Monte Carlo Path Tracing

COS 526, Fall 2012
Tom Funkhouser
Slides from Rusinkiewicz, Shirley

Outline

- Motivation
- Monte Carlo integration
- Variance reduction techniques
- Monte Carlo path tracing
- Sampling techniques
- Conclusion

Motivation

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

Motivation

\[ L_e = \int_{A_e} L_e dA \]

Motivation

\[ L_e(x) = L_e(x_e) + \int_{0}^{1} f(V(x_e, x_t)) d\tau \rightarrow S \rightarrow e \rightarrow \|V(x_e, x_t)\|G(x_e, x_t) dA \]

Motivation

\[ L_e(x) = L_e(x_e) + \int_{0}^{1} f(V(x_e, x_t)) d\tau \rightarrow S \rightarrow e \rightarrow \|V(x_e, x_t)\|G(x_e, x_t) dA \]
**Motivation**

- Rendering = integration
  - Antialiasing
  - Soft shadows
  - Indirect illumination
  - Caustics

\[ L(x,\vec{n}) = L(x,\vec{n}) + \int_{\Omega} f(x,\vec{x'},\vec{n}) L(\vec{x'},\vec{n'} \cdot \vec{n} \cdot \vec{n'}) \, d\vec{n'} \]

**Challenge**

- Rendering integrals are difficult to evaluate
  - Multiple dimensions
  - Discontinuities
    - Partial occluders
    - Highlights
    - Caustics

\[ L(x,\vec{n}) = L(x,\vec{n}) + \int_{\Omega} f(x,\vec{x'},\vec{n}) L(\vec{x'},\vec{n'} \cdot \vec{n} \cdot \vec{n'}) \, d\vec{n'} \]
**Outline**

- Motivation
- Monte Carlo integration
- Variance reduction techniques
- Monte Carlo path tracing
- Sampling techniques
- Conclusion

**Integration in 1D**

\[
\int_a^b f(x) \, dx = \int_a^b g(x) \, dx
\]

**We can approximate**

\[
\int_a^b f(x) \, dx = \int_a^b g(x) \, dx
\]

**Or we can average**

\[
\int_a^b f(x) \, dx = E(f(x))
\]

**Estimating the average**

\[
\int_a^b f(x) \, dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i)
\]

**Other Domains**

\[
\int_a^b f(x) \, dx = \frac{b-a}{N} \sum_{i=1}^{N} f(x_i)
\]
Multidimensional Domains

- Same ideas apply for integration over ...
  - Pixel areas
  - Surfaces
  - Projected areas
  - Directions
  - Camera apertures
  - Time
  - Paths

\[ \int_{\text{pixel}} f(x) \, dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

Efficiency?

\[ \text{Var}[f(x)] = \frac{1}{N} \sum_{i=1}^{N} (f(x_i) - E[f(x)])^2 \]

Efficiency?

\[ \text{Var}[E[f(x)]] = \frac{1}{N} \text{Var}[f(x)] \]

Variance decreases as $1/N$
Error decreases as $1/\sqrt{N}$

Outline

- Motivation
- Monte Carlo Integration
- Variance reduction techniques
- Monte Carlo path tracing
- Sampling techniques
- Conclusion

Variance Reduction Techniques

- Stratified sampling
- Importance sampling
- Metropolis sampling
- Quasi-random

\[ \int_{\Omega} f(x) \, dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

Stratified Sampling

- Estimate subdomains separately

\[ E_{\text{str}}(f(x)) \]

\[ x_1 \quad x_n \]
Stratified Sampling

- This is still unbiased

\[ E_{\text{strat}} = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

\[ = \frac{1}{N} \sum_{i=1}^{N} N_i E_i \]

Variances:

\[ \text{Var}[E_{\text{strat}}] = \frac{1}{N^2} \sum_{i=1}^{N} \text{Var}[E_i] \]

Variance Reduction Techniques

- Stratified sampling
- Importance sampling
- Metropolis sampling
- Quasi-random

\[ \int f(x) dx = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

Importance Sampling

- Put more samples where \( f(x) \) is bigger

\[ \int \frac{f(x)}{p(x)} dx = \frac{1}{N} \sum_{i=1}^{N} Y_i \]

\[ Y_i = \frac{f(x_i)}{p(x_i)} \]

- How do we draw samples with probability proportional to function value?

Importance Sampling

- This is still unbiased

\[ E[Y] = \int \frac{Y(x)}{p(x)} \int_{A} \]

\[ = \int_{A} \frac{f(x)}{p(x)} \int_{A} p(x) \int_{A} \]

\[ = \int_{A} f(x) dx \]

for all \( N \)
**Importance Sampling**

- Sampling uniform distribution:
  - Use random number generator

![Probability Distribution](image1)

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

![Cumulative Probability Distribution](image2)

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

![Cumulative Probability Distribution](image3)

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

![Cumulative Probability Distribution](image4)
Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

\[
\begin{align*}
\text{Cumulative Probability} & \quad 0 \quad \Omega \\
0 & \quad 0 \\
1 & \quad 1
\end{align*}
\]

Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

\[
\begin{align*}
\text{Probability} & \quad 0 \quad 1 \\
0 & \quad 0 \\
1 & \quad 1
\end{align*}
\]

Importance Sampling

- Sampling specific probability distribution:
  - Function inversion
  - Rejection

\[
\begin{align*}
\text{Probability} & \quad 0 \quad 1 \\
0 & \quad 0 \\
1 & \quad 1
\end{align*}
\]

Combining Multiple PDFs

- Balance heuristic
  - Use combination of samples generated for each PDF
  - Number of samples for each PDF chosen by weights
  - Near optimal

Outline

- Motivation
- Monte Carlo integration
- Variance reduction techniques
- Monte Carlo path tracing
- Sampling techniques
- Conclusion
Monte Carlo Path Tracing

- Integrate radiance for each pixel by sampling paths randomly.

\[ L(x, y) = L_0(x, y) - \int_D f(x, y, \omega) L(x, y, \omega') \, d\omega' \]

Monte Carlo Path Tracer

- For each pixel, repeat \( n \) times:
  - Choose a ray with \( p = \text{camera}, \ d = (\theta, \phi) \) within pixel
  - Pixel color += \( \frac{1}{n} \times \text{TracePath}(p, d) \)

- Use stratified sampling to select rays within each pixel.

\[ \int_{\Omega} f(x) \, d\omega = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \]

TracePath

- \( \text{TracePath}(p, d) \) returns \((r, g, b)\):
  - Trace ray \((p, d)\) to find nearest intersection \(p'\)
  - Sample radiance leaving \(p'\) towards \(p\)

TracePath

- Can sample radiance however we want, but contribution weighted by \(1/\text{probability}\).

\[ E(t(x)) = \frac{1}{N} \sum_{i=1}^{N} Y_i \]

TracePath

- \( \text{TracePath}(p, d) \) returns \((r, g, b)\):
  - Trace ray \((p, d)\) to find nearest intersection \(p'\)
  - If \( L_e = (0,0,0) \) then \( p_{\text{emit}} = 0 \)
  - else if \( f_r = (0,0,0) \) then \( p_{\text{emit}} = 1 \)
  - else \( p_{\text{emit}} = .9 \)
  - If \( \text{random()} < p_{\text{emit}} \) then
    - Emitted:
      \[ \text{return} \ (1/p_{\text{emit}}) \times (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) \]
    - Reflected:
      \[ \text{generate ray in random direction } d' \]
      \[ \text{return} \ (1/(1-p_{\text{emit}})) \times (L_{\text{red}}, L_{\text{green}}, L_{\text{blue}}) \times \text{TracePath}(p', d') \]
TracePath

- Reflected case:
  - Pick a light source
  - Trace a ray towards that light
  - Trace a ray anywhere except for that light
    - Rejection sampling
      - \( \frac{1}{\text{solid angle of light}} \) for ray to light source
      - \((1 - \text{the above})\) for non-light ray

TracePath(p, d) returns (r, g, b):

- Reflected:
  - \( \frac{1}{2} \frac{1}{p_{\text{light}}} \) *
  - \( \left((d \cdot d') \cdot (n \cdot d) \right) \) *
  - \( \text{TracePath}(p', d') \)

- Reflected:
  - \( \frac{1}{2} \frac{1}{1 - p_{\text{light}}} \) *
  - \( \left((d \cdot d') \cdot (n \cdot d) \right) \) *
  - \( \text{TracePath}(p', d') \)

return \( \frac{1}{1 - p_{\text{emit}}} \) * \( L_r \)

Reflected Ray Sampling

- Uniform directional sampling: how to generate random ray on hemisphere?
- Option #1: rejection sampling
  - Generate random numbers \((x, y, z)\), \(x, y, z \in -1..1\)
  - If \(x^2 + y^2 + z^2 > 1\), reject
  - Normalize \((x, y, z)\)
  - If pointing into surface (ray dot n < 0), flip

Reflected Ray Sampling

- Option #2: inversion method
  - In polar coords, density must be proportional to \( \sin \theta \)
  - Integrate, invert \( \rightarrow \cos^{-1} \)
  - So, recipe is
    - Generate \( \phi \) in \(0..2\pi\)
    - Generate \( z \) in \(0..1\)
    - Let \( \theta = \cos^{-1} z \)
    - \((x, y, z) = (\sin \phi \cos \theta, \sin \phi \sin \theta, \cos \theta)\)

BRDF Importance Sampling

- Better than uniform sampling: importance sampling
- Because you divide by probability, ideally:
  - probability \( \propto f_r \cdot \cos \theta \)
- \([\text{Lafortune, 1994}]\):
  \[ f_r(x, \alpha_l, \alpha_n) = k_r \frac{1}{\pi} + k_s \frac{n+2}{2\pi} \cos^n \alpha \]
BRDF Importance Sampling

- For cosine-weighted Lambertian:
  - Density = \( \cos \theta \sin \theta \)
  - Integrate, invert \( \rightarrow \cos^{-1}(\sqrt{z}) \)
- So, recipe is:
  - Generate \( f \) in 0..2\( \pi \)
  - Generate \( z \) in 0..1
  - Let \( \theta = \cos^{-1}(\sqrt{z}) \)

BRDF Importance Sampling

- Recipe for sampling specular term:
  - Generate \( z \) in 0..1
  - Let \( \alpha = \cos^{-1}(z^{1/(n+1)}) \)
  - Generate \( f_{\alpha} \) in 0..2\( \pi \)
- This gives direction w.r.t. ideal mirror direction

BRDF Importance Sampling

- Phong BRDF: \( f \propto \cos^n \alpha \) where \( \alpha \) is angle between outgoing ray and ideal mirror direction
- Constant scale = \( k_s(n+2)/(2\pi) \)
- Ideally we would sample this times \( \cos \theta \)
  - Difficult!
  - Easier to sample BRDF itself, then multiply by \( \cos \theta \)
  - That's OK — still better than random sampling

Recap

- Recap
  - TracePath\((p, d)\) returns \((r,g,b)\):
    - Trace ray \((p, d)\) to find nearest intersection \( p' \)
    - If \( Le = (0,0,0) \) then \( p' = \rho \cdot d \)
    - If \( \text{min} f_{\rho} = 0.0 \) then \( p_{\rho} = 1 \)
    - If \( p_{\rho} = 0 \) then \( \text{Random} \text{p}_{\rho} \)
      - Emit: \( \frac{1}{p_{\rho}} \cdot (Le_{\text{red}}, Le_{\text{green}}, Le_{\text{blue}}) \)
      - Reflect: \( \frac{1}{2} \cdot p_{\rho} + \frac{1}{2} \cdot (f_{\rho}(d \rightarrow d')) \cdot (n \cdot d') \cdot \text{TracePath}(p', d') \)
      - Refract: \( \frac{1}{2} \cdot p_{\rho} + \frac{1}{2} \cdot (1 - f_{\rho}(d \rightarrow d')) \cdot (n \cdot d') \cdot \text{TracePath}(p', d') \)
    - return \( \frac{1}{1-p_{\rho}} \cdot L_r \)

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
  - Slow convergence
  - Noise in final image
Summary

- Monte Carlo Integration Methods
  - Very general
  - Good for complex functions with high dimensionality
  - Converge slowly (but error appears as noise)

- Conclusion
  - Preferred method for difficult scenes
  - Noise removal (filtering) and irradiance caching (photon maps) used in practice

More Information

- Books
  - Realistic Ray Tracing, Peter Shirley
  - Realistic Image Synthesis Using Photon Mapping, Henrik Wann Jensen

- Theses
  - Robust Monte Carlo Methods for Light Transport Simulation, Eric Veach
  - Mathematical Models and Monte Carlo Methods for Physically Based Rendering, Eric La Fortune

- Course Notes