Ray Tracing and Beyond

JAMIE HONG, ANZE LIU, AKSHIT DEWAN, and TIM TU

ACM Reference Format:

Jamie Hong, Anze Liu, Akshit Dewan, and Tim Tu. 2023. Ray Tracing and Beyond. 1, 1 (April 2023), 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 OVERVIEW

Ray tracing is a powerful technique to render various scenes, but it can also be slow and inefficient, especially when sending out many rays into the scene to render. To address these issues, we plan to begin with a ray tracer and then add optimizations and features inspired by Disney's Hyperion renderer onto it. The challenges in this project involve figuring out how to efficiently optimize our ray tracer, adding new techniques, and exploring how to best demonstrate our ray tracer's ability to render complex scenes.

2 PHOTON MAPPING

2.1 What we have accomplished

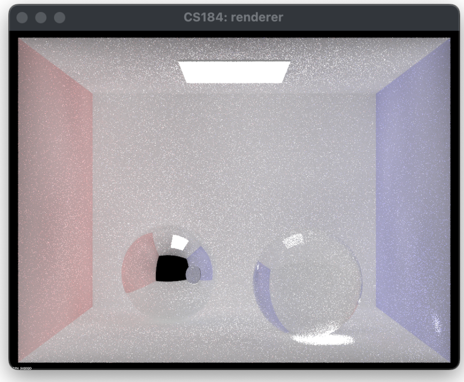
For photon mapping, our goal is to construct photon maps and to render scenes using radiance estimates from our maps.

After reading through the relevant section from "A Practical Guide to Global Illumination using Photon Mapping" [2], we have discussed and compiled a document of notes, showing the general implementation steps in pseudocode. We also have implemented the construction of a global photon map and using global photon map to estimate the radiance. To compute radiance estimate using the global photon map:

(1) Send rays out into the scene
(2) Check for ray-object intersection
(3) Compute the hash of the intersection position
(4) Get the bounding box and the surrounding bounding boxes
(5) Put all the photons in the above bounding boxes together
(6) Get an estimate for the radiance from each nearby photon
(7) Sum the estimates together
(8) Return the estimate of the radiance for this point

Authors' address: Jamie Hong; Anze Liu; Akshit Dewan; Tim Tu.

Manuscript submitted to ACM



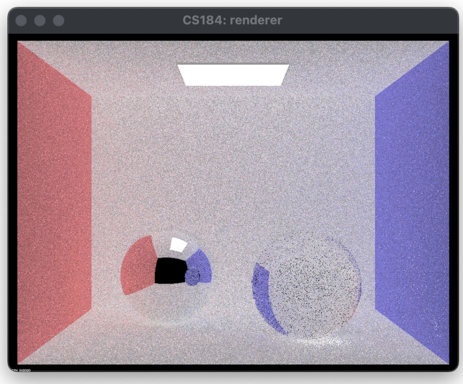


Fig. 1. (a) photon mapping with one bounce radiance

Fig. 2. (b) photon mapping with six-bounce radiance

Fig. 3. (c) photon mapping with bsdf in radiance estimate

2.2 Reflection on progress

We are making progress on photon mapping as planned and able to obtain a preliminary render the Cornell box scene with a glass sphere. However, we still need to tune and experiment with various parameters, for example, the number of nearest photons used to estimate the radiance at a particular intersection of interest, the number of rays sent into the scene to build the initial photon map, and so on. We also need to be able to render the shadows in the scene. We plan to focus on debugging and to efficiently achieve an approximation of global illumination with photon maps.

3 CACHE POINTS OPTIMIZATION

3.1 What we have accomplished

After reading through the Cache Points section in the paper, "The Design and Evolution of Disney's Hyperion Renderer" [1], we discussed the general approach and took notes.

(1) Motivation: looping over every light source for each point is expensive. Instead, we can "cache" region-specific lighting information into a certain number of cache points. From these, we can then use M of the nearest cache points to sample the necessary lighting information, saving in computation during the rendering phase.
(2) Pre-compute phase: building cache points
    (a) Initialize X number of cache points for bounding boxes by random distribution
    (b) For each cache point, create a single, discrete distribution over light sources from a union of 7 distributions (6 orientations + 1 omnidirection)
(3) Light sampling and aggregating phase: using cache points
    (a) For each point of interest, sample light from the closest M cache points: sample one light from each cache point's discrete probability distribution.
    (b) The light at the point of interest is computed to be the weighted average of light sampled from each cache point with the per-light visibility estimates as coefficients.

3.2 Reflection on progress

As we focused more on implementing photon mapping, we have not been able to implement cache points optimization yet. However, we have drafted the pseudocode, as summarized above, and will be implementing cache points soon.

REFERENCES

[1] Brent Burley, David Adler, Matt Jen-Yuan Chiang, Hank Driskill, Ralf Habel, Patrick Kelly, Peter Kutz, Yining Karl Li, and Daniel Teece. 2018. The Design and Evolution of Disney's Hyperion Renderer. *ACM Trans.Graph.* 37, 3 (July 2018). https://doi.org/10.1145/3182159
[2] Henrik Wann Jensen. 2002. A Practical Guide to Global Illumination using Photon Mapping. *Siggraph 2002 Course 43* (July 2002).