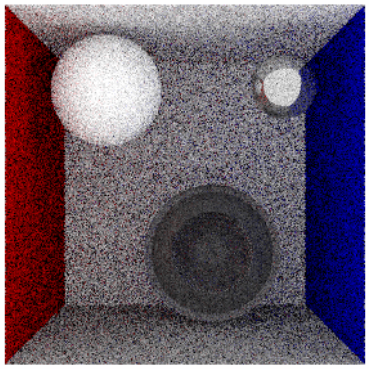
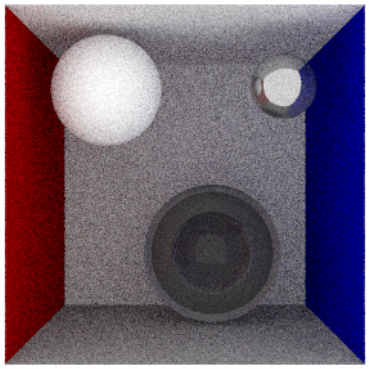
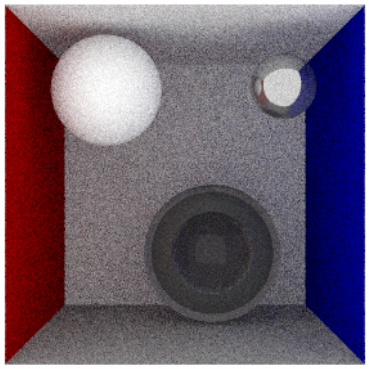
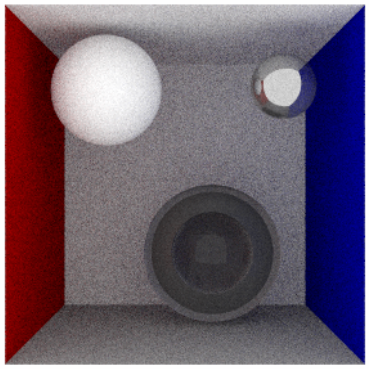
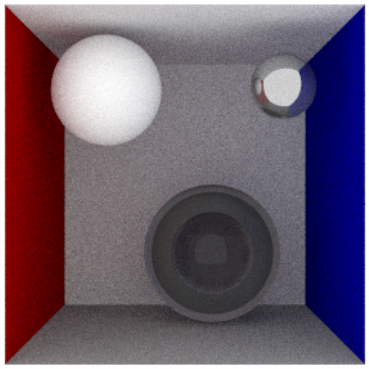
Improvements:

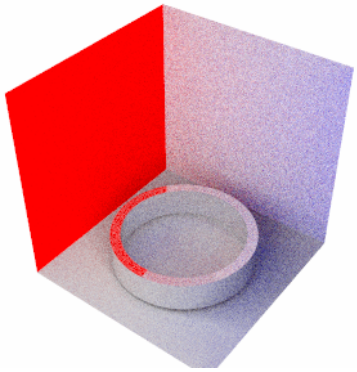
There are several ways we can work to improve the visual outcome of our code and make it run faster. From the visual side of things, there are tweaks that could be made to make our renders look a little more accurate; particularly in order to make renders of transparent objects look brighter. In addition, our path tracer does not handle open scenes very well; often, things will get washed out, particularly reflections or transparent surfaces. As for optimizations for speed, we could cut out a large number of raycasts by not making shadow casts whenever the diffuse component on either the light path or the camera path is {0, 0, 0}. We could do this because if the diffuse component on either side is {0, 0, 0}, the total contribution ends up becoming {0, 0, 0} as well.

Additionally, the path tracer could be modified to render more complex materials with a more complete BDRF model. This could allow materials with directional properties to be rendered, as well as materials such as glossy surfaces. Some other important extensions would be to make the path tracer handle multiple lights; and to use a spatial data structure to hold the mesh, in order to improve the intersection lookups, because the code scales very poorly as more objects are added to the scene.

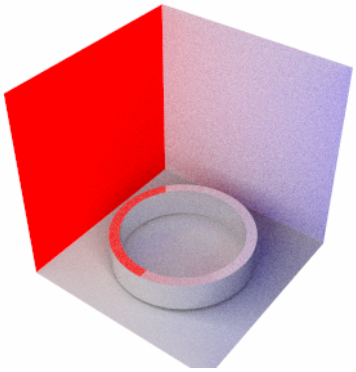
Images:



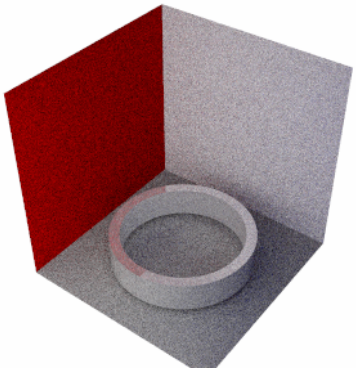
(Figure 1) 300x300 render, 1 sample per pixel, 3.9 second render time.	(Figure 2) 300x300 render, 10 samples per pixel, 25.4 second render time	(Figure 3) 300x300 render, 25 samples per pixel 85.2 second render time
 (Figure 4) 300x300 render, 50 samples per pixel, 170.7 second render time	 (Figure 5) 300x300 render, 100 samples per pixel, 338 second render time	 (Figure 6) 300x300 render, 200 samples per pixel, 680 second render time
 (Figure 7) 300x300 bidirectional render, 1 sample per pixel, 9.3 second render time	 (Figure 8) 300x300 bidirectional render, 10 samples per pixel, 104.4 second render time	 (Figure 9) 300x300 bidirectional render, 25 samples per pixel, 265 second render time
 (Figure 10) 300x300 bidirectional render, 50 samples per pixel, 537 second render time	 (Figure 11) 300x300 bidirectional render, 100 samples per pixel, 1061 second render time	 (Figure 12) 300x300 bidirectional render, 200 samples per pixel, 2116 second render time



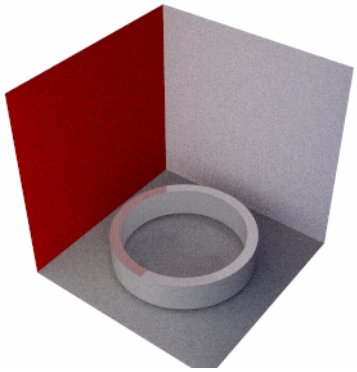
(Figure 13) 300x300 render, 50 samples per pixel, 185 second render time



(Figure 14) 300x300 render, 200 samples per pixel, 762 second render time.



(Figure 15) 300x300 bidirectional render, 50 samples per pixel, 333 second render time



(Figure 16) 300x300 bidirectional render 200 samples per pixel, 1346 second render time

Bibliography:

- Lafortune, E. P. and Willems, Y. D. (1994), "Bi-Directional Path Tracing", in *Proceedings of CompuGraphics* (Alvor, Portugal) pp. 145 - 153 (Dec. 1993)
- James T. Kajiya *The Rendering Equation*. In: SIGGRAPH '86 Proceedings of the 13th annual conference on Computer graphics and interactive techniques pp. 143 - 150
- Coulthurst, David, Dubla, Piotr, Debattista, Kurt, McIntosh-Smith, Simon and Chalmers, Alan (2008) *Parallel path tracing using incoherent path-atom binning.* In: SCCG08 Spring Conference on Computer Graphics, Budmerice Castle, Slovakia, 21-23 Apr 2008. Published in: Proceedings of the 24th Spring Conference on Computer Graphics pp. 91-95.