Recent Advances in Monte Carlo Offline Rendering

Lecture #6: Monday, 21 September 2009
Lecturer: Ravi Ramamoorthi
Scribe: Nicholas Kong

1 Introduction

The research problems in Monte Carlo rendering have changed over the years. At first, the question was one of efficiency: how can we render an image quickly? Recent research has focused on rendering more complex effects, such as soft shadows and motion blur. There has also been research into the Monte Carlo technique itself; while it is a well-known numerical technique, many subtleties have been discovered that are specific to its graphics application. This lecture covers some of the recent advances in Monte Carlo offline rendering and current problems researchers are investigating.

2 Improvements in Efficiency

Much of the rendering research focus in the 90s centered on improvements in efficiency. The techniques we will briefly cover include

1. Irradiance caching, which takes advantage of spatial coherence,
2. Improved sampling techniques to reduce variance, and
3. Photon mapping (good for complex effects like caustics).

In the past ten years, no big advances have been made in this area. The techniques listed above are all well-known and commonly used. However, that is not to say that all efficiency problems have been solved; unlike in theoretical computer science, we typically do not have proofs of the optimality of an algorithm in graphics, so it is very possible that further strides could be made in improving efficiency.

2.1 Irradiance Caching

Indirect lighting low-frequency and smooth, compared with direct lighting (see Figure 1). Direct lighting has much clearer definition. Irradiance caching is a technique for exploiting the low-frequency of diffuse interreflection (i.e., indirect lighting). It works by sampling irradiance at a few points on the surface of an object, then computing other irradiance values by interpolating from the samples.

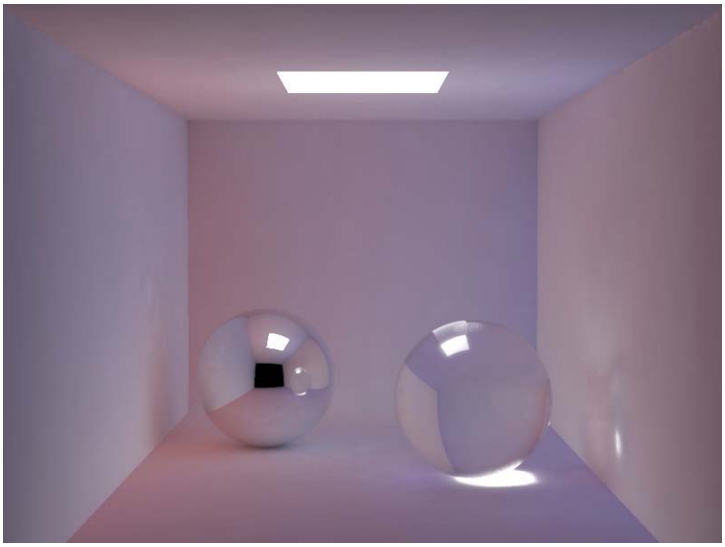

Figure 1: An example of a scene shaded by *indirect lighting* only. Note the low frequency of the diffuse interreflections.

2.1.1 Irradiance calculation

To compute irradiance, we must integrate radiance over a hemisphere (i.e., all possible incident light directions):

$$E(\mathbf{x}) = \int L_i(\mathbf{x}, \omega) \cos\theta d\omega$$

where $\mathbf{x} = (\rho, \theta, \phi)$ and $E(\mathbf{x})$ is the irradiance. When we hit a new point on the surface, we must decide whether to take another sample or to interpolate based on existing samples. Ward et al. [8] give a means of computing the estimated error of an interpolated value, which is used to determine whether a new sample is needed:

$$\epsilon(\mathbf{x}) \leq \left| \frac{\partial E}{\partial \mathbf{x}}(\mathbf{x} - \mathbf{x_o}) + \frac{\partial E}{\partial \theta}(\theta - \theta_o) \right|$$

Details are given in the paper. If adjacent samples exist that are within the error bound $\epsilon(\mathbf{x})$, the interpolated illuminance is given by:

$$E(\mathbf{x}) = \frac{\sum_i w(\mathbf{x_i}) E(\mathbf{x_i})}{\sum_i w(\mathbf{x_i})}, \; w(\mathbf{x}) = \frac{1}{\epsilon(\mathbf{x})}$$

To efficiently store the irradiance samples, Ward et al. recommend an octree, but other structures such as kd-trees would also work.

In summary, the irradiance caching algorithm is as follows:

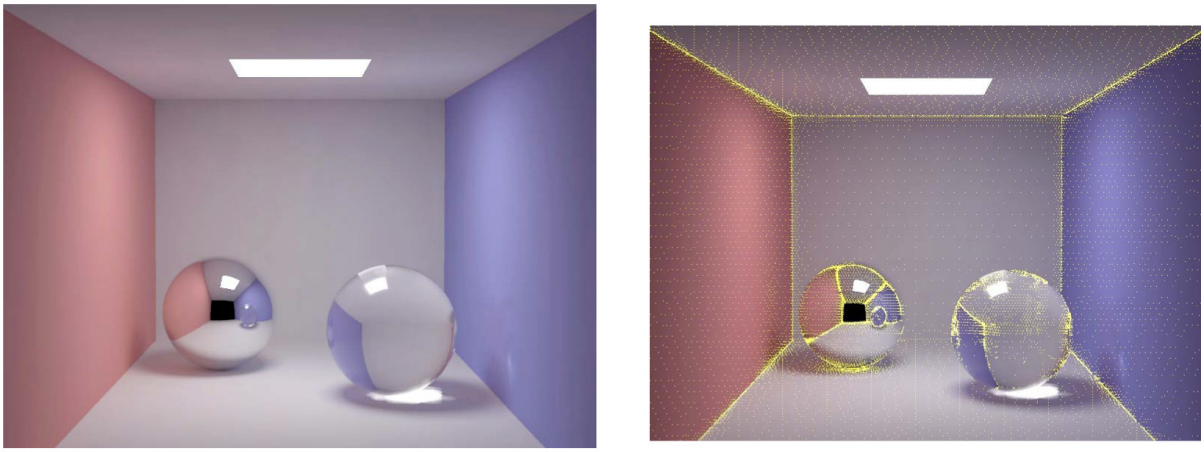

Figure 2: (a) The final rendered image with irradiance caching. (b) The irradiance sample points. Note how they mostly fall on the high frequency areas of the image (the edges) with sparser samples in the low frequency areas (the walls).

```
Find all samples with w(x) > q
if(samples found)
   interpolate
else
   compute new irradiance sample
```

Figure 2(a) shows a Cornell Box rendered using irradiance caching and Figure 2(b) shows the location of the irradiance samples.

2.2 Smarter Sampling Techniques

There are strategies to efficiently perform Monte Carlo sampling: techniques include stratified sampling, importance sampling, bi-directional ray tracing, multiple importance sampling, and Metropolis light transport. We will briefly discuss a few of these strategies here; for more information see Eric Veach's thesis.

2.2.1 Stratified Sampling

The central idea here is to break pixels into strata. Instead of placing samples on the strata, we jitter them slightly (see Figure 3). We estimate the value of each stratum separately, then interpolate over the entire image. This technique gives us an advantage if the variance within most of the strata is small. [6]

Figure 3 shows an edge that divides the light source into visible and non-visible portions. Note that only four of the sixteen blocks have some variance in it (e.g., between

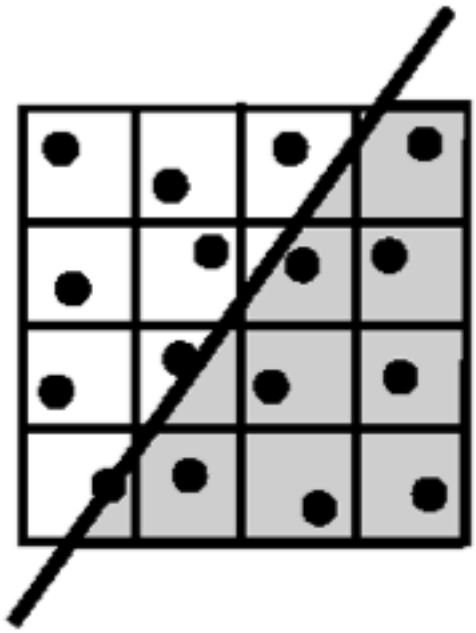

Figure 3: Example of a good use case for stratified sampling. An edge passes through the stratified pixel, but note that only four of the sixteen strata have non-zero variance.

visible and non-visible). Therefore, the variance $V[F_i] = 0$ in all but the four blocks which the edge runs through. If we didn't jitter the samples, then each pixel is either visible or non-visible resulting in banding artifacts. With jitter a noise pattern results, which eliminates such artifacts.

2.2.2 Space-time Patterns

Rendering a motion-blurred scene involves sampling through both time and space. We must therefore ensure that we jitter samples through both space and time. We want a decoherence between space and time: samples that are close in space should be far apart in time, and vice versa.

Figure 4 shows the Cook Pattern, one example of a sampling pattern that decorrelates time and space. The numbers correspond to time steps. [5]

2.2.3 Summary

The sampling method can significantly impact both how much noise is present in your image and the efficiency of the rendering.

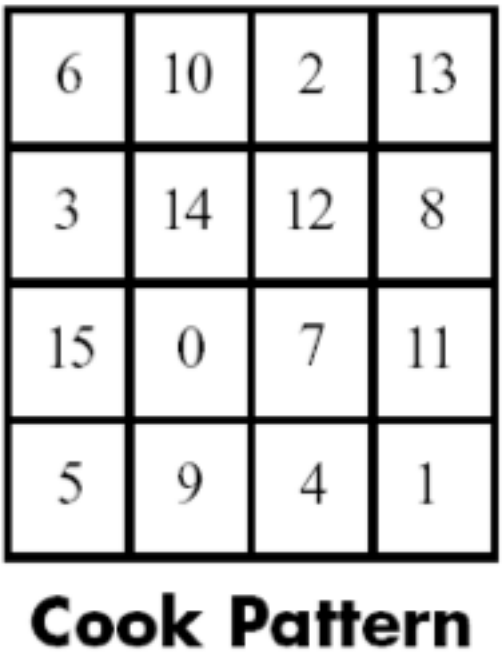

Figure 4: Cook Pattern for spacetime sampling.

3 Bidirectional Path Tracing

The idea behind bidirectional path tracing is to trace rays both from the eye and from the light. By tracing rays from both directions, there is a better chance of sampling all possible paths [7]. We first need to introduce an algorithm for light ray tracing.

3.1 Light Ray Tracing

Also known as *backwards ray tracing* [2], this is a technique where one traces rays from the light to the surface instead of from the eye to the surface. The algorithm can be described as follows:

```
1. Choose a light ray from the light
2. Find ray-surface intersection
3. Reflect or transmit
    u = Uniform()
    if u < reflectance(x)
        Choose new direction d drawn from BRDF(O|l)
        Goto 2
    else if u < reflectance(x) + transmittance(x)
        Choose new direction d drawn from BTDF(O|l)
        Goto 2
4. Deposit energy on the surface
```

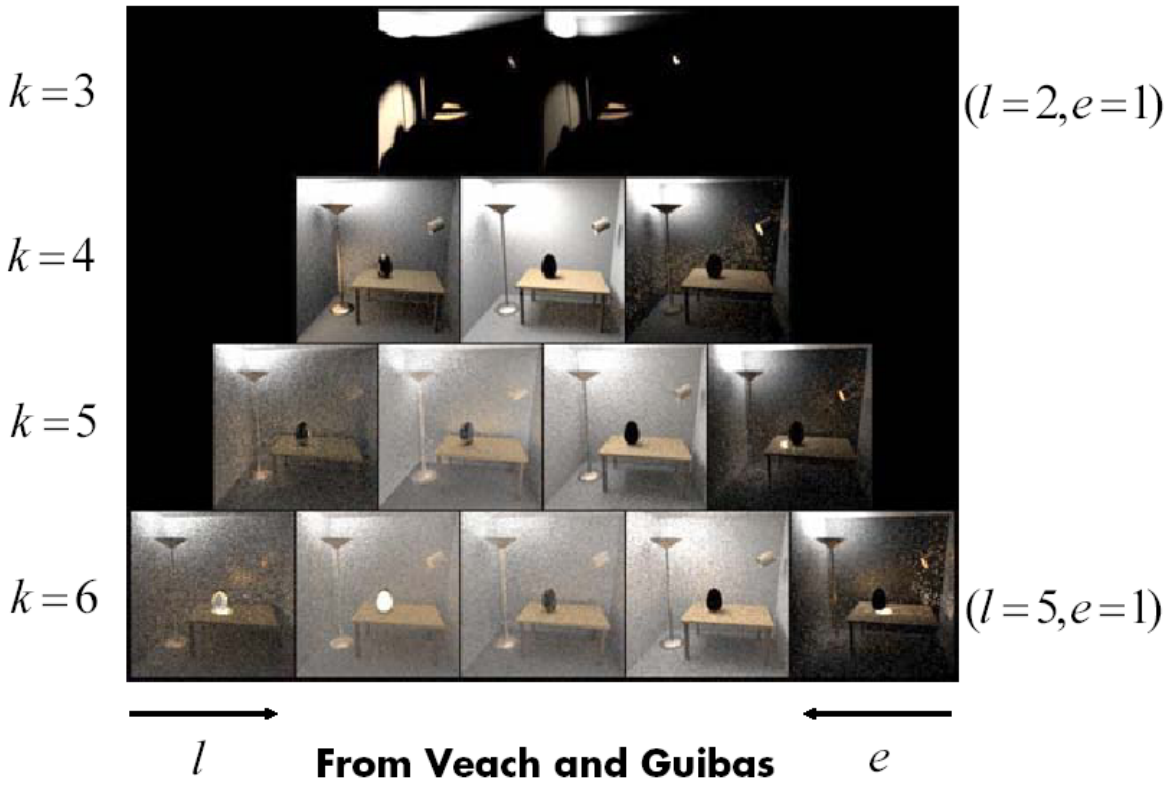

Figure 5: Bidirectional path tracing results. l is the number of bounces from the light source. e is the number of bounces from the eye.

3.2 Results

Figure 5 shows some results of bidirectional path tracing. k is the number of total bounces of light. l, increasing from left to right, is the number of bounces from the light source. e, increasing from right to left, is the number of bounces from the eye. The final scene will be some sum of all of these images.

4 Photon Mapping

Photon mapping is a technique similar to bidirectional path tracing, as it combines tracing from the eye and light sources, but computes a photon map once for all eye rays. The idea is to store "photons" in a scene in a kd-tree (the "energy" in the above bidirectional path tracing algorithm) and look up the values as necessary. Photon maps are especially useful for rendering caustics; standard path tracing may miss the light source altogether. [3]

It should be noted that irradiance caching and photon mapping are orthogonal methods. Irradiance caching generally refers to diffuse inter-reflection caching, whereas photon

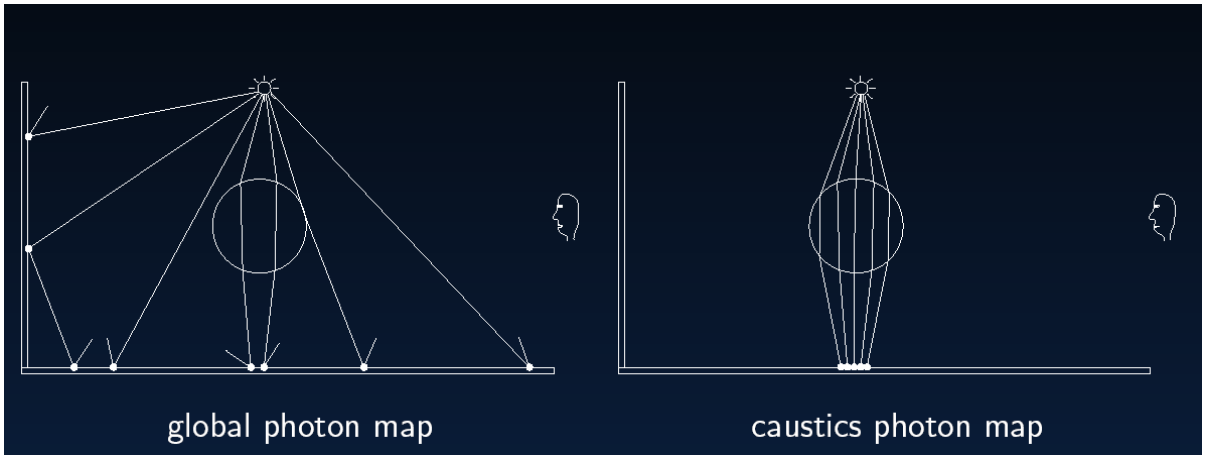

Figure 6: Computation of photon map.

maps are mostly used for caustics (although they too can be used for global illumination). Also, in irradiance caching one does not trace rays from the light source, but in photon mapping one does.

4.1 Rendering steps

In the first phase, we compute the photon map by shooting rays from the light source. Figure 6 shows both the global illumination and caustic photon maps. We then compute direct illumination and specular reflections via standard Whitted ray tracing. Finally, we use the precomputed caustics and global illumination photon maps to add caustics and indirect illumination respectively. Figure 8 shows a Cornell box rendering and associated global photon map.

The process of computing indirect illumination from the global photon map is known as "final gathering" and is a very expensive step. Figure 7 illustrates the process. Using irradiance caching on the global illumination photon map can speed up the process; one can then interpolate samples instead of gathering all photons for all rays shot from the eye.

Standard path tracing will also render a correct image, but takes a long time to run. Irradiance caching and photon mapping introduce bias into the solutions, however, but are correct in the limit; additional, you can further optimize by taking advantage of sampling tricks.

We will now move on to work on image-based appearance techniques, in particular the importance sampling of environment maps and BRDFs.

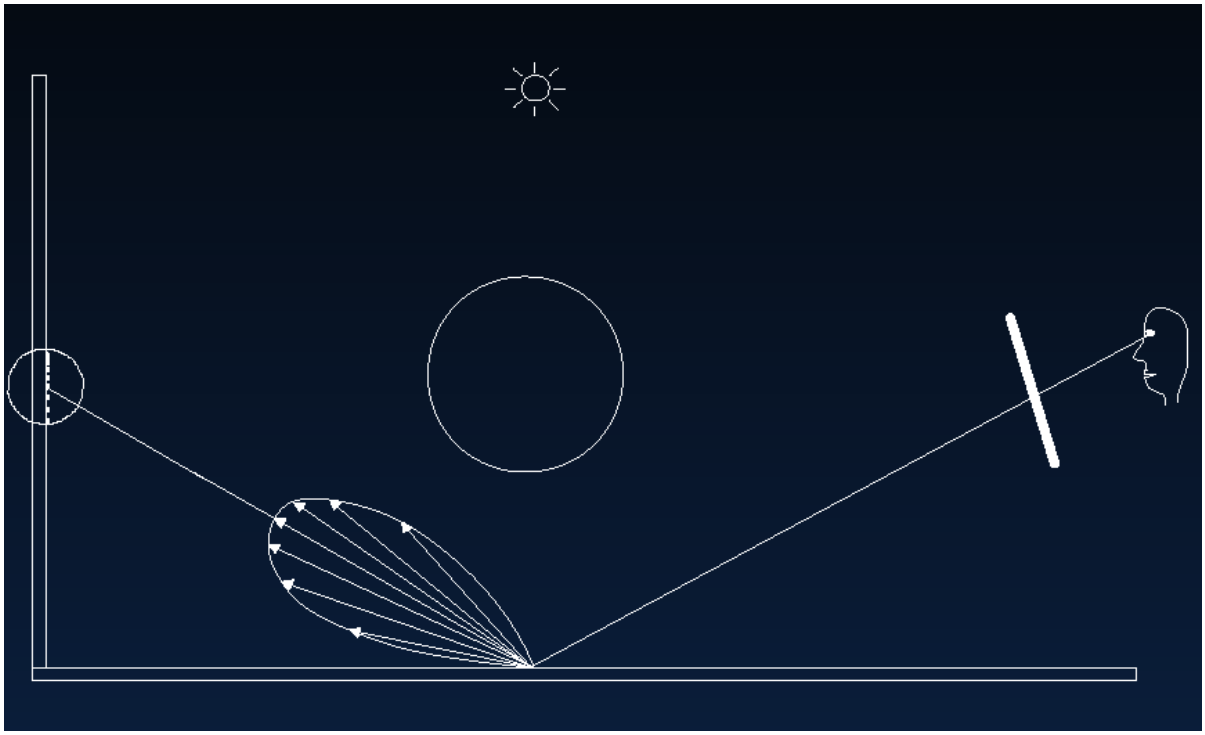

Figure 7: The final gathering step for indirect illumination.

Figure 8: The image on the left is a Cornell Box rendering using 200,000 global photons and 50,000 caustic photons. The image on the right is the photon map.

5 Image-Based Appearance

In the mid-1990s, interest began to build in appearance acquired from the real world, such as image-based lighting and measured BRDFs. However, there was little work in how to render directly from these acquired lighting environments and material properties.

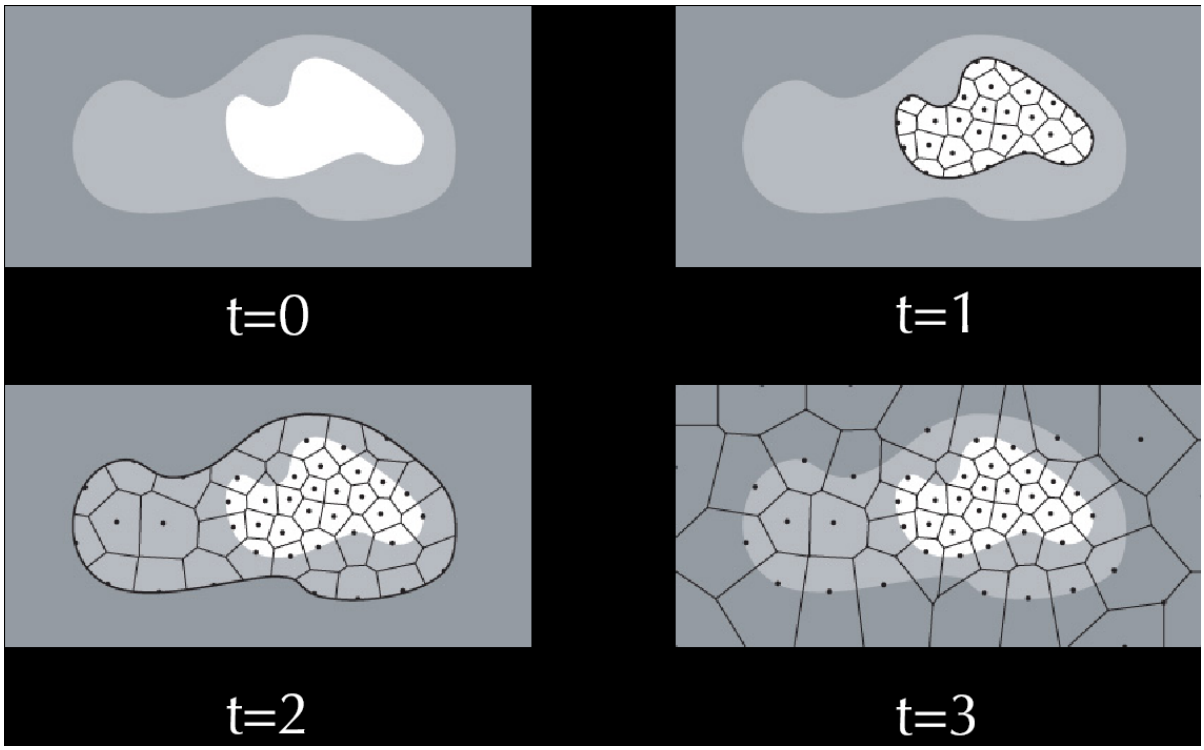

Figure 9: Steps in hierarchical stratification [1].

Rendering directly from an environment map, for example, is computationally infeasible; it is effectively a collection of a million light sources (one per pixel). Some sampling of the lighting data is necessary; the question, then, is how to properly importance sample lighting or BRDFs.

5.1 Structured Importance Sampling

The goal of structured importance sampling is to reduce an environment map to a set of point lights. These point lights can then be plugged into any renderer. The technique that was used is "hierarchical stratification". [1]

5.2 Hierarchical Stratification

Figure 9 shows four steps in the stratification. At t=0, the environment map is thresholded and hierarchically stratified. In the subsequent steps, each stratum is sampled uniformly. Note the Voronoi regions of the samples are small in the brightest samples and become larger as the intensity of the regions decrease. The strata could be used in a pure Monte Carlo rendering, or the center of each stratum could be approximated as a point light source (which may result in some banding). The algorithm is not real-time

and takes on the order of minutes, but it only needs to be run once per environment map. This makes transformations of the environment lighting (such as rotation) simple.

There have since been a number of follow-up papers that make the algorithm faster and make the frequency properties nicer.

5.3 BRDF Sampling

As of 2004, there were no good importance sampling schemes for BRDFs, despite the existence of a large number of complex BRDF models, both analytic and measured. Lawrence et al. [4] factored a BRDF into data-driven terms, each of which could be importance sampled.

5.3.1 Key Idea

The key idea was to project a BRDF (a 4D quantity) into a sum of products of 2D functions that depend on ω_o and 2D products that depend on ω_i:

$$f_r(\omega_o, \omega_i)(n \cdot \omega_i) = \sum_{j=1}^{J} F_j(\omega_o) G_j(\omega_p)$$

where ω_p depends only on the incoming direction and some re-parameterization of the hemisphere. This formula approaches the right result in the limit, but even a small number of factors gives good reconstruction.

When sampling the BRDF, only $G(\omega_i)$ is sampled; the F terms are pre-evaluated, as they are dependent only on exiting directions.

6 Summary

Monte Carlo rendering has gone through many iterations. Initially, path tracing was very slow, but starting in the 90s a number of techniques were proposed that made it more efficient. A little bit of a reboot was required to consider image-based rendering and image-based reflectance. Finally, researchers are working on computing high dimensional integrals to compute motion blur, soft shadows, etc.

References

[1] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. *ACM Trans. Graph.*, 22(3):605–612, 2003.

[2] James Arvo. Backward ray tracing. In *In ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, pages 259–263, 1986.

[3] Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, London, UK, 1996. Springer-Verlag.

[4] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient brdf importance sampling using a factored representation. *ACM Trans. Graph.*, 23(3):496–505, 2004.

[5] Don P. Mitchell. Spectrally optimal sampling for distribution ray tracing. In *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, pages 157–164, New York, NY, USA, 1991. ACM.

[6] Don P. Mitchell. Consequences of stratified sampling in graphics. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 277–280, New York, NY, USA, 1996. ACM.

[7] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, December 1997.

[8] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, pages 85–92, New York, NY, USA, 1988. ACM.