CS-714 Project Report
Photon Map for Visualization of Caustics

Ankita Victor
IMT2014005

Abstract

Photon maps makes it possible to efficiently compute global illumination including caustics, diffuse color bleeding, and participating media, by storing illumination information in a map. The method is capable of handling material interactions among a mixture of specular, diffuse, and refracting surface [1]. This report describes the implementation of a photon map that supports spheres, axis aligned planes and point lights, a global map that maintains direct illumination and other specular reflections, and its visualization. The report will discuss the photon-surface intersection algorithms, the photon tracing algorithm (specular reflection, refraction and diffuse reflection), and the photon map data structure implementation. The model used to test the implementation is a variation of the Cornell Box comprising spheres with pure specular and refractive material properties and diffuse walls.

1 Introduction

A caustic is formed when light reflected from or transmitted through a specular surfaces strikes a diffuse surface. This causes brightened areas to appear. Caustics are an addition to rendering that calculates photons of light starting at a light source and being propagated through the scene accurately simulating the ways light can move through a scene.

Compared to the total number of light rays starting from the source and reflected by an object, only a select few of them ever reach the eye. A computer simulation would have to potentially have to cast a very large number of photons from the light source to find a sufficiently large subset that would strike the eye. Hence, ordinary ray tracing works backwards—the rays start at the camera. The scene is scanned from the camera's point of view. Where the camera sees a reflective or refractive object, the ray that started at the camera may be bounced or bent and projected off that object into another direction. Only the rays from the camera are bent or bounced by ray tracing.

The reason that ray tracers are not well suited for generating caustics is that when an incoming ray from the camera or eye lands on a spot where a caustic

should be generated, the reflected ray is generated in only one of two directions—either the direction determined by the specularity of the surface, or the direction towards the light. However, caustics are a concentration of light in small spot and are much brighter than surrounding areas because a large number of light rays have been focused there. A ray which lands on this spot would somehow have to reverse this focusing phenomenon and guess a direction, not necessarily directly toward a light source, that will be focused at the light source.

Using a photon map avoids the need to reverse the focusing phenomenon that would be needed with ray tracing, and instead emits photons from a given light source and follows them until they land on diffuse surface, where they are stored. Photons are stored in a data structure called a called a photon map which is queried during rendering to get irradiance values at a given surface location.

Compared with finite element, global illumination techniques like radiosity, photon maps have the advantage of requiring no meshing—only appropriate intersection algorithms depending on the surface construction. The radiosity algorithm is faster for simple diffuse scenes but as the complexity of the scene increases, photon maps tend to scale better. Also the photon map method handles non-diffuse surfaces and caustics. Monte Carlo ray tracing methods such as path tracing, bidirectional path tracing, and Metropolis can simulate all global illumination effects in complex scenes with very little memory overhead. The main benefit of the photon map compared with these methods is efficiency, and the price paid is the extra memory used to store the photons [1]. This implementation of the photon map works with spheres, axis aligned planes, and point light sources. Two photon maps are maintained—a caustic photon map to capture LS^+D and a global photon for other direct and indirect illumination $L(S|D)^*(S|D)$.

2 Scene

The scene used to simulate this implementation of the photon map is a variation of the Cornell Box. It comprises five planes intersecting to form a box and two spheres—one perfectly specular or a perfect mirror, and another refracting sphere. If the photon map is implemented correctly the caustic should form a bright spot at the base of the refracting sphere, since the refracting is expected to concentrate light at a spot depending on the position of the light source (and based on observation of implementation results).

This implementation allows scene's light source to be moved left, right, in and out, using keyboard controls (*l, r, i, o*) to view how the scene's illumination, and correspondingly the caustic, changes. New spheres can be added by adding two lines of code in the initialization function—creating the sphere object and pushing it into the scene's list of spheres. Each sphere is primarily defined by its center, radius and material property—specular or refracting. Each plane is defined by its aligning axis, distance from the origin and color, and have all been

assumed to be diffuse. The planes are effectively infinite, the box is defined by their intersection.

3 Ray-Surface Intersection

A ray can be expressed in parametric form as -

$$O + tD$$

where O represents the origin, D represents the direction, and changing the value for t makes it possible to define any position along the ray. When t is strictly positive, the point is located in front of the ray's origin, when t is equal to 0 the point coincides with the ray's origin, and when t is negative the point is located behind its origin. From Figure 1 -

$$P = O + tD$$

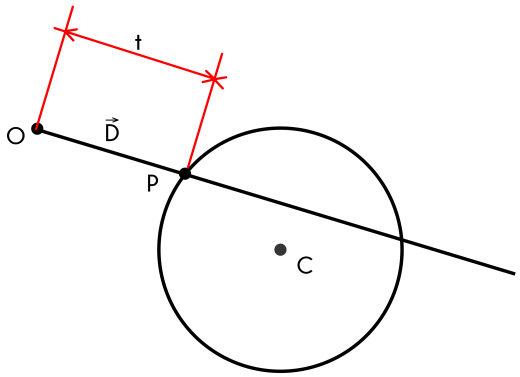

Figure 1: Ray-Sphere Intersection

The vector equation of the sphere in Figure 1 is given as -

$$(P - C) \cdot (P - C) - r^2 = 0$$

Substituting the ray equation for P -

$$(O + tD - C) \cdot (O + tD - C) - r^2 = 0$$

$$(D \cdot D)t^2 + 2D \cdot (O - C)t + (O - C) \cdot (O - C) - r^2 = 0$$

Solve for t using the quadratic formula where -

$$a = (D \cdot D)$$

$$b = 2D \cdot (O - C)$$

$$c = (O - C) \cdot (O - C) - r^2$$

$$t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

When the determinant is less than zero there is no intersection, else solve for t.

For ray-plane intersection, if the component of the ray along the plane axis is not zero we simply equation the vector equation of the ray with the plane's distance from origin -

$$O + tD = distanceFromOrigin$$

$$t = \frac{distanceFromOrigin - O}{D}$$

The final point of intersection from both is given by -

$$P = O + tD$$

4 Photon Tracing

A certain number of photons are shot into the scene starting from the point light source along a random direction given by a unit vector. The photon ray is then tested for intersection with the scene's planes and spheres. If there is an intersection depending on the material of the intersected surface the ray is either specularly reflected, refracted or diffusely reflected. The ray is bounced a certain number of times, and with each bounce power of the photon is reduced by the root of the number of bounces so far.

4.1 Reflection

For specular reflection the direction of the reflected ray is given as

$$d_r = d_i - 2(n \cdot d_i)N$$

where d_1 represents direction of the incident ray, n is the normal at the point of intersection obtained from above and d_r is the direction of the reflected ray.

From Figure 2,

$$d_I = A + B$$

$$d_R = A - B$$

$$B = cos(\theta)N$$

Replacing B we get,

$$d_I = A + cos(\theta)N$$

$$d_R = A - cos(\theta)N$$

From this,

$$A = d_I - cos(\theta)N$$

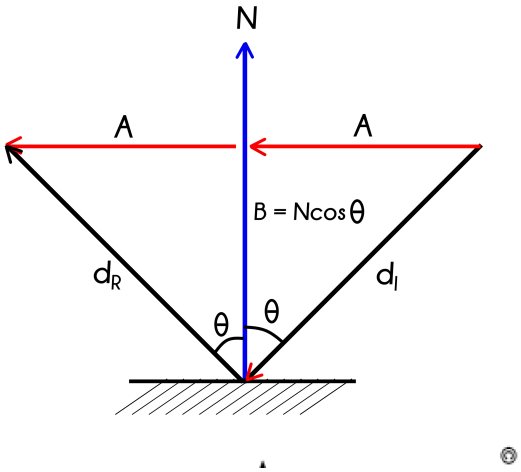

Figure 2: Reflection

d_R can be rewritten as

$$d_R = d_i - 2cos(\theta)N$$

or

$$d_R = d_I - 2(N \cdot d_I)N$$

4.2 Refraction

Refraction is given by Snell's law where

$$\frac{sin(\theta_1)}{sin(\theta_2)} = \frac{n_2}{n_1}$$

The direction of the refracted ray depends on two factors—the ray angle of incidence and the refractive index of the media.

$$cos(\theta_1) = d_I \cdot N$$

If $cos(\theta_1) < 0$ then the ray is traveling from outside (air) to inside, else from inside to outside. Depending on this the ratio of refractive indices $\eta = \frac{n_1}{n_2}$ or $\frac{n_2}{n_1}$

From Figure 3,

$$A = Msin(\theta_2) \text{ and} B = -Ncos(\theta_2)$$

The term $I + C$ gives a vector perpendicular to N or the tangent to the surface. To normalize it, we divide it by $sin(\theta_1)$. From this,

$$M = \frac{I + C}{sin(\theta_1)}$$

$$C = cos(\theta_1)N$$

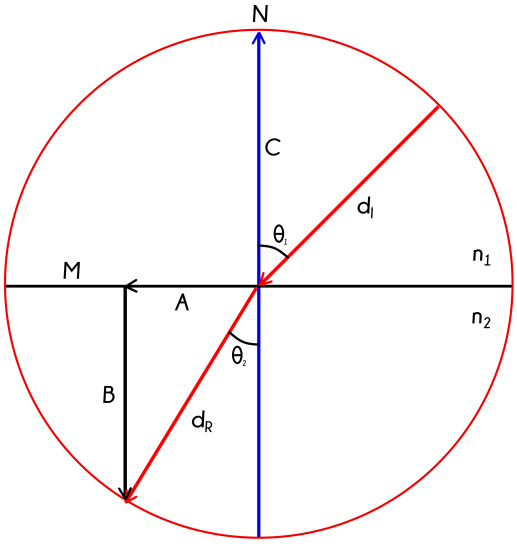

Figure 3: Refraction

$$\begin{aligned} d_R =& A + B \\ =& Msin(\theta_2) - Ncos(\theta_2) \\ =& \frac{sin(\theta_2)(I + C)}{sin(\theta_2)} - Ncos(\theta_2) \\ =& \frac{(I + cos(\theta_2)N)sin(\theta_2)}{sin(\theta_1)} - Ncos(\theta_2) \end{aligned}$$

From Snell's Law, $\frac{sin(\theta_1)}{sin(\theta_2)} = \frac{n_2}{n_1}$, so

$$d_R = \frac{n_1}{n_2}(I + cos(\theta_1)N) - Ncos(\theta_2)$$

Using the relationship between sin and cos and given that $sin(\theta_2) = \frac{n_1}{n_2}sin(\theta_1)$

We have

$$d_R = \frac{n_1}{n_2}(I + cos(\theta_2)N) - N\sqrt{1 - (\frac{n_1}{n_2})^2 sin(\theta_1)^2}$$

As mentioned earlier, depending on whether the ray is moving from outside to inside or inside to outside, we interchange n_1 and n_2.

4.3 Diffuse Reflection

For diffuse reflection, the incident ray is simply reflected in a random direction about the point of intersection. If the incident ray has suffered a previous refraction or reflection, the point is stored in the caustic photon map.

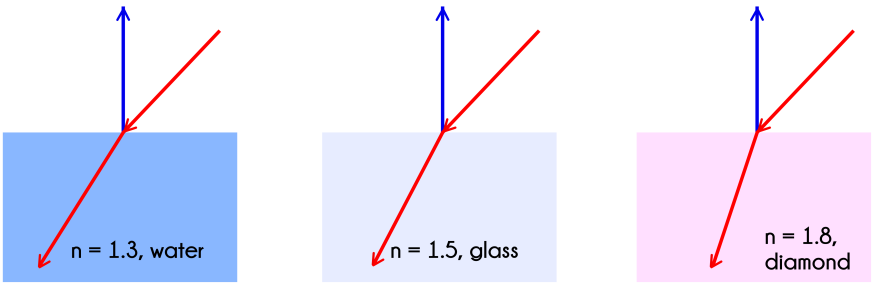

Figure 4: Different Refractive Indices

5 Photon Map Data Structure

The photon map is structured as a balanced kd-tree. The time it takes to locate one photon in a balanced kd-tree has a worst time performance of $O(logN)$, where N is the number of photons in the tree. The tree is represented using a heap data structure which means explicitly storing pointers to children is no longer needed as array element 1 is the tree root, and element i as left child and element $2i + 1$ as right child.

Balancing a kd-tree is similar to balancing a binary tree. The main difference is the choice at each node of a splitting dimension. When a splitting dimension of a set is selected, the median of the points in that dimension is chosen as the root node of the tree representing the set and the left and right subtrees are constructed from the two sets separated by the median point. The choice of a splitting dimension is based on the distribution of points within the set. The splitting dimension is thus chosen as the one which has the largest maximum distance between the points. The complexity of the balancing algorithm is $O(NlogN)$ where N is the number of photons in the photon map. [1].

6 Querying

During rendering, the photon map can be queried to get an estimate of irradiance about a point within a region of influence. A function firsts locates n closest photons about a point x with a distance d_2 and then sums up the irradiance of the returned photons.

```
locate_photons( p ) {
  if ( 2p+1 < number of photons ) {
    examine child nodes
                Compute distance to plane (just a subtract)
    δ = signed distance to splitting plane of node n
    if (δ < 0) {
                We are left of the plane - search left subtree first
      locate_photons( 2p )
      if ( δ² < d² )
        locate_photons( 2p+1 )        check right subtree
    } else {
                We are right of the plane - search right subtree first
      locate_photons( 2p+1 )
      if ( δ² < d² )
        locate_photons( 2p )          check left subtree
    }
  }
            Compute true squared distance to photon
  δ² = squared distance from photon p to x
  if ( δ² < d² ) {              Check if the photon is close enough?
    insert photon into max heap h
              Adjust maximum distance to prune the search
    d² = squared distance to photon in root node of h
  }
}
```

Figure 5: Locate Photons Function

7 Results

Figure 6 show visualizations of the photon with two spheres—one reflecting and one refracting with refractive index of 1.5—with change in light positions. 500,000 photons were initially shot into the scene. Figure 7 shows three spheres—one reflecting and two refracting. Figure 8 shows change in the caustic pattern on changing the refractive index from 1.5 to 1.8. The implementation was able to handle generation of upto 10,000,000 initial photons.

My attempt to integrate this with ray tracer did not work as expected. The code required to retrieve information is still accurate.

8 Challenges

The fun challenges in writing this implementation was deriving the equations of for object intersection and light interaction. One particular challenge was the actual design of code including the various classes. The implementation of the photon map data structure was tricky. My real struggle was with integrating the photon map implementation with a ray tracer. I think I incorrectly implemented the ray tracer itself which is why the integration did not work out. I did not try to use an off-the-shelf ray tracer because of the huge differences in code style, so I tried writing it from scratch. I'm still trying to figure out what went wrong. There are checks to avoid values of no color but there are still spots in the image.

9 Future Work

- Fix the integration with a ray tracer
- Adapt for any any arbitrary mesh, particle based fluid

References

[1] Jensen, Henrik Wann. "A practical guide to global illumination using ray tracing and photon mapping." ACM SIGGRAPH 2004 Course Notes. ACM, 2004.

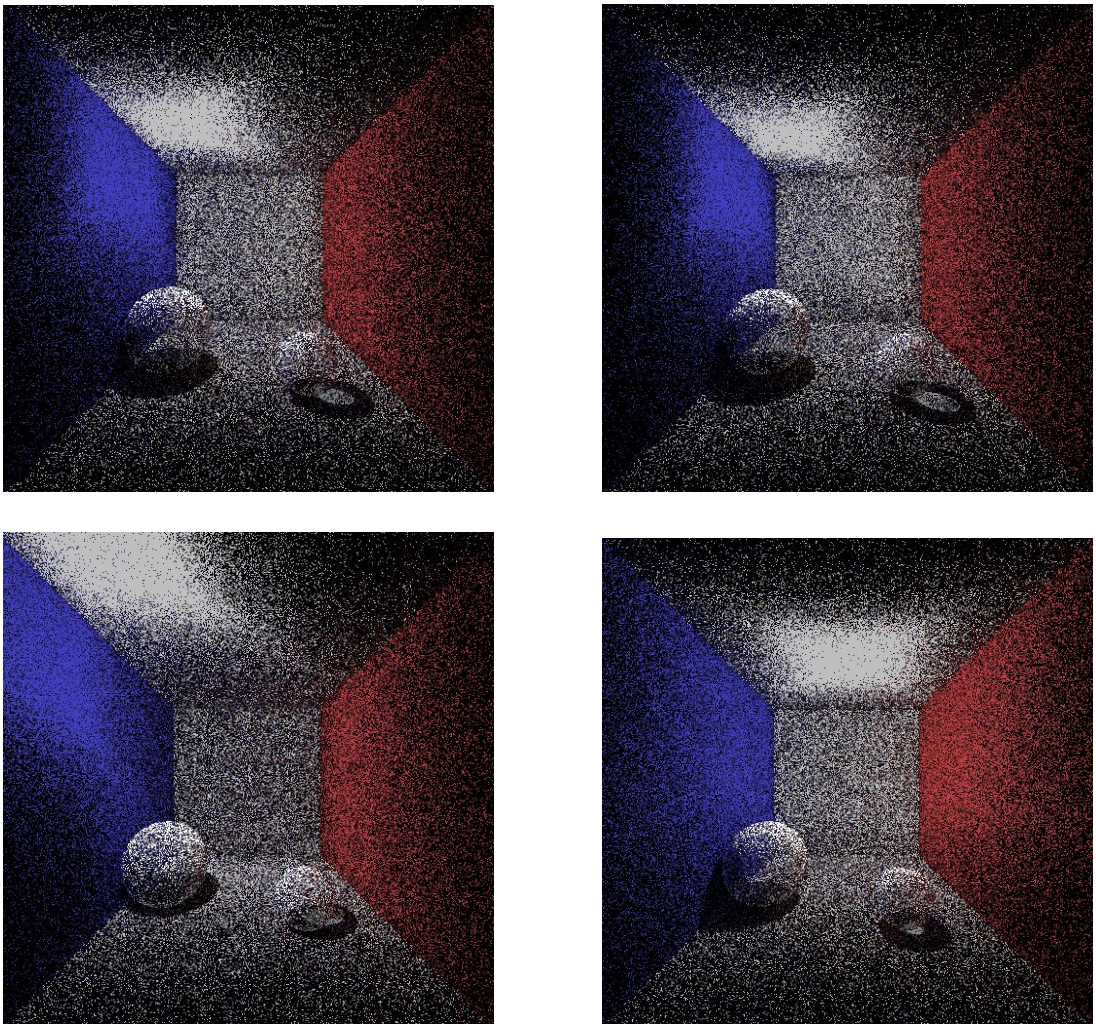

Figure 6: Caustic and Global Map Visualization With Two Spheres

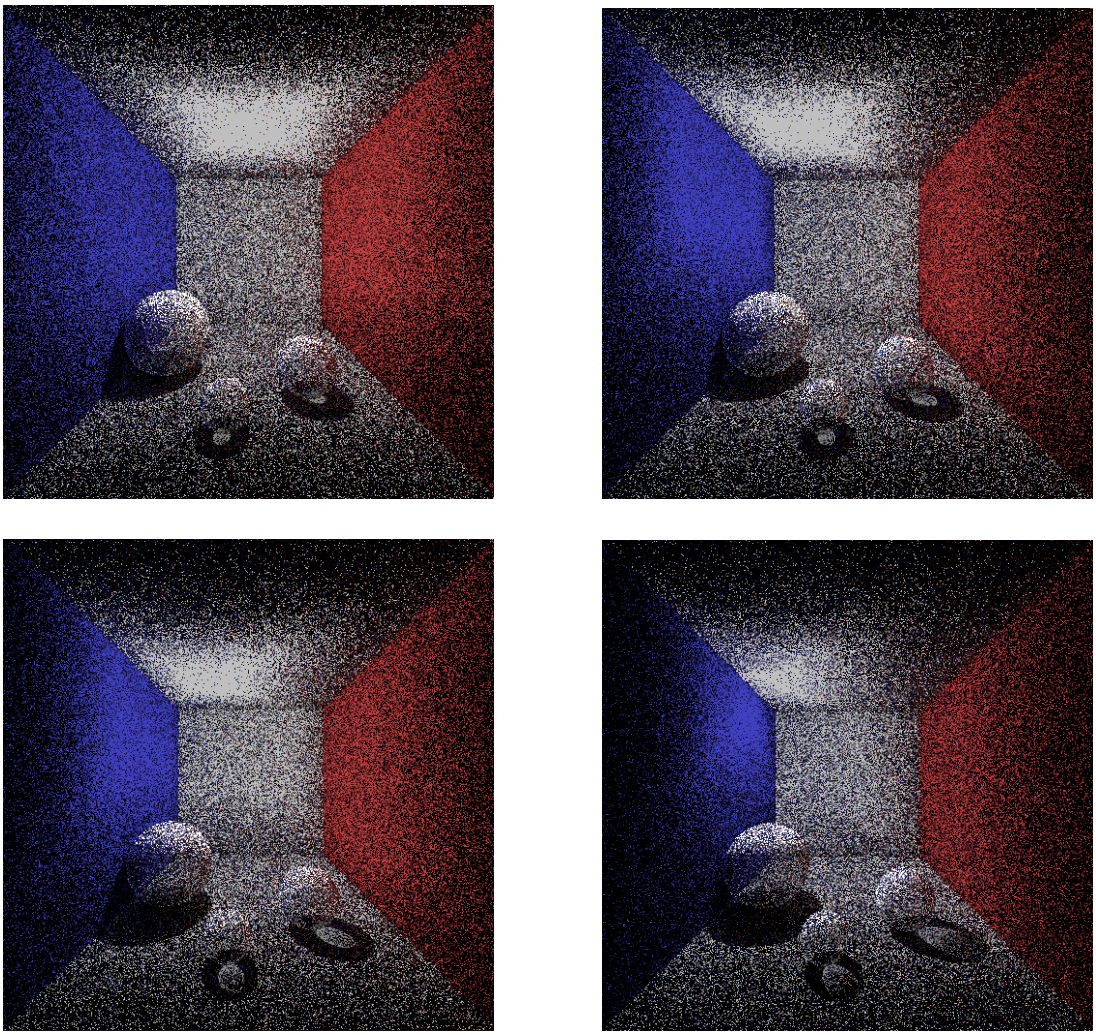

Figure 7: Caustic and Global Map Visualization With Three Spheres

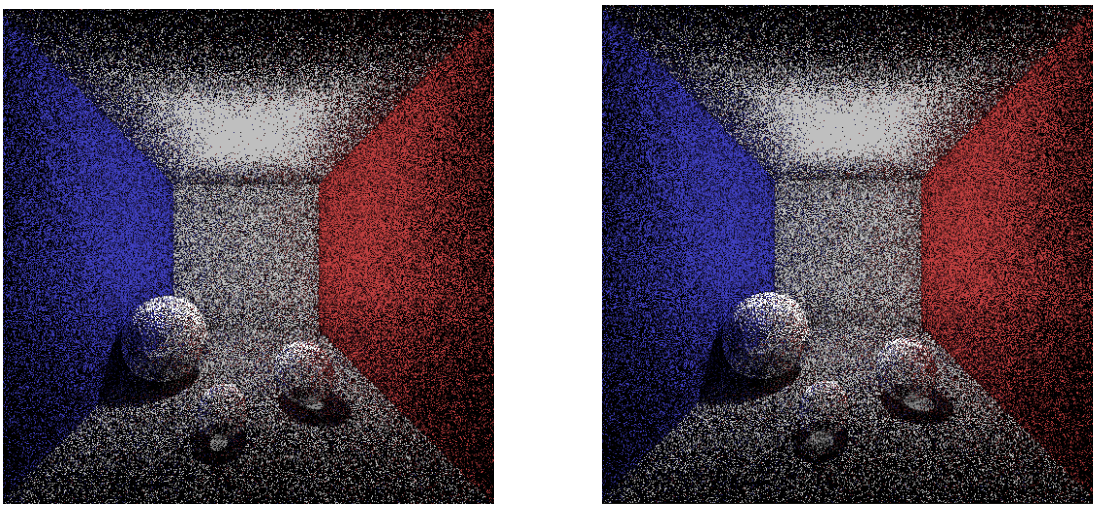

Figure 8: L: 1.5, R:1.8