Monte Carlo Path Tracing

Lecture #5: Wednesday, 16 September 2009
Lecturer: Ravi Ramamoorthi
Scribe: Mason Smith

1 Motivation

The rendering equation, given in the previous lectures, provides a theoretical solution for computing the radiance at any point in the scene. However, direct analytic computation of the integral is intractable for anything besides the most trivial of scenes. Monte Carlo path tracing is a practical general solution to the rendering problem, which tries to enumerate all light paths in the scene.

As we shall see later, in a standard practical implementation, we perform direct and indirect sampling separately. Last class, we talked about breaking up the rendering equation integral by object. We can further decompose the integral based on object types (indirectly illuminated, light source, etc.)

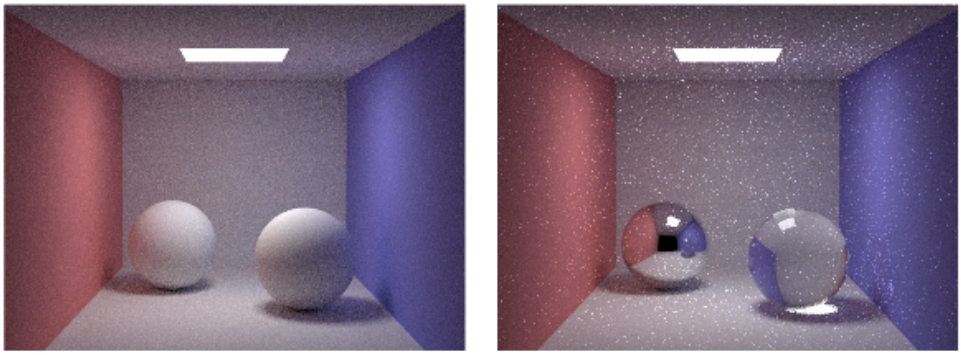

Figure 1: In both of these path tracing examples, 10 paths were used per pixel. Notice how the second example is much more noisy, due to the specular brdfs used for the spheres.

1.1 Advantages and Disadvantages

Many of the advantages of Monte Carlo path tracing (MCPT) come from its generality. MCPT is capaable of rendering any type of geometry, with any BRDF.

Figure 2: Examples from the arnold renderer

Furthermore, MCPT takes into account all types of light paths. We can enumerate light paths using regular expression notation. If we let S and D represent specular and diffuse surfaces and we let L and E represent the light and the camera eye repsectively, then we note that MCPT can handle all light paths $L(S|D)^*E$. By contrast, Whitted ray tracing cannot handle diffuse-diffuse reflections of the form $LDDD*E$, and radiosity does not handle specular reflections, e.g. of the form $LDSS*E$.

MCPT has other advantages. Basic MCPT is an unbiased algorithm. Recall that $T(X)$ is an unbiased estimator of X if $\mathbb{E}[T(X)] = X$. Thus, error in the image appears as noise do the the variance of the sample estimates, rather than less visually-appealing structured artifacts.

The main disadvantage to MCPT is its speed. The final result converges only in $\Theta(n)$, where n is the number of samples used in the approxmation. We can use various approaches to try to decrease the variance in the image or increase the speed of the algorithm, but these extensions often sacrificed the unbiased nature of the algorithm.

2 Monte Carlo Path Tracing

In MCPT, we integrate the radiance for each pixel by randomly sampling paths and computing the radiance along those paths. We use the samples as an estimate for the radiance.

2.1 Simple Implementation

Here we present a (very) naïve implementation of MCPT.

for all pixels p **do**
  Cast n samples within pixel square and average
  $pixel\_color += \frac{1}{n} \times TracePath(p, d)$
end for
$TracePath(p, d)$:
BEGIN
Find nearest intersection p'
if Emitted **then**
  { The extra factor of 2 comes from the probability $\frac{1}{2}$ reweighting. }
  return $2 * L_{emit}$
else if Reflected **then**
  Generate ray in random direction d'
  return $2 \times f_r(d \to d') \times (n \cdot d') \times TracePath(p', d')$
end if
END

This implementation is naïve in a number of ways. First, this method essentially doesn't support point-light sources. Since the probability of choosing a random direction point directly at a point light is essentially 0, we would never record any contribution from these lights.

Furthermore, this method wastes a large number of rays. For instance, for non-emissive objects in the scene, we waste 50% of the the rays by returning no luminance.

We want to improve efficiency, ideally without introducing bias. For this, we rely on some of the variance reduction techinques discussed in previous lectures.

3 Variance Reduction

3.1 Importance Sampling

In many scenes, certain areas contribute more to the scene than others. For instance, direct lighting on an object typically contributes much more to the observed radiance than indirect lighting, whenever direct lighting is present.

Similarly, output angles with a higher BRDF, given the incoming angles, will contribute more to the radiance at a point on a surface. As an extreme example, consider a perfectly specular mirror. For a given incoming direction, all of the radiance is contributed by exactly one outgoing direction. Even taking a thousand samples will likely yield no contribution, compared to just one sample in the ideal direction.

Importance sampling uses this phenomenon to our advantage. With importance sampling, we bias our sample paths towards those where we expect a larger contribution. This can occur in two ways, as outlined above:

On a "macro" level, we can change our bias towards reflected vs. emitted contributions, or we can cast more rays toward light sources. On a "micro" or "local" level, we use importance sampling on the BRDF to choose ray directions.

In order to keep our estimate unbiased, we need to weight our estimates by the inverse of the probability that they were picked. In other words, samples from low-probability areas are given a higher weight becuase they are more representative than the many samples from high-probability areas. This is no different from the original algorithm, where we used the factor fo 2 to compensate for the 50% probability of choosing either path.

3.1.1 "Macro"-level Importance Sampling'

Let's apply importance sampling to the previously outlined algorithm:

$TracePath(p, d)$:
BEGIN
Find nearest intersection p'
if $L_e = (0, 0, 0)$ **then**
  $p_{emit} := 0$
else
  $p_{emit} := 0.9$
end if
if $rand() < p_{emit}$ **then**
  return $\frac{1}{p_{emit}} \times L_{emit}$
else
  Generate ray in random direction d'
  return $\frac{1}{1-p_{emit}} \times f_r(d \to d') \times (n \cdot d') \times TracePath(p', d')$
end if
END

Aside: Do we want to sample regions with high values or high variance? In practice, finding high-value regions is easier.

3.1.2 'Micro"-level Importance Sampling

To reflect the importance of direct lighiting, we can try the following variation:

```
Reflected Ray variation:
  Pick a light source
  Trace a ray toward the light
  Trace a ray not hitting the light (via rejection sampling)
  1/(solid angle) for ray to light source
  (1 - above) for non-light ray
  Also include an extra factor of two, to compensate for shooting two rays
```

In some sense, this happens naturally anyway (show picture). However, this algorithm uses two rays on each iteration, rather than one, to decrease variance.

3.2 Russian Roulette

In addition to importance sampling, we can exploit other techniques to decrease variance.

In Russian roulette, we decrease the number of rays by terminating some rays prematurely and giving them extra weight. Specifically we choose a probability p_{term}. We keep track of the cummulative weight in TracePath, and once $weight < p_{term}$, we terminate the path, weighting it by $1/(1-p)$. Note that, using the terminated X' as an estimator for X, we have

$$\mathbb{E}[X'] = p * 0 + (1-p) * \frac{\mathbb{E}[X]}{1-p} = \mathbb{E}[X] \tag{1}$$

Thus, Russian roulette is also unbiased.

3.3 Path Tracing with Direct Lighting

In this modification, we add the direct lighting contribution on every iteration of $TracePath$, rather than stochastically in some iterations.

Step 1:
  Choose a camera ray $r := (x, y, u, v, t)$
  $weight := 1$
  $L = \vec{0}$
Step 2:
  Find surface-ray intersection p
Step 3:
  $L \mathrel{+}= weight \times L[light\_sources]$
  $weight \mathrel{*}= reflectance$
  Choose r'based on the $BRDF(p, r)$
  goto Step 2

4 Path Tracing Extensions

We briefly describe further extensions to the path-tracing algorithm, to improve the speed and quality of the resulting image.

4.1 Unbiased Extensions

4.1.1 Bi-Directional Path Tracing

Some of our previous variations accounted for the importance of direct lighting and weighed it appropriately. Similarly, some indirect lighting effects may be difficult to happen upon when tracing rays from the eye. In bi-directional path tracing, we sample paths starting at both the eye and the lights in the scene.

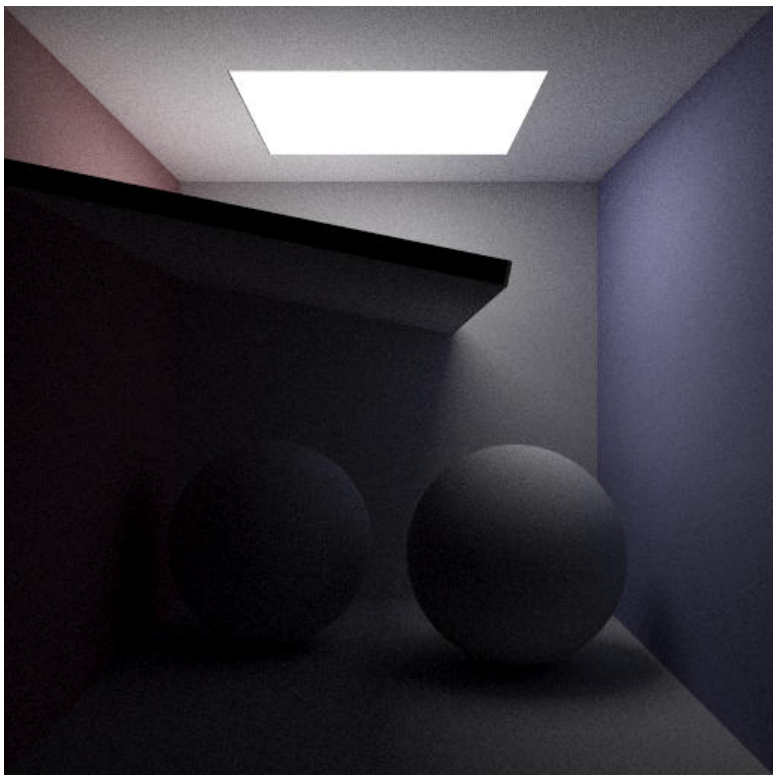

Figure 3: This image was rendered using bi-directional path tracing.

4.1.2 Metropolis Light Transport

Following on this idea of "backward" tracing, the Metropolis Light Tranpsort (MLT) algorithm similarly traces paths starting at the scene lights. The algorithm generates a sequence of paths by mutating a current path with some chosen probability of acceptance. The algorithm is particularly efficient when the scene is largely lit by indirect illumination. The Veach paper on MLT is here.

4.2 Biased, Consistent Extensions

Even if an estimator $T_N(X_1, \ldots, X_N)$ for X is biased, it can still be acceptable if it is consistent. An estimator T_N is consistent if

$$lim_{N->\infty} T_N(X_1, \ldots, X_N) = X$$

. That is, a consistent estimator approaches the correct value in the limit.

4.2.1 Noise Filtering

In path tracing images, most of the noise is due to indirect illumination over multiple bounces, rather than direct illumination. By separating the direct and indirect contributions and applying noise-reducing algorithms on the latter, we can significantly reduce noise in the image without blurring the entire image significantly. The UCSD Graphics page contains a useful reference.

4.2.2 Adaptive Sampling

In adaptive sampling, we devote extra samples to pixels where we have high variance.

4.2.3 Irradiance Caching

Irradiance caching takes advantage of the observation that indirect illumination typically more slowly over a surface than direct illumination. With irradiance caching, we store the results of indirect illumination samples, and use these samples as estimates for nearby points. Using this method, we only compute indirect illumination in areas where we don't have enough estimates, saving significant computational effort. One reference for irradiance caching is given here.

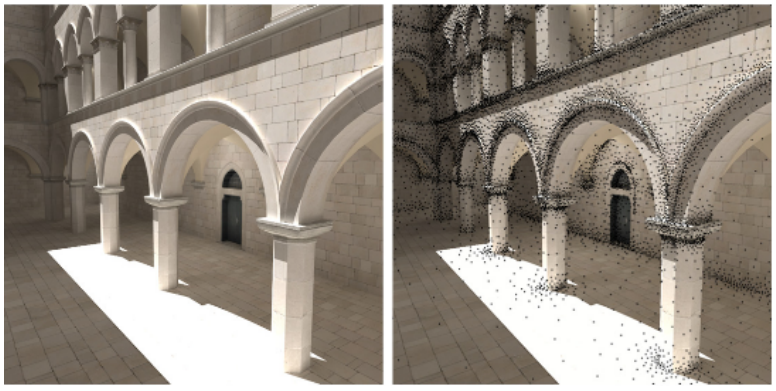

Figure 4: On the left is an image rendered using irradiance caching. On the right, the points where the indirect illumination estimates were cached are visualized on the image.

5 2D Sampling Techniques

We use sampling in a number of ways during path tracing. For instance, we might takes samples with a pixel, on the lens, over an area light source, or over a projected solid angle.

We can generate random directions over a hemisphere via two methods: rejection sampling and inversion. (We've seen these two methods in previous lectures as well.)

5.1 Rejection Sampling

In essence, we generate random numbers in the cube, reject elements outside of the sphere, normalize, and flip to the proper hemisphere if necessary.

1: Generate $(x, y, z), x, y, z \in [-1, 1]$
2: Reject if $x^2 + y^2 + z^2 > 1$
3: $(x', y', z') := Normalize(x, y, z)$
4: Flip if$(x', y', z') \cdot \vec{n} < 0$

Note that rejection sampling only works for uniform distributions. The inversion method, shown below, will generalize to non-uniform distributions.

5.2 Inversion Method

For the upper hemisphere, we have:

$$\begin{aligned} 1 &= \frac{1}{2\pi} \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \sin\theta \, d\theta \, d\phi \\ &= \int_0^{\frac{\pi}{2}} \sin\theta \, d\theta \int_0^{2\pi} \frac{1}{2\pi} \, d\phi \\ &= \int_0^{\frac{\pi}{2}} p(\theta) \, d\theta \int_0^{2\pi} q(\phi) \, d\phi \end{aligned}$$

where $p(\theta) = sin(\theta)$ and $q(\phi) = \frac{1}{2\pi}$ are the separable probability distribution functinos. These yield cumuluative distributions functions of

$$\begin{aligned} P(x) &= \int_0^x p(\theta) \, d\theta \\ &= -\cos\theta \Big]_0^x \\ &= 1 - \cos x \end{aligned}$$

and

$$\begin{aligned} Q(y) &= \int_0^y p(\phi)\, d\phi \\ &= \frac{\phi}{2\pi}\Big]_0^y \\ &= \frac{y}{2\pi} \end{aligned}$$

repsectively. By inverting these CDFs, we can sample the hemisphere surface uniformly. Note that inverting P directly suggests that we assign $\arccos(1-x)$ to θ, rather than $\arccos(x)$. However, sampling $1-x$ and x uniformly are equivalent, so we can use the simpler derivation.

1: Choose $x \in [0,1], y \in [0,1]$, both uniformly
2: $\theta := \arccos(x)$
3: $\phi := 2\pi y$
4: $(x, y, z) := (\sin\theta\cos\phi,\ \sin\theta\sin\phi,\ \cos\theta)$
5: Rotate to be around the normal, instead of the z-axis

5.3 Inversion Method For Importance Sampling

In the Lambertian diffuse model, we'd like to importance sample not only with respect to the BRDF, but also with respect to the cosine of the angle to the normal. Thus, we should use $p(\theta) = \sin\theta\cos\theta$, rather than the $p(\theta) = \sin\theta$ as above.

By performing a similar computation as above, we find that $P(x) = 1 - \cos^2(x)$, after normalization. Thus, for the Lambertian BRDF, we sample as follows:

1: Choose $x \in [0,1], y \in [0,1]$, both uniformly
2: $\theta := \arccos(\sqrt{x})$
3: $\phi := 2\pi y$
4: $(x, y, z) := (\sin\theta\cos\phi, \sin\theta\sin\phi, \cos\theta)$
5: Rotate to be around the normal, instead of the z-axis