Realistic Image Synthesis

- Density Estimation and Photon Mapping -

Philipp Slusallek
Karol Myszkowski
Gurprit Singh
Overview

• **Today**
  – Density estimation background
  – Density estimation methods
  – Global illumination algorithms based on density estimation
  – Photon mapping

• **Next lecture**
  – Advanced photon mapping
Reading Materials: Density Estimation

Basic:
• P. Dutre, P. Bekaert, and K. Bala, Advanced Global Illumination, AK Peters 2003

Advanced:
• B.W. Silverman, Density Estimation for Statistics and Data analysis, Chapman and Hall, 1986
• M.P. Wand and M.C. Jones, Kernel Smoothing, Chapman and Hall, 1995
Reading Materia: Photon Mapping

Basic:

- Siggraph Asia 2013 course: State of the Art in Photon Density Estimation,
  - http://users-cs.au.dk/toshiya/starpm2013a/

Advanced:

- G.J. Ward and Rob Shakespeare, Rendering with Radiance, Morgan Kaufman, 1988
Photon Transport Simulation

• Instead of simulating the exact system, an analog system which is easier to simulate can be used
  – must retain all the important characteristics of the original system.

• Photons used in global illumination algorithms are simplified analogs of photons (light particles) in physics.

• The simplified photon characteristics
  – emitted by light sources and carry some energy,
  – travel in space obeying geometrical optics laws,
  – traced in space until they are completely absorbed due to reflections and refractions.

• Time factor is ignored
  – it is assumed that photons are moving instantaneously.
Global Illumination via Density Estimation

- A typical algorithm consists of three consecutive phases:
  1. photon tracing (continuous random walk)
  2. lighting reconstruction via density estimation,
  3. lighting storage and rendering.

- The lighting function is available implicitly as the density of photons hitting points
  - Reconstructing illumination out of collected photons is a density estimation problem.

- Various techniques are used to store/display lighting:
  - illumination maps (textures), meshing, or a direct density estimation at chosen sample points.

The surfaces in the room are depicted “unfolded” in the three figures on the right.
Random Walks

Continuous vs. Discrete Random Walks

Solves integral equations: radiosity or rendering equations

Solves linear systems: discretization error propagation
Sample point on light source with probability proportional to self-emitted radiosity:

\[ S(x) = \frac{E(x)}{\Phi_T} \]
Making the First Transition (1)

- No absorption at the origin
- Sample direction according to directional distribution of self-emitted radiance.

Diffuse emission: pdf is \( \frac{\cos(\theta_x)}{\pi} \)
Making the First Transition (2)

- Shoot ray along sampled direction.
- Geometric density factor: \( \cos(\theta_y) / r_{xy}^2 \)
Making the First Transition (3)

- Full transition density $T(x,y)$ is product:
  $$\frac{\cos(\theta_x)\cos(\theta_y)}{\pi r_{xy}^2}$$
Further Transitions

1. Absorption / survival test according to albedo $\rho(y)$

Full transition

$T(x,y) \cdot \rho(y)$
Further Transitions (2)

2. Sample direction according to brdf
Further Transitions (3)

3. Shoot ray
Further Transitions (4)

- Full transition density:
Once More …

1. Absorption / survival test
2. Sample direction according to brdf
3. Shoot ray
• Full transition density
And Yet Once More

1. Absorption / survival test
2. Sample direction according to brdf
3. Shoot ray
• **Full transition density**
End of Game

1. Absorption
Collision Density

- In general:

\[ D(X) = S(X) + \int D(Y)T(Y, X)\,dY \]

Path origins at X

Visits to X from elsewhere

- Random walk simulation yields points with density which is solution of second kind Fredholm integral equation
Collision Density for Radiosity

- Radiosity integral equation:

\[ B(x) = E(x) + \int_S B(y) \frac{\cos \theta_y}{\pi} \frac{\cos \theta_x}{r_{yx}^2} \text{vis}(y, x) \, \rho(x) \, dA_y \]

Source density should be normalized, \( S(x) = E(x)/\Phi_T \), but we’re almost there!
Collision Density for Radiosity

- Divide by total self-emitted power:

\[
\frac{B(x)}{\Phi_T} = \frac{E(x)}{\Phi_T} + \int_S \frac{B(y)}{\Phi_T} \frac{\cos \theta_y}{\pi} \frac{\cos \theta_x}{r_{yx}^2} \text{vis}(y, x) \rho(x) \, dA_y
\]
Collision Density for Radiosity

\[
\frac{B(x)}{\Phi_T} = \frac{E(x)}{\Phi_T} + \int_S \frac{B(y)}{\Phi_T} \frac{\cos \theta_y}{\pi} \frac{\cos \theta_x}{r_{yx}^2} \text{vis}(y, x) \rho(x) \, dA_y
\]
Collision Density for Radiosity

\[
\frac{B(x)}{\Phi_T} = \frac{E(x)}{\Phi_T} + \int_S \frac{B(y)}{\Phi_T} \left( \frac{\cos \theta_y}{\pi} \frac{\cos \theta_x}{r_{yx}} \text{vis}(y, x) \rho(x) \right) dA_y
\]

Source density \( S(x) \):
- 1. sample cosine distributed direction at \( y \)
- 2. shoot ray; ray hits \( x \)
- 3. survival test at \( x \)

Transition density \( T(y, x) \):

Collision Density for Radiosity

- Collision density proportional to radiosity

\[
\frac{B(x)}{\Phi_T} = \frac{E(x)}{\Phi_T} + \int_S \frac{B(y)}{\Phi_T} \frac{\cos \theta_y}{\pi} \frac{\cos \theta_x}{r_{yx}^2} \text{vis}(y, x) \rho(x) \ dA_y
\]

**Source density** $S(x)$

**Transition density** $T(y, x)$:

1. sample cosine distributed direction at $y$
2. shoot ray; ray hits $x$
3. survival test at $x$

\[
D(x) = \frac{B(x)}{\Phi_T}
\]
Sampled Points

- 1,000 paths
Sampled Points

- 10,000 paths
Sampled Points

- 100,000 paths
Sampled Points

- Collision density is related to radiosity!!
Photon Tracing: Summary

For each photon repeat the following steps:

1. Choose probabilistically:
   - the wavelength of the photon by sampling the *emission spectrum*,
   - the location of the photon on the emitter surface by sampling the *positional emission power distribution*,
   - the direction of propagation of the photon by sampling the *directional power distribution*.

2. Assign the energy to the photon and trace it until it is absorbed:
   - find the first object hit by the photon (use ray tracing),
   - decide on photon absorption or reflection by testing a random number against surface albedo,
   - if the photon is reflected:
     • assign a reflected direction to the photon by sampling BRDF,
     • update the outgoing photon flux and continue tracing the photon.
Handling Photon’s Power

Left: Photon with the power 12 watts is emitted by a patch on the floor
Right: the resulting power stored in the photon hit patches
Russian Roulette

- **Use the Russian roulette technique to avoid bias** (systematic error) in the solution. Perform test whether the photon survives with the probability \( p \) or is absorbed (with the probability \( 1 - p \)) and modify photon energy as follows:

\[
L = \begin{cases} 
\xi < p & \frac{\Phi}{p} \\
\text{otherwise} & 0 
\end{cases}
\]

\[
E(\Phi) = \Pr(\text{absorption}) \cdot 0 + \Pr(\text{survival}) \cdot \frac{\Phi}{p} = (1 - p) \cdot 0 + p \frac{\Phi}{p} = \Phi
\]

- On average the photon has the right energy. Since the photon energy is sampled in this procedure the solution variance increases (more noise) and converges to the correct solution as enough photons are used.

- **Example: Lambertian surface**
  - Draw a random number \([0,1]\) from the uniform distribution and compare with the albedo of surface hit by the photon.
  - If the photon “survives” the absorption test it is reflected and further traced carrying the same power:
    \[
    \frac{\Phi}{\rho} \cdot \rho = \Phi
    \]

Realistic Image Synthesis SS21 – Density Estimation and Photon Mapping

Karol Myszkowski
Density Estimation

- As a result of *continuous random walk* the lighting function is known *implicitly* as the density of photon collision points.

- The lighting function in *explicit* form must be reconstructed.
  - This is a classic *density estimation* problem where an estimate of the probability density function is constructed from the observed data points.

- **Basic approaches to density estimation**
  - Parametric: a family of distributions is known and only predefined parameters must be found, e.g. mean $\mu$ and variance $\sigma^2$ for the normal distribution.
  - Nonparametric: less rigid assumptions
    - This is the case for the global illumination problem.
Density Estimation Methods

- **Histograms:**
  - A domain is subdivided into bins (buckets) in which the number of photons and/or their accumulated energy is stored.

- **Naïve estimator:**
  - Counts the number of collisions in a bin centered at point $x$.

- **Kernel estimators:**
  - The density is estimated as spatially spread energy distributions around each photon collision point.

- **Nearest neighbor methods:**
  - The density at a point $x$ is estimated by dividing the number of the nearest neighbor photons $k$ (usually fixed) by the area of a region centered at $x$, in which these photons are collected.

- **Orthogonal series estimators:**
  - Higher order basis functions are used for lighting reconstruction in each bin (generalization of histograms)
Density Estimation

- For simplicity let us consider 1D case.
- **Notation:**

\[ f(x) \text{ and } \hat{f}(x) : \text{reconstructed pdf and its estimate at point } x \]

\[ X_i : \text{photon collision location} \]

\[ n : \text{the number of photon collisions} \]

\[ K(x) : \text{the kernel function} \]

\[ h : \text{the kernel radius (called also bandwidth or smoothing parameter)} \]

or the bin width for histogram estimators
Histograms

$$\hat{f}(x) = \frac{1}{nh} \times (\text{no. of } X_i \text{ in the same bin as } x)$$

or more general

$$\hat{f}(x) = \frac{1}{n} \times \frac{\text{no. of } X_i \text{ in the same bin as } x}{\text{width of bin containing } x}$$

- $\hat{f}(x)$ strongly depends on $h$, the choice of an origin and orientation of the grid of bins

![The same distribution but different histograms](image1.png)

![... different histograms](image2.png)

Fig. 2.1 Histograms of eruption lengths of Old Faithful geyser.
The Naïve Estimator

\[ w(y) = \begin{cases} 
0.5 & \text{if } |y| < 1 \\
0 & \text{otherwise}
\end{cases} \]

\[ \hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} w\left(\frac{x - X_i}{h}\right) \]

Effectively, this measures the number of photons falling into the interval \([x-h, x+h]\)

jumps at points \(X_i +/- h\)

Fig. 2.3 Naïve estimate constructed from Old Faithful geyser data, \(h = 0.25\).
The Kernel Estimator

- Generalization of the naïve estimator

\[
\int_{-\infty}^{\infty} K(x) \, dx = 1
\]

\[
\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)
\]

- \(\hat{f}(x)\) inherits all the continuity and differentiability properties of the kernel \(K\) (usually a symmetric pdf)
- Well studied mathematically
- For a fixed \(h\) might have tendency to excessive smoothing \(\hat{f}(x)\) regions with dense photon collisions \(X_i\) and leaving out visible noise in regions with low density of \(X_i\)
The Kernel Estimator

At first the kernel function is chosen (usually smooth and easy to compute functions are used). Then the kernel radius $h$ (the extent of the kernel support) is decided. It can be done globally for the whole surface or locally based on complexity of lighting distribution. Finally, the kernel function is centered at every photon location, and the photon energy is splatted (distributed) according to the kernel shape. The final lighting is estimated by summing splatted energy from all photons.
Fig. 2.8 *Kernel estimate for Old Faithful geyser data, window width 0.25.*
Bandwidth Sensitivity

- **Original bimodal density distribution**

- **Left: Kernel estimates for 200 simulated data points drawn from this bimodal density for kernel widths (a) 0.1, (b) 0.3, and (c) 0.6.**

Fig. 2.6 Kernel estimates for 200 simulated data points drawn from a bimodal density. Window widths: (a) 0.1; (b) 0.3; (c) 0.6.
The Nearest Neighbor Method

\[ \hat{f}(x) = \frac{k}{2nd_k(x)} \]

where \( d_k(x) \) is such a distance that in the interval \([x - d_k, x + d_k]\) \( k \) nearest collisions \( X_i \) is located

\[ d_1(x) \leq d_2(x) \leq \ldots \leq d_k(x) \]

- The amount of smoothing locally adapts to the density of photon collisions \( X_i \)
- Reconstructed \( \hat{f}(x) \) is not a pdf since it does not integrate to unity (problems with energy conservation)
- The generalized \( k \)-th nearest neighbor estimate

\[ \hat{f}(x) = \frac{1}{nd_k(x)} \sum_{i=1}^{n} K\left(\frac{x - X_i}{d_k(x)}\right) \]
The Nearest Neighbor Method

Fig. 2.10 Nearest neighbour estimate for Old Faithful geyser data, $k = 20$. 
Generalization to Higher Dimensions

- All discussed methods have similar properties when higher dimensional density estimation is considered.
- 2D is considered in the global illumination computation:
  - Histograms
    \[ \hat{f}(x) = \frac{1}{n} \times \frac{\text{no. of } X_i \text{ in the same bin as } x}{\text{area of bin containing } x} \]
  - Kernel methods
    \[
    \int_{-\infty}^{\infty} K(x) dx = 1
    \]
    \[
    \hat{f}(x) = \frac{1}{nh^2} \sum_{i=1}^{n} K\left\{ \frac{1}{h}(x - X_i) \right\}
    \]
Histogram Method

- Break surfaces in small elements. Count photons hitting each element:

\[
\hat{f}(x) = \frac{1}{n} \times \frac{\text{no. of } X_i \text{ in the same bin as } x}{\text{area of bin containing } x}
\]
Histogram Method
Orthogonal Series Density Estimation

- Linear, bi-linear, quadratic, cubic, … approximations

\( \hat{f}(x) \) can be reconstructed using \( K \) orthonormal basis functions \( \psi_v(x) \):

\[
\hat{f}(x) = \sum_{v=1}^{K} \hat{f}_v \psi_v(x)
\]

with the projection coefficients \( \hat{f}_v \):

\[
\hat{f}_v = \frac{1}{n} \sum_{i=1}^{n} \psi_v(X_i)
\]

\[
\int_{-\infty}^{+\infty} \psi_\mu(x) \psi_v(x) dx = \delta_{\mu v} = \begin{cases} 
1 & \mu = v \\
0 & \text{otherwise}
\end{cases}
\]
Kernel Density Estimation

- Place finite-width density kernel at each sample point

\[ \int_{-\infty}^{\infty} K(x) \, dx = 1 \]

\[ \hat{f}(x) = \frac{1}{nh^2} \sum_{i=1}^{n} K\left\{ \frac{1}{h}(x - X_i) \right\} \]
Cylindrical Kernel
Gaussian Kernel
Comparison of Density Estimation Methods

- **Problems with energy conservation**
  - Nearest neighbor method

- **Discontinuities in reconstructed density**
  - Histogram, naïve, and nearest neighbor methods

- **Complexity as a function of the photon number** $n$
  - $O(n)$
    - Histogram - fast
    - Naïve and kernel estimators – average
  - $O(n \log n)$
    - Nearest neighbor method – slow

- **Adaptability to local density fluctuations**
  - Histogram, naïve and kernel estimators – poor
  - Nearest neighbor method – good
Bias and Random Error

• The error between the illumination function $f(x)$ and its estimate $\hat{f}(x)$ can be expressed as:

$$\phi(x) = f(x) - \hat{f}(x) = \underbrace{f(x) - E(\hat{f}(x))}_{\phi_{\text{bias}}(x)} + \underbrace{E(\hat{f}(x)) - \hat{f}(x)}_{\phi_{\text{random}}(x)}$$

$$E\hat{f}(x) = E\left[ \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \right] = \frac{1}{nh} \sum_{i=1}^{n} E\left[ K\left(\frac{x - X_i}{h}\right) \right] = \frac{1}{h} \int K\left(\frac{x - y}{h}\right) f(y) dy$$

• The bias is a smoothed version of the true density $f$
  - Bias = convolution of $f$ with the kernel $K$ scaled by the kernel size $h$

$$\hat{f}(x) = \text{smoothed version of true density} + \text{random error}$$

• The bias does not depend directly on the number of photons
  - Bias cannot be eliminated just by shooting more photons!
Bias and Random Error

Noise $\sim \frac{1}{nr^2}$

Bias $\sim r^2$

Noise vanishes

Bias increases
Optimal Kernel

• The optimal kernel can be found by minimizing \( \text{noise}&\text{bias} \)
  
  – Epanechnikov

\[
\frac{3}{4 \sqrt{5}} \left(1 - \frac{1}{5} t^2 \right) \quad \text{for} \quad |t| < \sqrt{5}
\]

• Even simple kernels such as gaussian or cylindrical lead to only small increase of density estimation error.
Main problems:

• The parameters of density estimation are usually decided \textit{globally} for the whole scene or surfaces
  – This may result in uncontrolled smoothing/noise for complex illumination patterns.

• Local estimate of the lighting reconstruction error would be useful to find optimal bandwidth for a given scene region.
Adaptive vs. Fixed Bandwidth $h$

<table>
<thead>
<tr>
<th>Photons/texel</th>
<th>Adaptive</th>
<th>$\varepsilon_{\text{adapt}}$</th>
<th>Fixed</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>10</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>100</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>1000</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
<td><img src="image15.png" alt="Image" /></td>
<td><img src="image16.png" alt="Image" /></td>
</tr>
</tbody>
</table>
## Adaptive vs. Fixed Bandwidth $h$

<table>
<thead>
<tr>
<th>Photons/texel</th>
<th>Adaptive</th>
<th>$\varepsilon_{\text{adapt}}$</th>
<th>Fixed</th>
<th>$\varepsilon$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image" alt="Adaptive 1 photons" /></td>
<td><img src="image" alt="Adaptive 1 photons" /></td>
<td><img src="image" alt="Fixed 1 photons" /></td>
<td><img src="image" alt="Fixed 1 photons" /></td>
</tr>
<tr>
<td>10</td>
<td><img src="image" alt="Adaptive 10 photons" /></td>
<td><img src="image" alt="Adaptive 10 photons" /></td>
<td><img src="image" alt="Fixed 10 photons" /></td>
<td><img src="image" alt="Fixed 10 photons" /></td>
</tr>
<tr>
<td>100</td>
<td><img src="image" alt="Adaptive 100 photons" /></td>
<td><img src="image" alt="Adaptive 100 photons" /></td>
<td><img src="image" alt="Fixed 100 photons" /></td>
<td><img src="image" alt="Fixed 100 photons" /></td>
</tr>
<tr>
<td>1000</td>
<td><img src="image" alt="Adaptive 1000 photons" /></td>
<td><img src="image" alt="Adaptive 1000 photons" /></td>
<td><img src="image" alt="Fixed 1000 photons" /></td>
<td><img src="image" alt="Fixed 1000 photons" /></td>
</tr>
</tbody>
</table>
Bias Compensation for the Nearest Neighbor Method

• **Algorithm**
  - For each point \( x \) perform estimate of illumination \( \hat{f}(x,1),...,\hat{f}(x,N_{\min}) \) for the number of nearest photons ranging from 1,\(...,N_{\min} \)
  - Compute the expected value and variance of \( f(x,j) \) using some weighting function \( w(j) \)
    \[
    \mu[\hat{f}(x,N_P)] = \sum_{j=1}^{N_{\min}} w(j) \cdot \hat{f}(x,j)
    \]
    \[
    \sigma^2[\hat{f}(x,N_P)] = \mu[\hat{f}^2(x,N_P)] - \mu^2[\hat{f}(x,N_P)]
    \]
  - Recursively split the interval \([N_{\min},N_{\max}]\) at \(N_{\text{mid}}=(N_{\max}-N_{\min})/2\) and decide which interval to choose based on a density estimate:
    \[
    \hat{f}(x,N_{\text{mid}}) \text{ using } N_{\text{mid}} \text{ photons}
    \]
    \[
    \epsilon[\hat{f}(x,N_{\text{mid}})] = \hat{f}(x,N_{\text{mid}}) - \mu[\hat{f}(x,N_P)]
    \]
    with the probability \( p \) that \( \epsilon \) is attributed to noise:
    \[
    p = e^{-\epsilon^2[\hat{f}(x,N_{\text{mid}})]/2\sigma^2[\hat{f}(x,N_P)]}
    \]
    where based on the central limit theorem \( p \) is the likelihood that \( \epsilon \) is due to noise
procedure biascomp($\bar{x}, min, max$) 
\begin{align*}
N_{\text{min}} &= min \\
N_{\text{max}} &= max \\
gather N_{\text{max}} \text{ photons} \\
\text{for } j = 1 \text{ to } N_{\text{min}} \text{ do} \\
& \quad \text{partition}(j, j+1, N_{\text{max}}) \\
& \quad \text{get irradiance estimate } \hat{f}(\bar{x}, j) \text{ for } j \text{ closest photons} \\
& \quad \text{include } \hat{f}(\bar{x}, j) \text{ in average } \mu \\
\text{end for} \\
evaluate \hat{\sigma}^2 \\
\text{while } N_{\text{min}} < N_{\text{max}} \text{ do} \\
N_{\text{mid}} &= (N_{\text{min}} + N_{\text{max}}) / 2 \\
\text{partition}(N_{\text{min}}, N_{\text{mid}}, N_{\text{max}}) \\
\text{get irradiance estimate } \hat{f}(\bar{x}, N_{\text{mid}}) \text{ for } N_{\text{mid}} \text{ closest photons} \\
\epsilon &= \hat{f}(\bar{x}, N_{\text{mid}}) - \mu \\
p &= \exp(-\epsilon^2 / 2\hat{\sigma}^2) \\
& \quad \text{if random } \xi \in [0, 1] < p \text{ then } \{ \epsilon \text{ probably noise, recurse in } [N_{\text{mid}}, N_{\text{max}}] \} \\
& \quad \quad \text{include } \hat{f}(\bar{x}, N_{\text{mid}}) \text{ in average } \mu \\
& \quad \quad \text{update } \hat{\sigma}^2 \\
& \quad \quad N_{\text{min}} = N_{\text{mid}} \\
& \quad \text{else } \{ \epsilon \text{ probably bias, recurse in } [N_{\text{min}}, N_{\text{mid}}] \} \\
& \quad \quad N_{\text{max}} = N_{\text{mid}} \\
& \text{end if} \\
\text{end while} \\
\text{return } \hat{f}(\bar{x}, N_{\text{mid}}) \)
Bias Compensation for the Nearest Neighbor Method

Bias Case Study: Highlight Cross-Section

- 2500 photons
- 50-2500 photons

Irradiance [W/m²]

Realistic Image Synthesis SS21 – Density Estimation and Photon Mapping
Roland Schregle
Bias Compensation for the Nearest Neighbor Method

Photon density

Bias compensation with adaptive bandwidth for 50-5,000 nearest photons

Fixed bandwidth for 5,000 nearest photons
Bias Compensation for the Nearest Neighbor Method

Fixed bandwidth with 50 nearest photons

Fixed bandwidth with 500 nearest photons

Adaptive bandwidth with 50-500 nearest photons
Bias Compensation for the Nearest Neighbor Method

Fixed bandwidth with 50 nearest photons

Fixed bandwidth with 2,000 nearest photons

Adaptive bandwidth with 50-2,000 nearest photons
Photon Mapping: Motivation

- Designed as an alternative for finite element radiosity and density estimation methods that require scene meshing
  - Moderate memory requirements
  - Handling all types of geometry, e.g., fractal surfaces
  - Adaptability to local lighting distribution in the scene

- Efficiency in combating perceivable noise inherent for Monte Carlo ray tracing methods
  - At expense of user controlled bias in the solution

- Simulation of all global illumination effects
  - *Consistent estimator* of the rendering equation

\[ E\{X\} \neq \int \ldots \text{ but } \lim_{n \to \infty} E\{X\} \to \int \ldots \]
Photon Mapping: Two-pass Method

- **Photon tracing**
  - Similar to other density estimation techniques
  - Photon hit points registered in the photon map data structure
  - *Projection maps* – maps of geometry as seen from the light source are used to guide photons towards the scene

- **Rendering**
  - Distribution ray tracing used for rendering
  - Lighting function reconstructed at ray-object intersection points using the photon map data structure
    - Photon density estimation using the nearest neighbor method
Photon Tracing

- Photon registration only for diffuse and glossy surfaces
- Specular component reconstructed explicitly during rendering
  - too many photons would be required to reconstruct specular lighting based on the photon map
- Two photon maps
  - Global map
    - Low resolution,
    - Rendered indirectly, through final gathering procedure
  - Caustic map
    - High resolution
    - Rendered directly
Caustic Photon Map

- Projection maps also used to reinforce shooting more photons towards specular surfaces
- This will not work for caustics resulting from strong secondary light sources

Global photon map

Caustic photon map
Photon-map Data Structure

- **Problem:**
  - efficient searching for nearest neighbor photons in the 3D space

- **Data structure choice: kd-tree**
  - Each node contains a photon and pointers to two subtrees
    - Pointers to subtrees are not necessary for heap data structure
      - Element $i$ has children $2i$ and $2i+1$
  - For a photon map with $n$ photons
    - The worst case searching time for one photon is $O(\log n)$ for a balanced tree and $O(n)$ for an unbalanced tree
    - The average search time for $k$ photons is $O(k + \log n)$

- **Photon representation (20 bytes)**
  ```
  struct photon {
    float x,y,z; // position: 3 x 32 bit floats
    char power[4]; // stored in Ward’s shared exponent RGB-format
    char phi, theta; // compressed incident direction
    short flags; // used in kd-tree
  }
  ```
  - Ward’s format may lead to ~0.5% bias (empty diffuse sphere test)
Density Estimation

- Derivation

\[
L\ (x, \omega_o) = \int_{\Omega_+} f_r(\omega_i, x, \omega_o) L(x, \omega_i) \cos \theta_i \ d\omega_i =
\]

\[
\int_{\Omega_+} f_r(\omega_i, x, \omega_o) \frac{d^2 \Phi(x, \omega_i)}{d\omega_i dA \cos \theta_i} \cos \theta_i \ d\omega_i \approx
\]

\[
\sum_{p=1}^{k} f_r(\omega_i, x, \omega_o) \frac{\Delta \Phi_p(x, \omega_i)}{\pi r^2}
\]
Bias Types

- **Proximity Bias**: Blurred details in reconstructed lighting
  - Filtering with weights increasing for closer photons can help
- **Boundary Bias**: Overestimated area results in darkening near edges
- **Topological Bias**: Underestimated area results in excessive radiance estimate
- **Occlusion Bias**: Light leaks
  - Only photons hitting surfaces with similar normals should be considered
Direct Visualization of the Caustic Map

Caustic on a glossy surface (340,000 photons)
Direct Visualization of the Caustic Map

Realistic Image Synthesis SS21 – Density Estimation and Photon Mapping

Henrik Wann Jensen
Direct Visualization of the Global Map

- 200,000 photons in the photon map
- Radiance estimate using:
  - 50 nearest photons
  - 500 nearest photons
Practical Rendering Algorithm (1)

- **Global photon map contains all illumination types**
  - Direct illumination + indirect illumination + caustics
- **Caustic photon map contains just caustics**

![Global photon map](image1)

![Caustic photon map](image2)
Practical Rendering Algorithm (2)

1) Direct lighting computation
2) Mirror reflection
3) Caustic computation
4) Indirect lighting through final gathering
Practical Rendering Algorithm (3)

Reflection Equation Decomposition

\[
f_r(\omega_i, x, \omega_o) = f_{r,\text{spec}}(\omega_i, x, \omega_o) + f_{r,\text{diffuse}}(\omega_i, x, \omega_o)
\]

\[
L(x, \omega_i) = L_{\text{direct}}(x, \omega_i) + L_{\text{caustic}}(x, \omega_i) + L_{\text{indirect}}(x, \omega_i)
\]

\[
L(x, \omega_o) = \int_{\Omega_+} f_r(\omega_i, x, \omega_o) L(x, \omega_i) \cos \theta_i \, d\omega_i =
\]

1) \[
\int_{\Omega_+} f_r(\omega_i, x, \omega_o) L_{\text{direct}}(x, \omega_i) \cos \theta_i \, d\omega_i +
\]

2) \[
\int_{\Omega_+} f_{r,\text{spec}}(\omega_i, x, \omega_o)(L_{\text{caustic}}(x, \omega_i) + L_{\text{indirect}}(x, \omega_i)) \cos \theta_i \, d\omega_i +
\]

3) \[
\int_{\Omega_+} f_{r,\text{diffuse}}(\omega_i, x, \omega_o) L_{\text{caustic}}(x, \omega_i) \cos \theta_i \, d\omega_i +
\]

4) \[
\int_{\Omega_+} f_{r,\text{diffuse}}(\omega_i, x, \omega_o) L_{\text{indirect}}(x, \omega_i) \cos \theta_i \, d\omega_i
\]
Results

200,000 and 50,000 photons in the global and caustic maps

Ray traced image (direct lighting only)  Full global illumination
Results: Fractal Box

200,000 and 50,000 photons in the global and caustic maps
Results: Box with Water

500,000 photons in both the global and caustic maps, 100 nearest photons used in the radiance estimate
Results

The natural lighting (skylight and sunlight) simulation at various lighting conditions
Final Gathering

- Final gathering performed for each pixel representing Lambertian surfaces is very costly
  - 200 – 5,000 sample rays must be considered
  - Samples are stratified over the hemisphere

\[
\int_{0}^{2\pi/2} \int_{0}^{\pi} L(x, \theta, \phi) \cos \theta \sin \theta \, d\theta \, d\phi \approx \frac{\pi}{MN} \sum_{m=0}^{M} \sum_{n=0}^{N} L(\theta_m, \phi_n)
\]

where \( M \cdot N \) are the number of strata and

\[
\theta_m = \arcsin \left( \sqrt{\frac{m - \xi_1}{M}} \right) \quad \text{and} \quad \phi_n = 2\pi \frac{n - \xi_2}{N}
\]

- Radiance values collected by sample rays obtained directly from the global photon map
  - Problem: for points located near some other objects, e.g., in the room corner, very similar density estimates would be obtained due to a very small distance – secondary final gather is needed for such points.
Final Gathering: Indirect Lighting Sampling

Figure 2.3: (a) A hemispherical fisheye view of a conference room scene from a point on the floor. (b) Incoming radiance samples generated by our stratified sampling strategy. The light sources appear dark in (b) since direct light emission from the intersected objects is ignored in indirect illumination sampling. (Images courtesy of Greg Ward.)
Irradiance Caching

• **Observation:**
  – Indirect lighting computed using final gathering usually changes slowly (caustics are processed separately)

• **Idea:**
  – Final gathering results can be cached (in some 3D data structure, e.g., octree) and re-used for neighboring pixels

---

**Algorithm**  Lazy irradiance evaluation used in irradiance caching.

```plaintext
function IrradianceCaching(p, n)
    if one or more irradiance values can be used for interpolation at p then
        return irradiance interpolated from the cached values.
    else
        Compute new irradiance value and gradients by hemisphere sampling.
        Store the value with gradients as a new record in the cache.
        return the new irradiance value.
    end if
end function
```
Irradiance Caching

- **Irradiance change estimate:**
  - \( E(x_i, n_i) \): irradiance at point \( x_i \) with normal \( n_i \)

  \[
  \text{Error estimate: } \varepsilon(x, \bar{n}) = E_i(x_i) \left( \frac{4 \|x - x_i\|}{\pi R_i} + \sqrt{2 - 2\bar{n} \cdot \bar{n}_i} \right)
  \]

  - \( R_i \): harmonic mean distance at \( x_i \)

  \[
  R_i = \frac{1}{\frac{1}{M \cdot N} \sum_{m=0}^{M} \sum_{n=0}^{N} \frac{1}{r_{m,n}}} \quad \text{where } r_{m,n} \text{ – distance to the nearest object}
  \]

- **Irradiance interpolation:**

  \[
  E(x, \bar{n}) \approx \frac{\sum_{i,w_i > \frac{1}{a}} w_i(x, \bar{n}) E_i(x_i)}{\sum_{i,w_i > \frac{1}{a}} w_i(x, \bar{n})}
  \]

  \[
  w_i(x, \bar{n}) = \frac{1}{\frac{\|x - x_i\|}{R_i} + \sqrt{1 - \bar{n} \cdot \bar{n}_i}}
  \]
Irradiance Cache Positions

- Final gathering: 1,000 sample rays
- Irradiance cache: $w > 10$
Irradiance Cache Positions

- Final gathering: 1,000 sample rays
- Irradiance cache: $w > 20$
Irradiance Cache Gradients

\[
E(\vec{P}) = \sum_{S} \frac{w_i(\vec{P}) \left[ E_i + (\hat{n}_i \times \hat{n}) \cdot \vec{\nabla}_r E_i + (\vec{P} - \vec{P}_i) \cdot \vec{\nabla}_t E_i \right]}{\sum_{S} w_i(\vec{P})}
\]

- Weights \( w_i(\vec{P}) \) the same as in no gradient case

- Gradients modify \( E_i \) used for interpolation
Irradiance Cache Gradients

without gradient

with gradient

Irradiance Interpolation Error

\( x = 6.875 \)

Relative Error (%) vs Y Position
Irradiance Cache Data Structure

```c
struct indirect_irradiance_value {
    float P[3];    /* position in space */
    float N[3];    /* normal direction */
    float R;       /* validity radius */
    COLOR E;       /* computed irradiance value */
    float dP[3];   /* gradient wrt. position */
    float dN[3];   /* gradient wrt. direction */
};
```
Irradiance Caching Example
Irradiance Caching Example
Optimizations

• Faster final gathering
  – Precomputed irradiance stored at photon locations
  – Only the nearest photon must be found instead of density estimation computation
  – Reported speedup up 6-10 times

• Photon density control in the map
  – In bright regions with slowly changing illumination the power of redundant photons is distributed to their neighbors
  – Photon number reduced 2-4 times for the same image quality

• Photon stratification
  – Deterministic quasi-Monte Carlo (QMC) sequences are used
  – Low discrepancy: try to maximize the local distance between photon paths (multi-dimensional stratification)
  – For caustics QMC sequences may add some noticeable patterns

• Importance sampling: BRDF and incoming flux
Optimizations

- **Avoiding energy overestimation**
  - Gather one extra photon and compute the average distances of the two furthest photons
  - The extra photon (located beyond this average distance) is discarded in density estimation
  - Without this optimization energy overestimation of 1% and higher was observed for analytical tests (e.g., empty diffuse sphere).

- **Be careful with your choice of random number generator because it may lead to some bias too**
  - System V-style `drand48()` performed slightly better than `erand48()` and much better than BSD-style `random()`.

Industrial Products

– Brazil Rendering System (www.splutterfish.com)
Industrial Products

- finalRender (www.finalrender.com)
Industrial Products

- Mental Ray (www.mentalray.com)
Industrial Products

- Photorealistic RenderMan (www.pixar.com)
Industrial Products

- SoftImage|XSI (www.softimage.com)
Industrial Products

- VirtuaLight (www.3dvirtualight.com)
Industrial Products

– VRay (www.vrayrender.com)
Acknowledgements

• I would like to thank Philippe Bekaert, Henrik Wann Jensen, Jaroslav Krivanek, and Roland Schregle for sharing with me some slides and images that I used during this lecture.