Computer Graphics
Global Illumination:
Monte-Carlo Ray Tracing and
Photon Mapping

Lecture 14
Taku Komura
In the last lecture

• We did ray tracing and radiosity
• Ray tracing is good to render specular objects but cannot handle indirect diffuse reflections well
• Radiosity can render indirect diffuse reflections but not specular reflections
• They have to be ad-hocly combined to synthesize photo-realistic images
• Still cannot simulate effects such as caustics (LSDE)
Today

• Other practical methods to synthesize photo-realistic images
• Monte-Carlo Ray Tracing
  – Path Tracing
  – Bidirectional Path Tracing
• Photon Mapping
Today : Global Illumination Methods

- Monte-Carlo Ray Tracing
- Photon Mapping
Lighting at diffuse surfaces in ray tracing

- In regular ray-tracing, we simply do a local illumination at diffuse surfaces using the direct light.
- The in-direct light that reaches the diffuse surface are not considered.
Problems simulating indirect lighting by ray-tracing

- Caustics and color bleeding are produced by such light – how can we simulate such effects in the ray tracing framework?
- We do not know where the light that lit the diffuse surface is coming from
Two ways to simulate indirect light

- Launch tracing rays in random directions at diffuse surfaces  \(\rightarrow\) Path tracing
- Shoot rays that represent the path of light from the light source  \(\rightarrow\) Bidirectional Path Tracing
Monte-Carlo Ray Tracing: Path Tracing

- An enhancement of the ordinary ray-tracing scheme
- But when hitting a diffuse surface, pick one ray 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: \( \text{Color} = \text{Color}_{\text{ambient}} \)
  – Do the local illumination: \( \text{Color} += \text{Color}_{\text{local}} \) (not shadowed)
  – If specular compute the reflection vector \( R \)
    • \( \text{Color} += \text{Trace}(R) \)
  – If refractive compute the refractive vector \( T \)
    • \( \text{Color} += \text{Trace}(T) \)
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’)

[Diagram showing light, camera, and diffuse surface]
Path Tracing: algorithm

**Render image using path tracing**
for each pixel
  color = 0
  For each sample
    pick ray in pixel
    color = color + \text{trace}(ray)
  pixel\_color = color/#samples

\text{trace}(ray)
  find nearest intersection with scene
  compute intersection point and normal
  color = \text{shade}(point, normal)
return color

**Shade** (point, normal)
color = 0
for each light source
  test visibility on light source
  if visible
    color=color+direct illumination
  \text{if diffuse surface}
    color = color + \text{trace}(a \text{ randomly reflected ray})
  \text{else if specular}
    color = color + \text{trace}(reflection ray)
  \text{else if refraction}
    color = color + \text{trace}(refraction ray)
return color
Path tracing: problems

- Variance in the pixel colors, appearing as noise
- Need many samples for precise results
  - Requires 1000 ~ 10000 samples per pixel for good results
Path Tracing: Problems (2)

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

- When the light hits a diffuse surface, the light is reflected in random directions
  - This is like another light source
  - We should use this for local illumination (shadow ray) too
Bidirectional Path Tracing

• Send paths from light source, record path vertices
• Send paths from eye, record path vertices
• Connect every vertex of eye path with every vertex in light path
• Assume the vertex of the light paths are light sources

Lafortune & Willems, Compugraphics ’93, Veach & Guibas, EGRW 94
Procedure

- Compute light path $y_0, y_1, \ldots, y_n$
- Compute eye path $x_0, x_1, \ldots, x_m$
- The color 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 $x_1$
Procedure

• The color 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 $x_0$

\[
L_{i,j}(x_1 \rightarrow x_0) = \left( f_r(y_j \rightarrow x_1 \rightarrow x_0) V(x_1, y_j) \left( \frac{(y_j \rightarrow x_1) \cdot \hat{n}_{x_1}}{||x_1 - y_j||^2} \right) I(y_j \rightarrow x_1) \right)
\]
Procedure

\[ L_{1,j}(x_1 \to x_0) = \]

\[ f_r(y_j \to x_1 \to x_0) \cdot V(x_1, y_j) \left( \frac{(y_j \to x_1) \cdot \vec{n}_{x_1}}{\|x_1 - y_j\|^2} \right) I(y_j \to x_1) \]

BRDF at \( x_1 \)

Light coming from \( y_j \) reflected at \( x_1 \)
towards \( x_0 \)
Procedure

\[ L_{i,j}(x_1 \rightarrow x_0) = \]

\[ f_r(y_j \rightarrow x_1 \rightarrow x_0)V(x_1, y_j) \left( \frac{(y_j \rightarrow x_1) \cdot \vec{n}_{x_1}}{\|x_1 - y_j\|^2} \right) I(y_j \rightarrow x_1) \]

Visibility of \textit{x1} from \textit{yj} :
if visible 1, otherwise 0
Procedure

\[ L_{1,j}(x_1 \rightarrow x_0) = \]

\[ f_r(y_j \rightarrow x_1 \rightarrow x_0) V(x_1, y_j) \frac{|(y_j \rightarrow x_1) \cdot \vec{n}_{x_1}|}{\|x_1 - y_j\|^2} I(y_j \rightarrow x_1) \]

Energy leaving \( y_j \) towards \( x_1 \)

\[ I(y_j \rightarrow x_i) = \Phi_i(y_j) |(y_j \rightarrow x_i) \cdot \vec{n}_{y_j}| f_r(y_{j-1} \rightarrow y_j \rightarrow x_i) \]

\( x_1 \Phi_i(y_j) : \) flux of the incoming photon at \( y_j \)
Procedure

\[ L_{1,j}(x_1 \rightarrow x_0) = \]

\[ f_r(y_j \rightarrow x_1 \rightarrow x_0)V(x_1, y_j) \left| \frac{(y_j \rightarrow x_1) \cdot \vec{n}_{x_1}}{\|x_1 - y_j\|^2} \right| I(y_j \rightarrow x_1) \]

Energy reaching \( x_1 \) from \( y_j \)
Sum this for all the light bounces
Procedure

- The color 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 \( x_0 \)

\[
L_{i,j}(x_1 \rightarrow x_0) = \sum_{j=0}^{n} f_r(y_j \rightarrow x_1 \rightarrow x_0) V(x_1, y_j) \left| \frac{(y_j \rightarrow x_1) \cdot \tilde{n}_{x_1}}{||x_1 - y_j||^2} \right| I(y_j \rightarrow x_1)
\]
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))
What about the scenes below?
In what case bidirectional methods work better than one way path tracing?

• Caustics
  – Easier to produce by tracing from the light source
• When the light sources are not easy to reach from the eye
  – less shadow rays reach the light source
• Indoor scenes where indirect lighting is important
  – Bidirectional methods take into account the inter-reflections at diffuse surfaces
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
Today : Global Illumination Methods

• Monte-Carlo Ray Tracing
• Photon Mapping
Photon Mapping

- A fast, global illumination algorithm based on Monte-Carlo method
  1. Casting photons from the light source, and
  2. saving the information of reflection in the “photon map”, then
  3. render the results

A stochastic approach that estimates the radiance from limited number of samples

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

• A two pass global illumination algorithm
  – First Pass - Photon Tracing
  – Second Pass – Rendering (radiance estimate)
Photon Tracing

• The process of emitting discrete photons from the light sources and
• tracing them through the scene
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 to do when the photons hit surfaces?

• Attenuate the power and reflect the photon
  – For arbitrary BRDFs

• Use Russian Roulette techniques
  – Decide whether the photon is reflected or not based on the probability
If the surface is diffusive+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 diffuse and specular 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 is

\[
P_s = \frac{s_r + s_g + s_b}{d_r + d_g + d_b + s_r + s_g + s_b} \, P_r = P_r - P_d.
\]
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
• Colour bleeding
The power $P_{ref}$ of the reflected photon is:

- $P_{ref, sr} = \frac{P_{inc, r} Sr}{P_s}$
- $P_{ref, sg} = \frac{P_{inc, g} Sg}{P_s}$
- $P_{ref, sb} = \frac{P_{inc, b} Sb}{P_s}$

where $P_{inc}$ is the power of the incident photon, $P_s$ is the probability of specular reflection.

The above equation is for specular reflection, but so the same for diffusive reflection.
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

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

- We need an efficient data structure for retrieving photon maps when coloring the pixels
  - KD-tree
  - Spatial Hash
Saving photons: KD tree

- A hierarchical data structure for saving spatial data
- KD-tree:
  - dividing the samples at the median
  - The median sample becomes the parent node, and the larger data set form a right child tree, the smaller data set form a left child tree
  - Further subdivide the children trees
- Can efficiently find the neighbours when rendering the scene
Saving photons: Spatial Hashing

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

- 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
By 10,000 photons and 50 samples (left), and 500,000 photons and 500 samples (right)
Figure 20: Global photon map radiance estimates visualized directly using 100 photons (left) and 500 photons (right) in the radiance estimate.

Figure 21: Global photon map radiance estimates visualized directly using 500 photons and a disc to locate the photons. Notice the reduced false color bleeding at the edges.
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
Readings

- Realistic Image Synthesis Using Photon Mapping by Henrik Wann Jensen, AK Peters
- Global Illumination using Photon Maps (EGRW ‘96) Henrik Wann Jensen
- Caustics Generation by using Photon Mapping, Presentation by Michael Kaiser and Christian Finger
- A Practical Guide to Global Illumination using Photon Maps
  - Siggraph 2000 Course 8
http://www.cc.gatech.edu/~phlosoft/photon/