

Sampling and Reconstruction of Visual Appearance

CSE 274 [Winter 2018], Lecture 4

Ravi Ramamoorthi

<http://www.cs.ucsd.edu/~ravir>



Motivation: Monte Carlo Path Tracing

- Key application area for sampling/reconstruction
- Core method to solve rendering equation
- Widely used in production (with sample/recon)
- General solution to rendering, global illumination
- Suitable for a variety of general scenes
- Based on Monte Carlo methods
- Enumerate all paths of light transport
- We mostly treat this as a black box, but background is still important

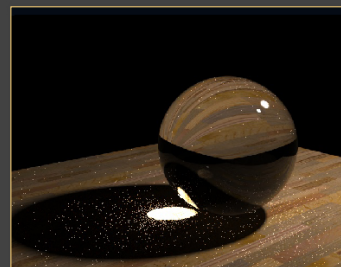
Monte Carlo Path Tracing



Big diffuse light source, 20 minutes

Jensen

Monte Carlo Path Tracing



1000 paths/pixel

Jensen

Monte Carlo Path Tracing

Advantages

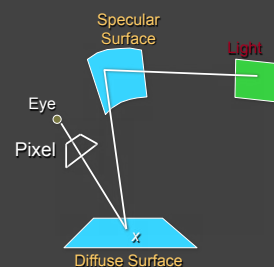
- Any type of geometry (procedural, curved, ...)
- Any type of BRDF (specular, glossy, diffuse, ...)
- Samples all types of paths ($L(SD)^*E$)
- Accuracy controlled at pixel level
- Low memory consumption
- Unbiased - error appears as noise in final image

Disadvantages (standard Monte Carlo problems)

- Slow convergence (square root of number of samples)
- Noise in final image

Monte Carlo Path Tracing

Integrate radiance for each pixel by sampling paths randomly



$$L_o(x, \vec{w}) = L_e(x, \vec{w}) + \int_{\Omega} f_r(x, \vec{w}', \vec{w}) L_r(x, \vec{w}') (\vec{w}' \cdot \vec{n}) d\vec{w}'$$

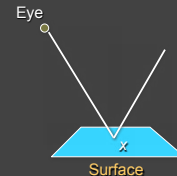
Simple Monte Carlo Path Tracer

- Step 1: Choose a ray (u, v, θ, ϕ) [per pixel]; assign weight = 1
- Step 2: Trace ray to find intersection with nearest surface
- Step 3: Randomly choose between emitted and reflected light
 - Step 3a: If emitted, return $\text{weight}' * L_e$
 - Step 3b: If reflected, $\text{weight}'' = \text{reflectance}$
Generate ray in random direction
Go to step 2

Sampling Techniques

Problem: how do we generate random points/directions during path tracing and reduce variance?

- Importance sampling (e.g. by BRDF)
- Stratified sampling



Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- Other variance reduction methods
- Specific 2D sampling techniques

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p = \text{camera}$, $d = (\theta, \phi)$ within pixel
- Pixel color $\text{+= } (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f_i(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average over paths

- Choose a ray with $p = \text{camera}$, $d = (\theta, \phi)$ within pixel
- Pixel color $\text{+= } (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f_i(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with $p = \text{camera}$, $d = (\theta, \phi)$ within pixel
- Pixel color $\text{+= } (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f_i(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Weight = 1/probability
Remember: unbiased
requires having $f(x) / p(x)$

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

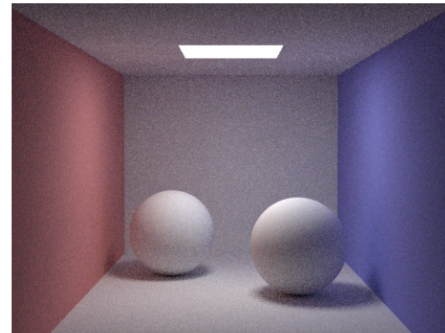
- Choose a ray with p =camera, $d=(\theta, \phi)$ within pixel
- Pixel color $\text{+=} (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Path terminated when
Emission evaluated

Path Tracing



CS348B Lecture 14

10 paths / pixel

Pat Hanrahan, Spring 2009

Arnold Renderer (M. Fajardo)

- Works well diffuse surfaces, hemispherical light



From UCB class many years ago



Advantages and Drawbacks

- Advantage: general scenes, reflectance, so on
 - By contrast, standard recursive ray tracing only mirrors
- This algorithm is *unbiased*, but horribly inefficient
 - Sample "emitted" 50% of the time, even if emitted=0
 - Reflect rays in random directions, even if mirror
 - If light source is small, rarely hit it
- Goal: improve efficiency without introducing bias
 - Variance reduction using many of the methods discussed for Monte Carlo integration last week
 - Subject of much interest in graphics in 90s till today

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- Other variance reduction methods
- Specific 2D sampling techniques

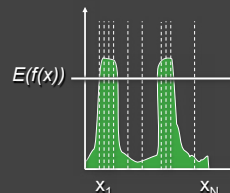
Importance Sampling

- Pick paths based on energy or expected contribution
 - More samples for high-energy paths
 - Don't pick low-energy paths
- At "macro" level, use to select between reflected vs emitted, or in casting more rays toward light sources
- At "micro" level, importance sample the BRDF to pick ray directions
- Tons of papers in 90s on tricks to reduce variance in Monte Carlo rendering
- Importance sampling now standard in production. I consulted on Pixar's system for upcoming movies

Importance Sampling

Can pick paths however we want, but contribution weighted by 1/probability

- Already seen this division of 1/prob in weights to emission, reflectance



$$\int_{\Omega} f(x) dx = \frac{1}{N} \sum_{i=1}^N Y_i$$

$$Y_i = \frac{f(x_i)}{p(x_i)}$$

Simplest Monte Carlo Path Tracer

For each pixel, cast n samples and average

- Choose a ray with p =camera, $d=(\theta, \phi)$ within pixel
- Pixel color $\approx (1/n) * \text{TracePath}(p, d)$

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- Select with probability (say) 50%:
 - Emitted:
 - return $2 * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}}) // 2 = 1/(50\%)$
 - Reflected:
 - generate ray in random direction d'
 - return $2 * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Importance sample Emit vs Reflect

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- If $L_e = (0,0,0)$ then $p_{\text{emit}} = 0$ else $p_{\text{emit}} = 0.9$ (say)
- If $\text{random}() < p_{\text{emit}}$ then:
 - Emitted:
 - return $(1/p_{\text{emit}}) * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}})$
 - Else Reflected:
 - generate ray in random direction d'
 - return $(1/(1-p_{\text{emit}})) * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Importance sample Emit vs Reflect

$\text{TracePath}(p, d)$ returns (r,g,b) [and calls itself recursively]:

- Trace ray (p, d) to find nearest intersection p'
- If $L_e = (0,0,0)$ then $p_{\text{emit}} = 0$ else $p_{\text{emit}} = 0.9$ (say)
- If $\text{random}() < p_{\text{emit}}$ then:
 - Emitted:
 - return $(1/p_{\text{emit}}) * (L_{e_{\text{red}}}, L_{e_{\text{green}}}, L_{e_{\text{blue}}})$
 - Else Reflected:
 - generate ray in random direction d'
 - return $(1/(1-p_{\text{emit}})) * f_r(d \rightarrow d') * (n * d') * \text{TracePath}(p', d')$

Can never be 1 unless
Reflectance is 0

Outline

- Motivation and Basic Idea
- Implementation of simple path tracer
- Variance Reduction: Importance sampling
- Other variance reduction methods
- Specific 2D sampling techniques

More variance reduction

- Discussed “macro” importance sampling
 - Emitted vs reflected
- How about “micro” importance sampling
 - Shoot rays towards light sources in scene
 - Distribute rays according to BRDF

One Variation for Reflected Ray

- Pick a light source
- Trace a ray towards that light
- Trace a ray anywhere except for that light
 - Rejection sampling
- Divide by probabilities
 - $1/(\text{solid angle of light})$ for ray to light source
 - $(1 - \text{the above})$ for non-light ray
 - Extra factor of 2 because shooting 2 rays

Russian Roulette

- Maintain current weight along path (need another parameter to TracePath)
- Terminate ray iff $|\text{weight}| < \text{const.}$
- Be sure to weight by $1/\text{probability}$

Russian Roulette

Terminate photon with probability p

Adjust weight of the result by $1/(1-p)$

$$E(X) = p \cdot 0 + (1-p) \frac{E(X)}{1-p} = E(X)$$

Intuition:

Reflecting from a surface with $R=0.5$

100 incoming photons with power 2 W

1. Reflect 100 photons with power 1 W
2. Reflect 50 photons with power 2 W

CS348B Lecture 14

Pat Hanrahan, Spring 2009

Path Tracing: Include Direct Lighting

```
Step 1. Choose a camera ray  $r$  given the
(x,y,u,v,t) sample
weight = 1;
L = 0
Step 2. Find ray-surface intersection
Step 3.
L += weight * Lr(light sources)
weight *= reflectance(r)
Choose new ray  $r' \sim \text{BRDF pdf}(r)$ 
Go to Step 2.
```

CS348B Lecture 14

Pat Hanrahan, Spring 2009

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching

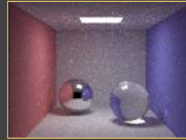
Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



Unfiltered



Filtered

Jensen

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching



Fixed



Adaptive

Ohbuchi

Monte Carlo Extensions

Unbiased

- Bidirectional path tracing
- Metropolis light transport

Biased, but consistent

- Noise filtering
- Adaptive sampling
- Irradiance caching

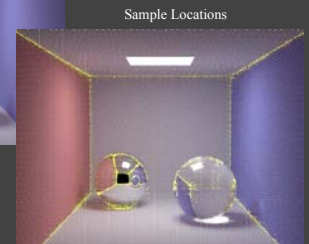


Jensen

Irradiance Caching Example

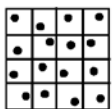


Final Image



Sample Locations

Stratified Sampling



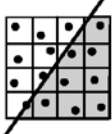
Stratified sampling like jittered sampling

Allocate samples per region

$$N = \sum_{i=1}^m N_i \quad F_N = \frac{1}{N} \sum_{i=1}^m N_i F_i$$

New variance

$$V[F_N] = \frac{1}{N^2} \sum_{i=1}^m N_i V[F_i]$$



Thus, if the variance in regions is less than the overall variance, there will be a reduction in resulting variance

For example: An edge through a pixel

$$V[F_N] = \frac{1}{N^2} \sum_{i=1}^m N_i V[F_i] = \frac{V[F_E]}{N^{1.5}}$$

CS348B Lecture 9

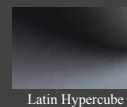
Pat Hanrahan, Spring 2002

D. Mitchell 95, Consequences of stratified sampling in graphics

Comparison of simple patterns



Ground Truth



Latin Hypercube



Quasi Monte Carlo



Uniform

Random

Stratified

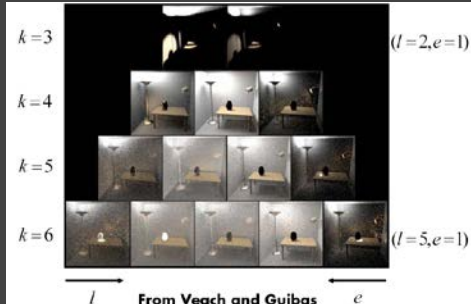
16 samples for area light, 4 samples per pixel, total 64 samples

If interested, see my recent paper "A Theory of Monte Carlo Visibility Sampling"

Figures courtesy Tianyu Liu

Bidirectional Path Tracing

Path pyramid ($k = l + e = \text{total number of bounces}$)



Comparison



Mies House: Swimming Pool



Summary

- Monte Carlo methods robust and simple (at least until nitty gritty details) for global illumination
- Must handle many variance reduction methods in practice
- Importance sampling, Bidirectional path tracing, Russian roulette etc.
- Rich field with many papers, systems researched over last 30 years
- For rest of the course, we largely take this as a black box, focusing on sampling and reconstruction