Computer Graphics 14 - Global illumination 1

Tom Thorne

Slides courtesy of Taku Komura
www.inf.ed.ac.uk/teaching/courses/cg
Overview

- Global illumination and light transport
- Monte-Carlo integration
- Monte-Carlo Ray Tracing
  - Path Tracing
  - Bidirectional Path Tracing
- Photon Mapping
Colour bleeding
Caustics
Light transport notations

- It is useful to be able to describe the path that light takes through a scene:
  - L light source
  - E the eye
  - S specular reflection or refraction
  - D diffuse reflection
Light transport notations

- Regular expressions
  - \((k)^+\) : one or more of event k
  - \((k)^*\) : zero or more of event k
  - \((k)^?\) : zero or one of event k
  - \((k|k')\) : event k or k’
Examples

LDDE

LSDE
Ray Tracing: review

- Shadow ray, reflection ray, etc.
- We calculate local illumination at diffuse surfaces using the direct lighting
- We do not know where the indirect (ambient) light illuminating diffuse surfaces comes from
Indirect lighting by ray-tracing

- Caustics and colour bleeding are produced by indirect light – how can we simulate such effects in the ray tracing framework?
Rendering equation

\[ L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{\Omega} L_i(x, \omega_i) f_r(x, \omega_i, \omega_o)(\omega_i \cdot n_x) d\omega_i \]

- \( L_o(x, \omega_o) \) outgoing radiance at point \( x \) in direction \( \omega_o \)
- \( L_e \) emitted radiance
- \( L_i(x, \omega_i) \) incoming radiance from direction \( \omega_i \)
- \( f_r \) is the Bidirectional Reflectance Distribution Function (BRDF) of the surface
- \( n_x \) is the surface normal at point \( x \)
- \( \Omega \) is the hemisphere of incoming directions at point \( x \)
Monte Carlo integration

To estimate an integral for some function $f$,

$$I = \int_{\Omega} f(x) dx,$$

we can generate $N$ uniform random samples within $\Omega$, $\xi_1, \ldots, \xi_N$ and then approximate $I$ as:

$$I \approx V \frac{1}{N} \sum_i f(\xi_i),$$

where $V = \int_{\Omega} dx$. Then

$$\lim_{N \to \infty} V \frac{1}{N} \sum_{i=1}^{N} f(\xi_i) = I.$$
Monte Carlo integration example

For the region
\[ D = (-1 \leq x \leq 1, -1 \leq y \leq 1), \]
we define the function \( f(x, y) : \)

- \[ f(x, y) = \begin{cases} 1 & \text{if } x^2 + y^2 \leq 1 \\ 0 & \text{otherwise} \end{cases} \]
- \[ \int_D f(x, y) \, dx \, dy = \pi \]
- \[ \pi \approx 4 \frac{1}{N} \sum_i f(x_i, y_i) \]
Two ways to simulate indirect light

- Launch tracing rays in random directions at diffuse surfaces: Path tracing
- Shoot rays that represent the path of light from the light source: Bidirectional Path Tracing, Photon Mapping
Path Tracing

- An enhancement of the ordinary ray-tracing scheme
- When hitting a diffuse surface, pick one direction at random, and find the colour of the incoming light
- Trace many paths per pixel (100-10000 per pixel)
- by Kajiya, SIGGRAPH 86
Original Ray Tracing Algorithm

- **Trace (ray)**
  - Find the intersection of the ray and the scene
  - Compute the shadow ray: Color = Color_ambient
  - Do the local illumination: Color += Color_local (not shadowed)
  - If specular compute the reflection vector R
    - **Color += Trace(R)**
  - If refractive compute the refractive vector T
    - **Color += Trace(T)**
Path Tracing Algorithm

• **Trace (ray)**
  - Find the intersection of the ray and the scene
  - Compute the shadow ray: Color = Color_ambient
  - Do the local illumination: Color += Color_local (not shadowed)
  - If specular compute the reflection vector R
    • Color += Trace(R)
  - If refractive compute the refractive vector T
    • Color += Trace(T)
  - Else if diffuse compute a random vector R’
    • Color += Trace(R’)
Path tracing: problems

- Variance in the pixel colours, appearing as noise
- Need many samples for precise results
  - Requires 1000~10000 samples per pixel for good results

10 paths/pixel    100 paths/pixel    1000 paths/pixel
Path tracing: problems

- Some lights are difficult to reach from the camera - such as those produced by spot lights
- For such lights, we cannot simulate indirect light well
- Results in a very dim image with high variance
Why? Shadow rays are always occluded

- For the pixel to be lit, the path must be lucky enough to reach the light source

Point light  Spot light
Bidirectional path tracing

- Compute a light path $y_0, y_1, \ldots, y_n$
- Compute an eye path $x_0, x_1, \ldots, x_m$
- The colour of the fragment at $x_1$ is
  - The amount of light reaching $x_1$ from $y_0, \ldots, y_n$ and reflecting towards $x_0$ plus
  - The amount of light reaching $x_1$ from $x_2$ and reflecting towards $x_0$
Comparison

(a) Bidirectional path tracing with 25 samples per pixel

(b) Standard path tracing with 56 samples per pixel (the same computation time as (a))
Benefits of bidirectional method

• Caustics
  • Easier to produce by tracing from the light source
  • When the light sources are not easy to reach from the eye
Metropolis-Hastings algorithm

If we want to sample from some probability density function $f(x)$, we can generate a Markov Chain whose stationary distribution is $f(x)$ using the following algorithm:

\begin{verbatim}
for $i = 1 \ldots N$ do
    Generate $x' \sim q(x \rightarrow \cdot)$;
    Generate $t \sim \text{Uniform}(0, 1)$;
    $a \leftarrow \min \left(1, \frac{f(x')q(x' \rightarrow x)}{f(x)q(x \rightarrow x')} \right)$;
    if $t < a$ then
        $x \leftarrow x'$;
    end
end
\end{verbatim}

where $q(x \rightarrow x')$ is a proposal distribution that generates random moves from the current state of the chain.
Metropolis light transport

- Bidirectional mutation:
  - Delete a subpath and sample a new one

- Perturbation:
  - Move intersection points within a subpath

- If a proposal is not valid, reject immediately

Top: Bidirectional path tracing, Bottom: Metropolis light transport. Same computation time as path tracing.
Summary for Monte Carlo Ray Tracing

- An approach that simulates the light reflection at diffuse surfaces
- Can simulate indirect lighting
- Results are subject to variance
  - Requires a lot of samples per pixel to reduce the noise
  - Bidirectional methods can reduce the noise
Overview

- Global illumination and light transport
- Monte-Carlo integration
- Monte-Carlo Ray Tracing
  - Path Tracing
  - Bidirectional Path Tracing
- Photon Mapping
Photon Mapping

- A fast, global illumination algorithm based on Monte-Carlo method
- A stochastic approach that estimates the radiance from a limited number of samples

http://www.youtube.com/watch?v=wqWRVcsIcAQ
Photon Mapping

- A two pass global illumination algorithm
  - First Pass - photon tracing:
    - Casting photons from the light source
    - Storing photon positions in the “photon map”,
  - Second Pass – rendering (radiance estimate):
    - the shading of pixels is estimated from the photon map
Photon emission

- A photon’s life begins at the light source.
- Different types of light sources
- Brighter lights emit more photons
Photon scattering

- Emitted photons 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.
What happens when photons hit surfaces?

- Photons are reflected or absorbed. There are two ways to determine this:
  - Attenuate the power and reflect the photon
    - For arbitrary BRDFs
  - Use Russian Roulette techniques
    - Decide stochastically whether the photon is reflected or absorbed based on the probability of reflection, and do not attenuate power if it is reflected.
Russian Roulette

- If the surface is diffuse and specular, a Monte Carlo technique called Russian Roulette is used to probabilistically decide whether photons are reflected, refracted or absorbed.
- Produce a random number between 0 and 1
- Determine whether to transmit, absorb or reflect in a specular or diffusive manner, according to the value.
Probability of reflection and absorption

- Probability of reflection

\[ P_r = \max(d_r + s_r, d_g + s_g, d_b + s_b) \]

- Probability of diffuse reflection

\[ P_d = \frac{d_r + d_g + d_b}{d_r + d_g + d_b + s_r + s_g + s_b} \ P_r . \]

- Probability of specular reflection

\[ P_s = \frac{s_r + s_g + s_b}{d_r + d_g + d_b + s_r + s_g + s_b} \ P_r = P_r - P_d . \]
Diffuse and specular reflection

- If the photon is to make a diffuse reflection, randomly determine the direction.
- If the photon is to make a specular reflection, reflect in the mirror direction.
Power attenuation

- The colour of the light must change after specular/diffuse reflection
- This is essential for producing effects like colour bleeding
The power after reflection $\Phi_{\text{ref}}$ for incident photon with power $\Phi_i$ is

Specular reflection:
\[
\begin{align*}
\Phi_{\text{ref},r} &= \frac{\Phi_{i,r}s_r}{P_s} \\
\Phi_{\text{ref},g} &= \frac{\Phi_{i,g}s_g}{P_s} \\
\Phi_{\text{ref},b} &= \frac{\Phi_{i,b}s_b}{P_s}
\end{align*}
\]

Diffuse reflection:
\[
\begin{align*}
\Phi_{\text{ref},r} &= \frac{\Phi_{i,r}d_r}{P_d} \\
\Phi_{\text{ref},g} &= \frac{\Phi_{i,g}d_g}{P_d} \\
\Phi_{\text{ref},b} &= \frac{\Phi_{i,b}d_b}{P_d}
\end{align*}
\]
Photon Map

• When a photon makes a diffuse bounce, or is absorbed at the surface, the ray intersection is stored in memory:
  - 3D coordinates on the surface
  - Color intensity
  - Incident direction
• The data structure is called Photon Map
• The photon data is not recorded for specular reflections
Second Pass – Rendering

- Finally, a traditional ray tracing procedure is performed by shooting rays from the camera.
- At the location the ray hits the scene, a sphere is created and enlarged until it includes $N$ photons.
Radiance Estimation

- The radiance estimate can be written by the following equation:

\[ L_r(x, \omega) = \sum_{p=1}^{N} f_r(x, \omega_p, \omega) \frac{\Delta \Phi_p(x, \omega_p)}{\Delta A} \]

- \( x \): location the ray hits the scene
- \( \omega \): direction towards the camera
- \( \omega_p \): incident vector of photon \( p \)
- \( f_r \): BRDF
- \( N \): the number of photons
- \( \Delta \Phi_p \): power of photon \( p \)
- \( \Delta A \): Area of the circle \( \pi r^2 \)
Radiance Estimation

\[ L_r(x, \omega) = \sum_{p=1}^{N} f_r(x, \omega_p, \omega) \frac{\Delta \Phi_p(x, \omega_p)}{\Delta A} \]
Radiance Estimation

\[ L_r(x, \omega) = \sum_{p=1}^{N} f_r(x, \omega_p, \omega) \frac{\Delta \Phi_p(x, \omega_p)}{\Delta A} \]

Amount of light coming in from \( \omega_p \) going back toward \( \omega \)
Radiance Estimation

\[
L_r(x, \omega) = \sum_{p=1}^{N} f_r(x, \omega_p, \omega) \frac{\Delta \Phi_p(x, \omega_p)}{\Delta A}
\]
Data structure for photon data

• We need an efficient data structure for retrieving photon maps when colouring the pixels
  - KD-tree
  - Spatial Hash
Storing photons: kd-tree

- An efficient hierarchical data structure for saving spatial data
- Procedure to produce it:
  - divide the samples at the median along current axis (e.g. x,y or z)
  - The median sample becomes the parent node, and the samples on either side become the child nodes
  - Further subdivide the child trees on the next axis (rotating through x,y,z)
- Can efficiently find the neighbours when rendering the scene
Query for $N$-nearest neighbouring photons

- Given a point $X$, we traverse the tree to find the nearest $N$ points to $X$
- Start from the root, check if the bounding circle is totally within one side or not
- If it is, then you do not have to search the other side
Query for N-nearest neighbouring photons

- If the photon is within bounding volume, you add it into the heap
- Descend to the children (only if they are within the bounding distance)
- The heap is sorted so that the farthest photon is on the top.
- Only the top N photons are kept in the heap.
Storing photons: spacial hashing

- A uniform 3D grid based hashing system
- Create a hash function that maps each grid region to a list that stores the photons in that region
- Scan the photons in the list to find those close to the sample point
Nearest neighbour search

- Decide the maximum radius of search
- Examine the distance between the sample point and the photons in the grid
- Gradually increase the radius, search in all the reachable grids until we reach the photon count
- Suitable for hardware implementation
Precision

- The precision of the final results depends on
  - the number of photons emitted
  - the number of photons counted for calculating the radiance
- 10000 photons and 50 samples (left), and 500000 photons and 500 samples (right)
Photon mapping: 250000 photons, 15 seconds

Path tracing
Summary

- **Monte Carlo Ray Tracing**
  - Accurate but requires a lot of samples per pixel
  - Suffers from noise which is due to variance
  - Bidirectional method can reduce the variance

- **Photon Mapping**
  - A stochastic approach that estimates the radiance from a limited number of photons
  - Requires less samples compared to path tracing
References

★ Shirley Chapter 24 (Global illumination)

★ A Practical Guide to Global Illumination using Photon Maps
  – Siggraph 2000 Course 8

• http://graphics.stanford.edu/papers/metro/


• Global Illumination using Photon Maps (Rendering Techniques ‘96) Henrik Wann Jensen (http://graphics.ucsd.edu/~henrik/)