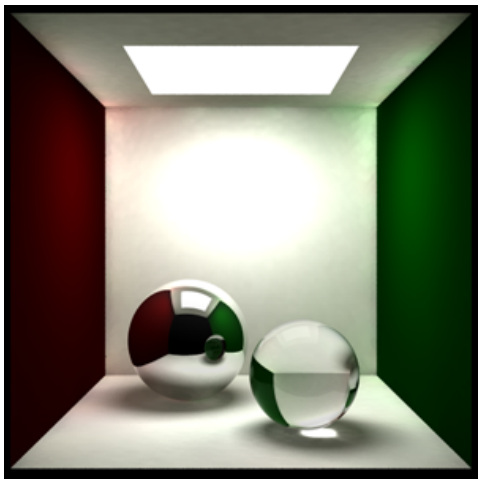
Photon Mapping

Zack Waters

Introduction

Global illumination is critical in generating synthetic images that are indistinguishable from images taken of the real world. Two major categories for GI algorithms are radiosity methods and point sampling techniques such as path tracing. Global illumination algorithms are designed to approximate the rendering equation which describes the complete transport of light within an environment.



Radiosity is a finite element method that computes a view independent GI solution as a finite mesh. The environment is divided up into patches that are uniform and perfectly diffuse and energy is transferred amongst these patches until equalized. This tight coupling of the lighting information to the geometry is costly to calculate and meshing artifacts can occur if not carefully constructed. Radiosity also doesn't handle arbitrary reflection models.

Path tracing is a probabilistic view dependent point sampling technique that extends raytracing to approximate all light paths such as those that contribute to indirect illumination and caustics. The scene is rendered by tracing a set of ray paths from the eye back to the lights, where each path does not branch. Path tracing is a random process where each path is a simple random walk and thus a Markov chain. Path tracing can be very computationally expensive as a large number of samples are needed to converge or variance will show up as noise in the final image.

Fortunately, Monte Carlo techniques have been around for a long time and there are several techniques that can be borrowed to reduce variance and achieve convergence in a finite number of steps. Most notable are importance sampling and Russian roulette. Other optimization strategies to further reduce computational costs include irradiance caching and irradiance gradients. Even with these optimizations path traced images are still plagued with high frequency noise generated by caustics.

Photon Mapping is a two pass global illumination algorithm developed by Henrik Jensen as an efficient alternative to pure Monte Carlo raytracing techniques. Photon mapping decouples the illumination solution from the geometry and the solution is represented in a spatial data structure called the photon map. This decoupling proves to be quite powerful as the rendering equation's terms can be calculated separately and stored into separate photon maps. It also is the reason that Photon mapping is very flexible as parts of the rendering equation can be solved using other techniques. Photon mapping has also been extended to account for participating media effects such as sub-surface scattering and volume caustics.

In this write-up I plan to give an over-view of the two passes of Photon Mapping. I will not touch upon participating media and the various optimizations relating to photon mapping and uses in other algorithms.

First Pass - Photon Tracing

Photon tracing is the process of emitting discrete photons from the light sources and tracing them through the scene. The primary goal of this pass is to populate the photon maps that are used in the rendering pass to calculate the reflected radiance at surfaces and out-scattered radiance in participating media.

Photon Emission

A photon's life begins at the light source. For each light source in the scene we create a set of photons and divide the overall power of the light source amongst them. Brighter lights emit more photons then dimmer lights. Finding the number of photons to create at each light depends largely on whether decent radiance estimates can be made during the rendering pass. For good radiance estimates the local density of the photons at surfaces need to provide a good statistic of the illumination.

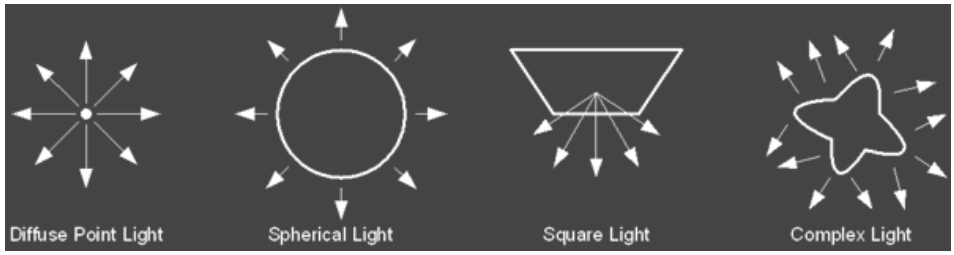


Figure recreated from [1]

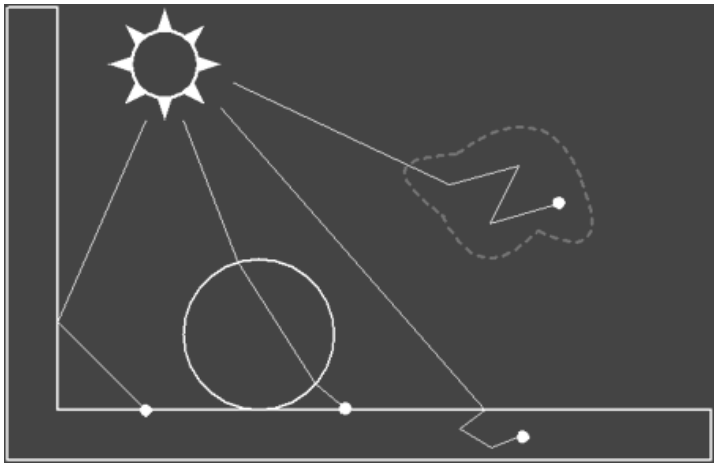
Jensen states that any type of light source can be used and describes several emission models. For point light sources we want to emit photons uniformly in all directions. For area light sources we pick a random position on the surface and then pick a random direction in the hemisphere above this position.

The emission strategy can be changed depending on the scene. For example, for sparse scenes we want to focus our photon emission at the geometry or most photons will be lost. Jensen's solution to this is to use projection maps. Projection maps optimize photon emission by directing photons towards important objects. Projection maps are typically implemented as bitmaps that are warped onto a bounding shape for the light source where each bit determines whether geometry of importance is in that direction.

Projection maps are also very important for effects such as caustics. Caustics are generated from focused light coming from specular surfaces and require a higher density of photons for an accurate radiance estimate. With projection maps we can focus more photons towards specular surfaces.

Photon Scattering

Emitted photons from light sources are scattered through a scene and are eventually absorbed or lost. When a photon hits a surface we can decide how much of its energy is absorbed, reflected and refracted based on the surface's material properties.



To account for the distribution of the photon's energy at the surface we can break it up into smaller photons as needed. However, it is easy to see that the generation of new photons at each surface interaction will be very costly in terms of computation and storage. Instead, Jensen advocates using a standard Monte Carlo technique called Russian Roulette. We use Russian Roulette to probabilistically decide whether photons are reflected, refracted or absorbed. Using this technique we reduce both the computational and storage costs while still obtaining the correct result.

To see why Russian Roulette works Jensen gives the following example. Given a probability, *p*, that another radiance estimate, *Ln*, is made we find that:

$$\xi \in [0,1] \text{ is a random variable}$$

$$p \in [0,1] \text{ is probability that another radiance estimate, } L_n, \text{ is made.}$$

$$L_n = \begin{cases} \xi < p \text{ then } \dfrac{L}{p} \\ otherwise \quad 0 \end{cases}$$

The expected value for the next radiance estimate is calculated as follows.

$$E\{L\} = \text{Prob(Termination)} * 0 + \text{Prob(Survival)} * \frac{E\{L\}}{p} = E\{L\}$$

L is computed by tracing another ray and is weighted by its probability. The probability of taking another radiance estimate is *p*, otherwise *(1–p)*.

$$E\{L\} = (1-p) * 0 + p * \frac{E\{L\}}{p} = E\{L\}$$

The probability events we are concerned with in Photon Mapping are whether photons are absorbed, reflected or refracted. If we only consider the probability events for reflection and absorption we have the following.

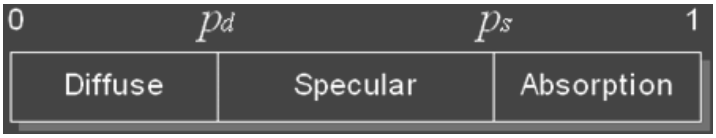
$\xi \in [0,1]$ is a random variable
$p \in [0,1]$ is probability of reflection
Φ_p is power of incoming photon

if $(\xi < p)$
then reflect photon at power Φ_p
else
photon is absorbed

It is important to note that the power of the reflected photon is not modified. Correctness of the overall result will converge with more samples. The probability is given by the reflectivity of a surface. If the reflectivity for a surface is 0.5 then 50% of the photons will be reflected at full power while the other 50% will be absorbed. To show why Russian Roulette reduces computational and storage costs let's consider shooting 1000 photons at a surface with reflectivity of 0.5. We could reflect 1000 photons at half power or we can reflect 500 at full power using Russian Roulette.

In typical environments there are more probability events we need to consider. For example, what is the probability for a photon to be refracted through a surface? Are we dealing with a diffuse or specular reflection or transmission? To handle these events we can divide our distribution domain according to the material properties of a surface.

Considering only diffuse and specular reflection and absorption we have the following domain



Again, we decide what to do with the photon by generating a random number within the domain. For more information see [1].

Photon Storing

For a given scene we may shoot millions of photons from the light sources. It is desirable that our photon map is compact to reduce the storage costs. We also want it to support fast three dimensional spatial searches as we will need to query the photon map millions of times during the rendering phase.

The data structure that Jensen recommends using for the photon map is a kd-tree. It is one of the few data structures that is ideal for handling non-uniform distributions of photons. The worst-case complexity of locating photons in a kd-tree is O(n) where as if it is balanced it is O(log n). After all photons are stored in the map we want to make sure it is balanced.

For each photon we store its position, power, and incident direction. Jensen proposes the following structure.

```
struct photon {
    float x, y, z;       // position ( 3 x 32 bit floats )
    char p[4];           // power(rgb) packed as 4 chars
```

```
    char phi, theta;     // compressed incident direction
    short flag;          // flag used for kd-tree
}
```

Jensen prescribes Ward's shared-exponent RGB format for packing the power into 4 bytes. The phi and theta are spherical coordinates that mapped to 65536 possible directions. The position can possibly be compressed further into a 24bit fixed point format.

However, for most serious implementations the structure will be as compressed as possible to allow for extremely large and complex scenes. For naïve renderers the structure can remain largely uncompressed for ease of use.

Second Pass - Rendering

Photon mapping is used to calculate the effects of indirect lighting and caustics. Too many photons would be needed in the photon map to accurately handle specular/glossy surfaces and direct lighting.

Approximating the Rendering Equation

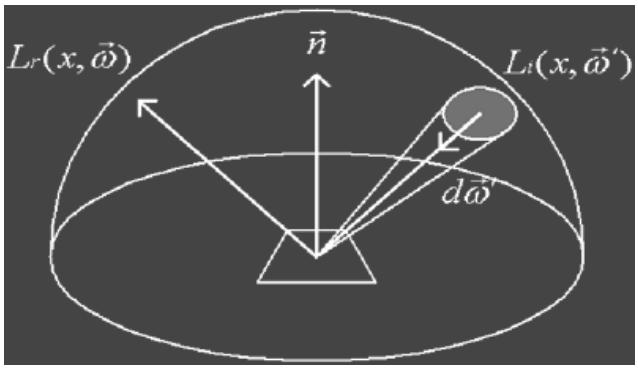
During the rendering pass we are using an approximation of the rendering equation to compute the reflected radiance at surface locations. The rendering equation in the form that is typically used is described as:

$$L_r(x,\vec{\omega}) = \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega}) L_i(x,\vec{\omega}') \cos\theta_i d\vec{\omega}'$$

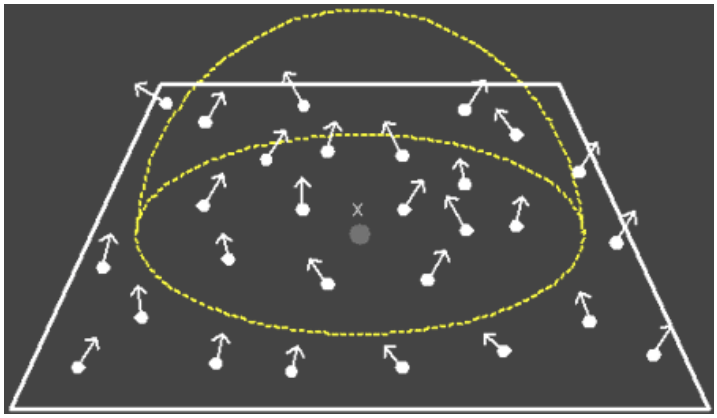
where:

x is surface location
$\vec{\omega}'$ is direction of incoming radiance
$\vec{\omega}$ is direction away from surface
f_r is the bidirection reflectance distribution function (BRDF)
L_i is incident radiance at x.
$\cos\theta_i$ is the projection of the area subtended by the solid angle to base of hemisphere
$d\vec{\omega}'$ is the solid angle
Ω is the hemisphere above point x

In the above equation we are integrating over the hemisphere above x the incoming radiance per solid angle as seen by x.



We can use density estimation to estimate the illumination by querying for the nearest N photons from the photon map.



This density estimate gives us the incoming flux or irradiance. If we ignore the wave phenomenon of light and time then irradiance basically tells us how much light is arriving at a particular area. In other words incident irradiance, Ei, is the sum of the number of photons, omega sub i , arriving per unit area.

$$E_i = \sum_{i=0}^{n} \frac{\Phi_i}{A}$$

With a differential solid angle, dw' , the differential irradiance, dEi , can be expressed in terms of incident radiance Li as:

$$dE_i(\vec{\omega}') = L_i(\vec{\omega}')\cos\theta_i d\vec{\omega}' \text{ or as } L_i(\vec{\omega}') = \frac{dE_i(\vec{\omega}')}{\cos\theta_i d\vec{\omega}'}$$

Using this relationship we can substitute for Li in the rendering equation:

$$L_r(x,\vec{\omega}) = \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega}) \frac{dE_i(\vec{\omega}')}{\cos\theta_i d\vec{\omega}'} \cos\theta_i d\vec{\omega}'$$

canceling out the terms we have:

$$L_r(x,\vec{\omega}) = \int_{\Omega} f_r(x,\vec{\omega}',\vec{\omega}) dE_i(\vec{\omega}')$$

Using information from our nearest neighbor query gives us a set of N photons for a given area A in which we know the incident direction for each. This gives us the following approximation:

$$L_r(x,\vec{\omega}) = \sum_{p=1}^{n} f_r(x,\vec{\omega}_p,\vec{\omega}) \frac{\Delta\Phi_p(x,\vec{\omega}_p)}{\pi r^2}$$

In order to get a good radiance estimate we need a sufficient number of photons in our density estimate. Of course this can directly relate to the number of photons that are emitted from the light sources. The more photons that are used the more accurate is our estimate. We also need to be careful how we collect photons.

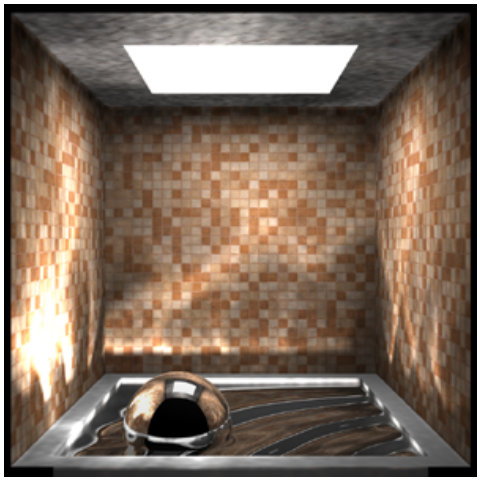
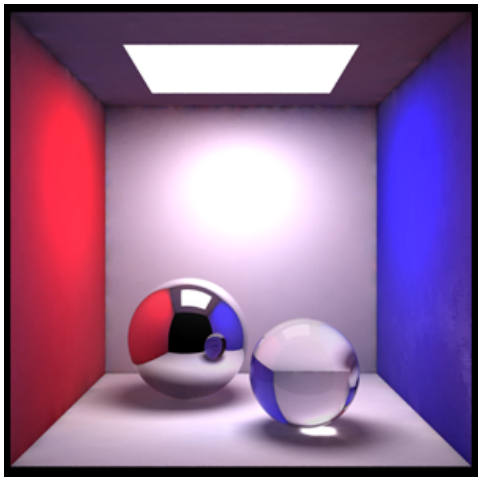
The main idea behind gathering photons is that we are hoping to get an idea of what the illumination at x is by examining the nearest N photons. If we include photons from other surfaces, with drastically different surface normals, or from volumes in our estimate then we can degrade the accuracy of our estimate.

Caustics

Rendering caustics is fairly straightforward as you can directly visualize the photon map. Visualizing the photon map directly basically means we gather N nearest photons and estimate the radiance directly from that information using the last equation

from the previous section. During the photon tracing phase a separate photon map is used for caustics. Otherwise many more photons would be needed in a single photon map to produce good caustics. For sharper caustics a filter can be used such as cone filter where more closer photons are weighted more then photons that are father from point x.

click on images to enlarge

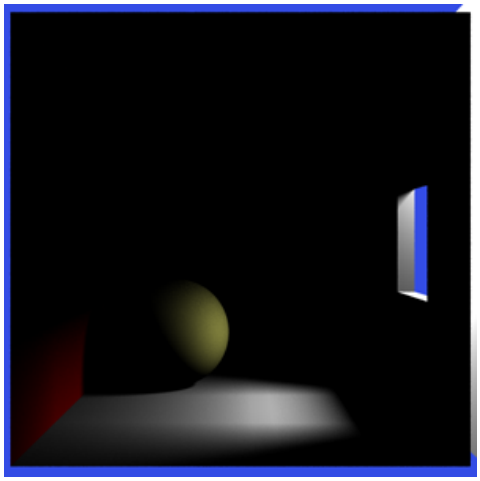
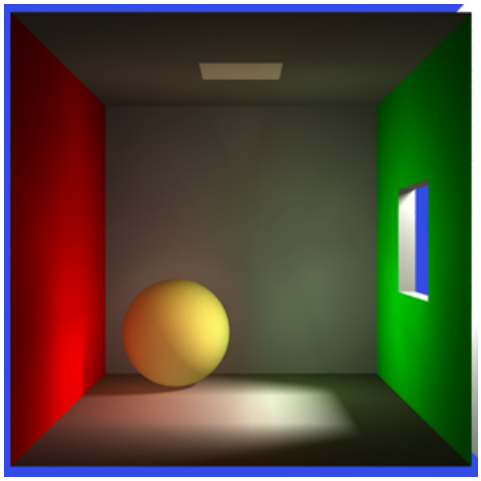



Indirect Illumination

Indirect illumination is much trickier to get right. In this case, the information from the photon map can be used in a more accurate solution such as path tracing. However, good results can be obtained using photon mapping directly. Often times a final gathering step is employed to improve the results using the photon map. During the final gather step we can sample the hemisphere above the surface at x for incoming radiance. For each sample we can compute a radiance estimate directly from the photon map. The first image in this write-up uses this technique as well as the image on the right in the previous section.

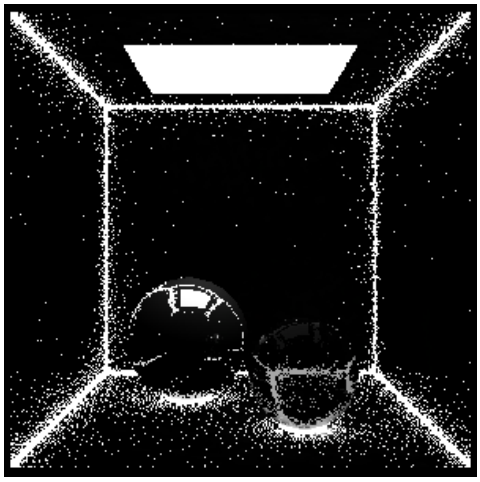
However, even without a final gather step it is possible to achieve images such as the one below. The image on the left shows the contribution from direct lighting only and the image on the right was rendered using photon mapping. The image on the right does not use a final gather step. Only 26,000 photons actually made it into the window and 500 were used in the radiance estimate. Notice the color bleeding on the yellow ball and on the floor.

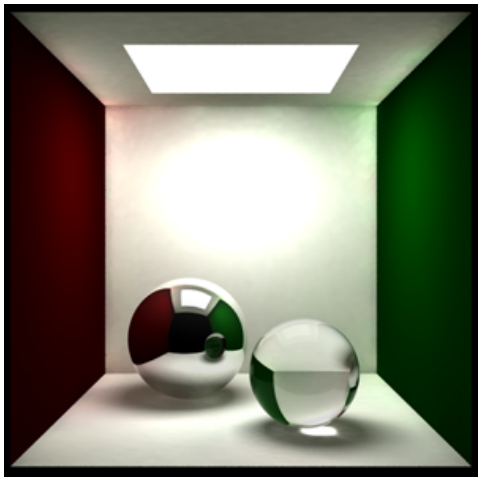
click on images to enlarge

Even using the photon map directly images can take a long time to render. Using final gathering in the rendering step can cause the render time to increase dramatically. Ward[6] developed a method called irradiance caching where the illumination is cached and re-used. Illumination values from the cache can be interpolated at a point. The image on the left shows a visualization of the irradiance cache. All points that are black are the points where a cached value was used in the right image.

click on images to enlarge





Flexibility

The rendering equation can be split up into four separate terms describing direct illumination, specular reflection, caustics, and indirect illumination.

$$L_r = L_{direct} + L_{specular} + L_{caustics} + L_{indirect}$$

Due to the fact that information from the photon map describes the incoming flux and that this information is decoupled from the geometry we have the flexibility to solve different parts of the rendering equation by using multiple photon maps. This is also another reason why photon mapping can easily be incorporated into existing renderers.

For example, we may have an existing path tracer where we would like to use photon mapping to handle caustics. Path tracing does a great job at computing the direct and indirect lighting while photon mapping handles caustics which removes the high frequency noise typically seen in path traced images. In addition, we can use information from the photon map to help us importance sample the hemisphere.

Another great feature of the photon maps is that it provides information that can be used as heuristics for other parts of the rendering equation. For example, with a little extra work, information from the photon map can provide information as to whether shadow rays are needed at a location. This alone can reduce the rendering times significantly. In path tracers the photon map can also be used for importance sampling.

References

[1] Jensen, Henrik W., Realistic Image Synthesis Using Photon Mapping, A K Peters, Ltd., Massachusetts, 2001

[2] Shirley, Peter Realistic Raytracing A K Peters, Ltd. Massachusetts, 2000.

[3] James T. Kajiya. The Rendering Equation. Computer Graphics, 20(4):143-150, August 1986. ACM Siggraph ’86 Conference Proceedings.

[4] Shirley, Peter Fundamentals of Computer Graphics, A K Peters, Ltd, Massachusetts, 2002

[5] Glassner, Andrew Principles of Digital Image Synthesis Morgan Kaufmann Publishers, Inc., San Fransico, 1995

[6] Ward et al, A Ray Tracing Solution for Diffuse Interreflection.Computer Graphics, 22(4): 85-92 August 1988. ACM Siggraph '88 Conference Proceedings.
