CS580:
MC Ray Tracing:
Part III, Acceleration and Biased Tech.

Sung-Eui Yoon
(윤성의)

Course URL:
http://sglab.kaist.ac.kr/~sungeui/GCG

KAIST

Class Objectives:

- **Extensions to the basic MC path tracer**
  - **Bidirectional path tracer**
  - **Metropolis sampling**

- **Biased techniques**
  - **Irradiance caching**
  - **Photon mapping**

KAIST

General GI Algorithm

- **Design path generators**

- **Path generators determine efficiency of GI algorithm**

- **Black boxes**
  - **Evaluate BRDF, ray intersection, visibility evaluations, etc**

Other Rendering Techniques

- **Bidirectional path tracing**

- **Metropolis**

- **Biased techniques**
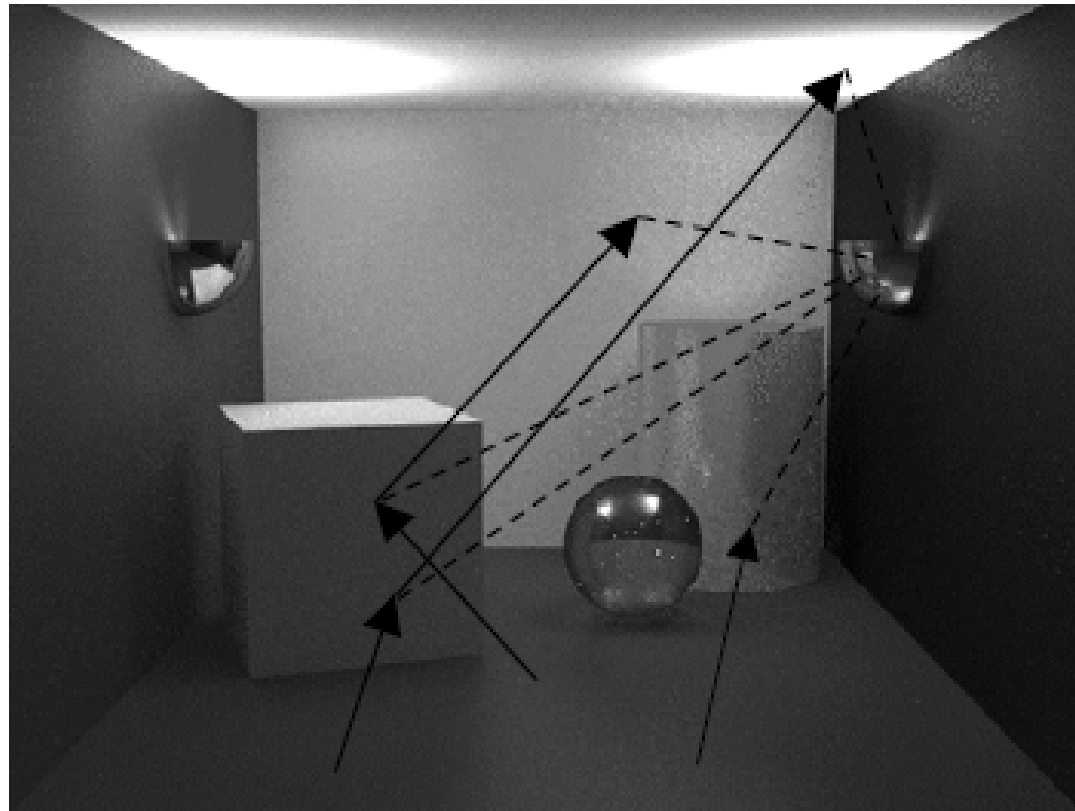 - **Irradiance caching**
  - **Photon mapping**

KAIST

Stochastic ray tracing: limitations

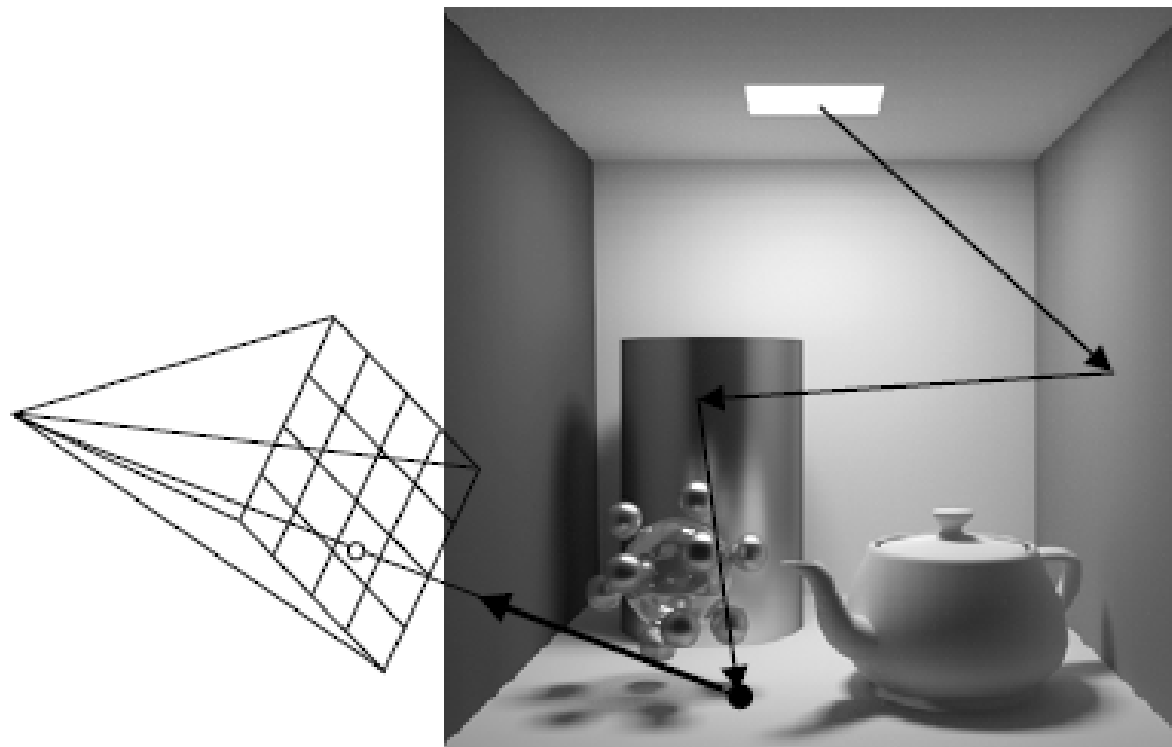- Generate a path from the eye to the light source
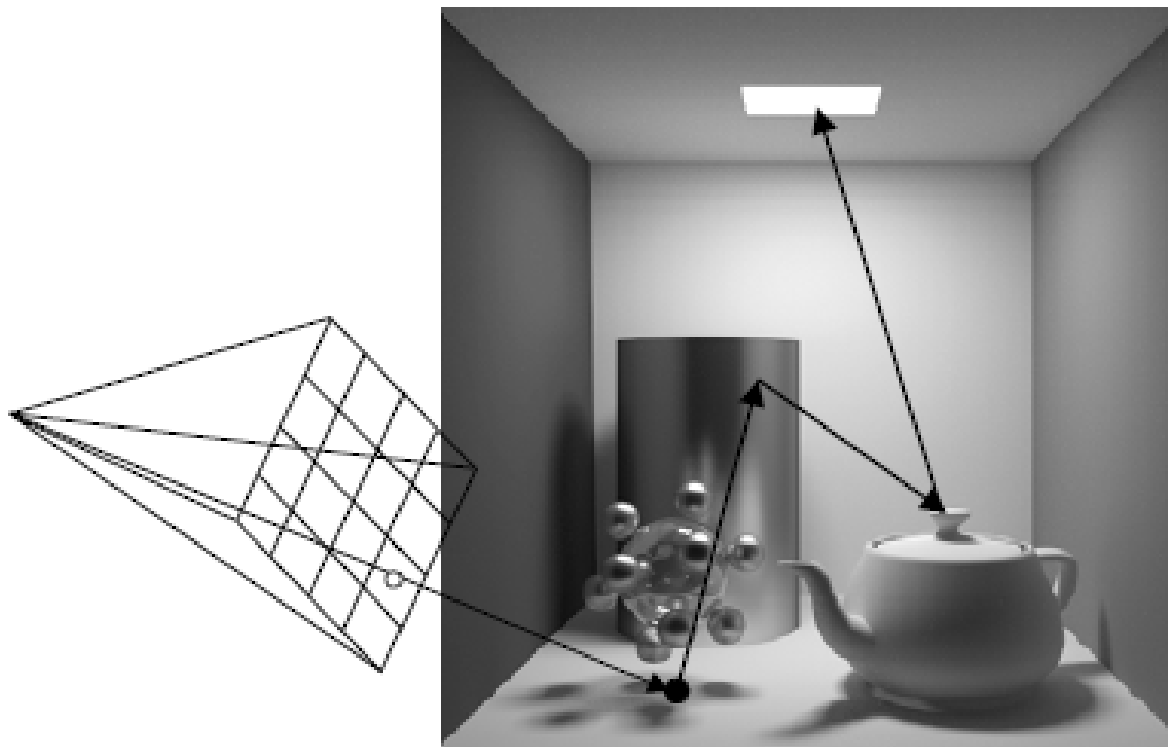
When does it not work?

- Scenes in which indirect lighting dominates
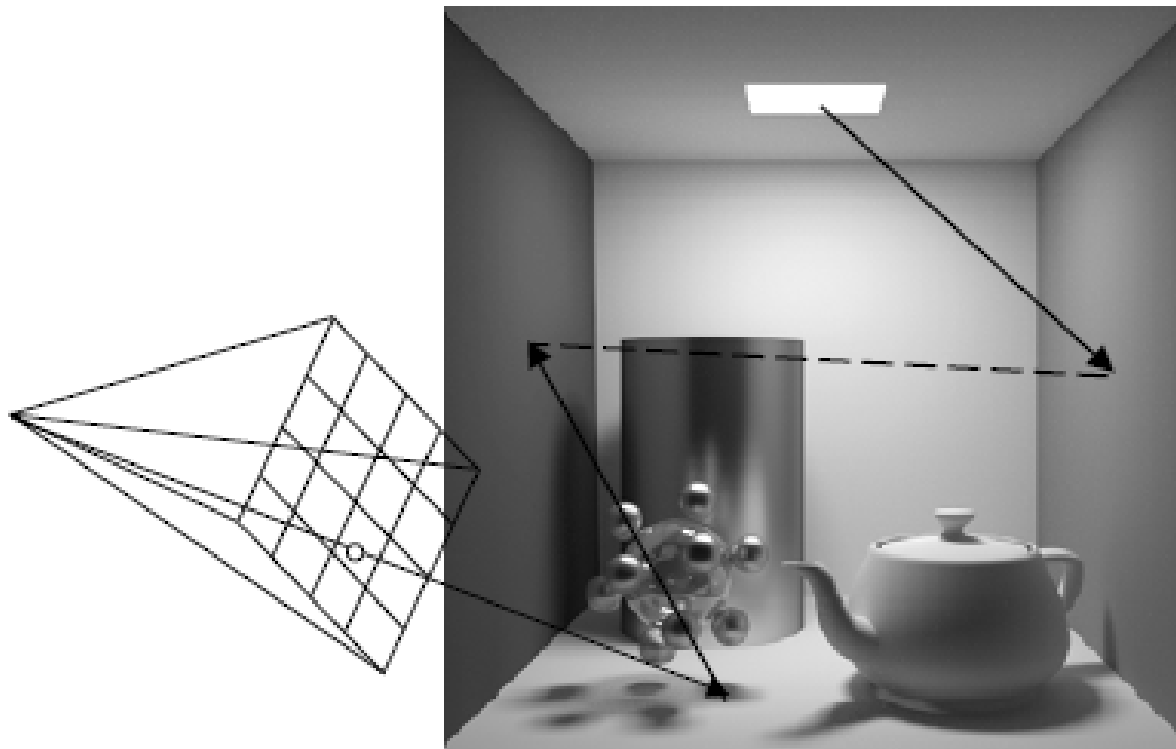
Bidirectional Path Tracing

- So … we can generate paths starting from the light sources!

- Shoot ray to camera to see what pixels get contributions
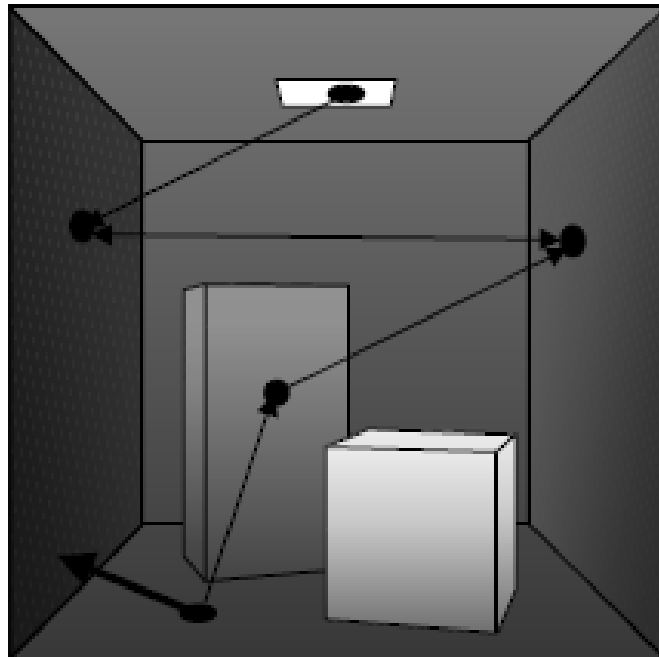
Bidirectional Path Tracing

- Or paths generated from both camera and source at the same time ...!

- Connect endpoints to compute final contribution
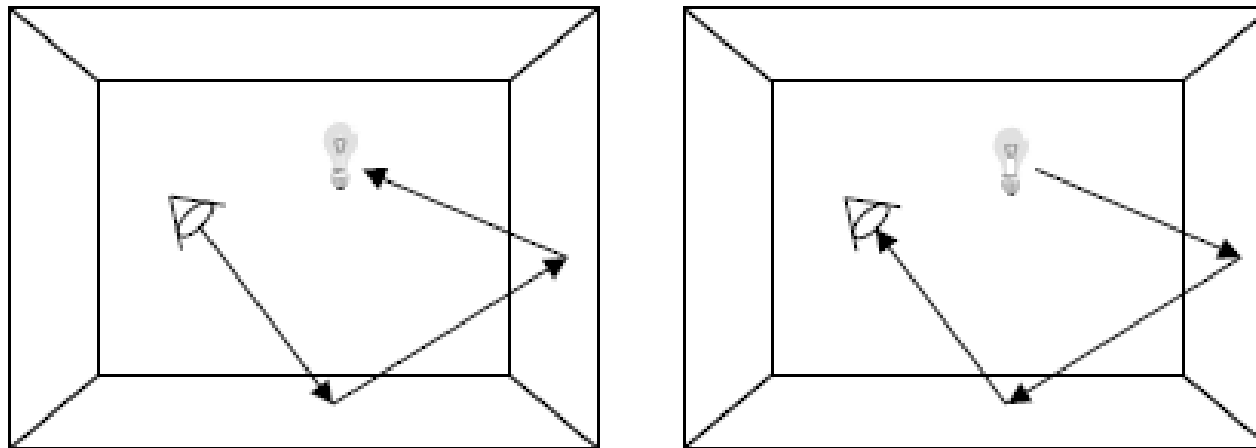
Complex path generators

- Bidirectional ray tracing
  - shoot a path from light source
  - shoot a path from receiver
  - connect end points

Why? BRDF - Reciprocity

- Direction in which path is generated, is not important: Reciprocity

$$f(\Psi \to \Theta) = f(\Theta \to \Psi) = f(\Psi \leftrightarrow \Theta)$$

- Algorithms:
  - trace rays from the eye to the light source
  - trace rays from light source to eye
  - any combination of the above

Bidirectional ray tracing

- Parameters
  - eye path length = 0: shooting from source
  - light path length = 0: gathering at receiver

- When useful?
  - Light sources difficult to reach
  - Specific brdf evaluations (e.g., caustics)

Other Rendering Techniques

- **Metropolis**

- **Biased techniques**
  - **Irradiance caching**
  - **Photon mapping**

KAIST

Metropolis

- **Based on Metropolis sampling (1950's)**
  - **Introduced by Veach and Guibas to CG**

- **Deals with hard to find light paths**
  - **Robust**
- **Hairy math, but it works**
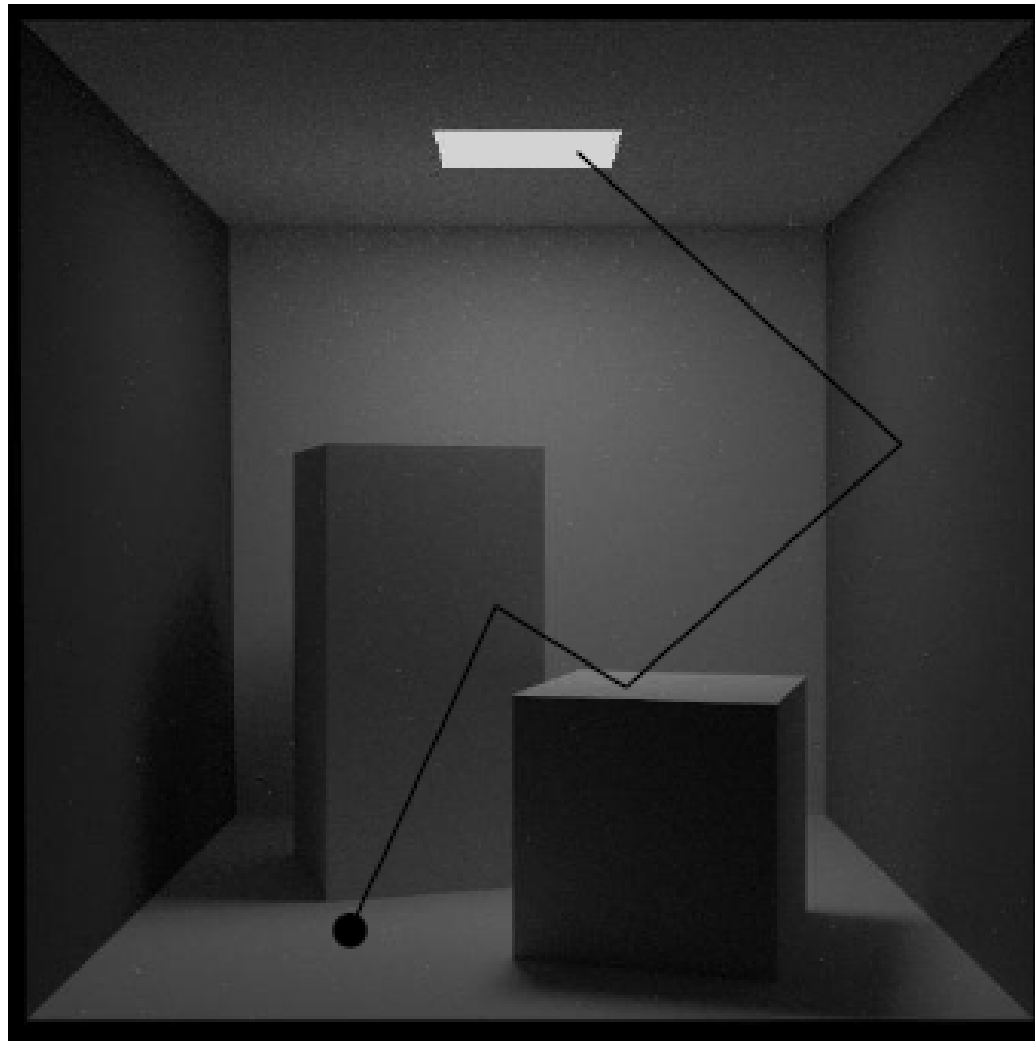 - **Not that easy to implement**

KAIST

Metropolis

- **Generate paths**

- **Once a valid path is found, mutate it to generate new valid paths**

- **Advantages:**
  - **Path re-use**
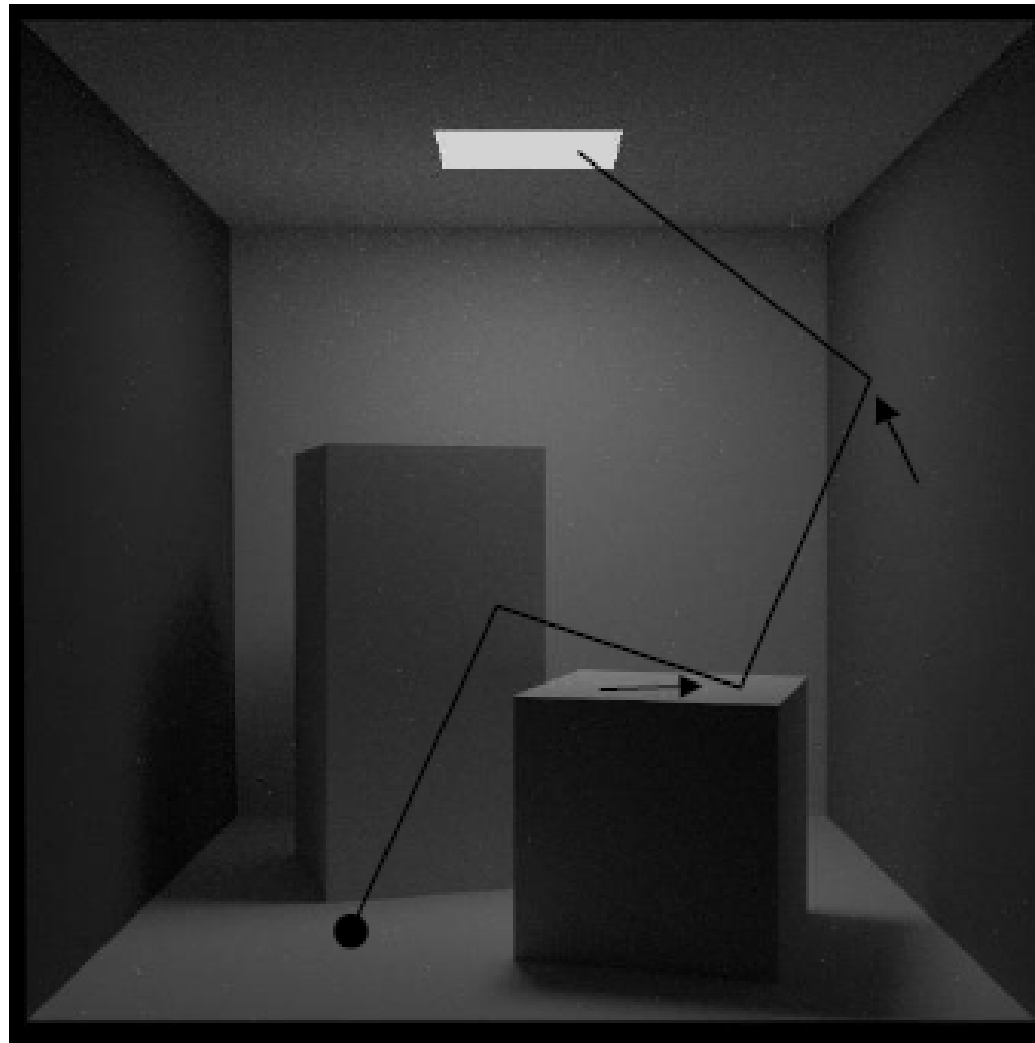 - **Local exploration: found hard-to-find light distribution, mutate to find other such paths**

KAIST
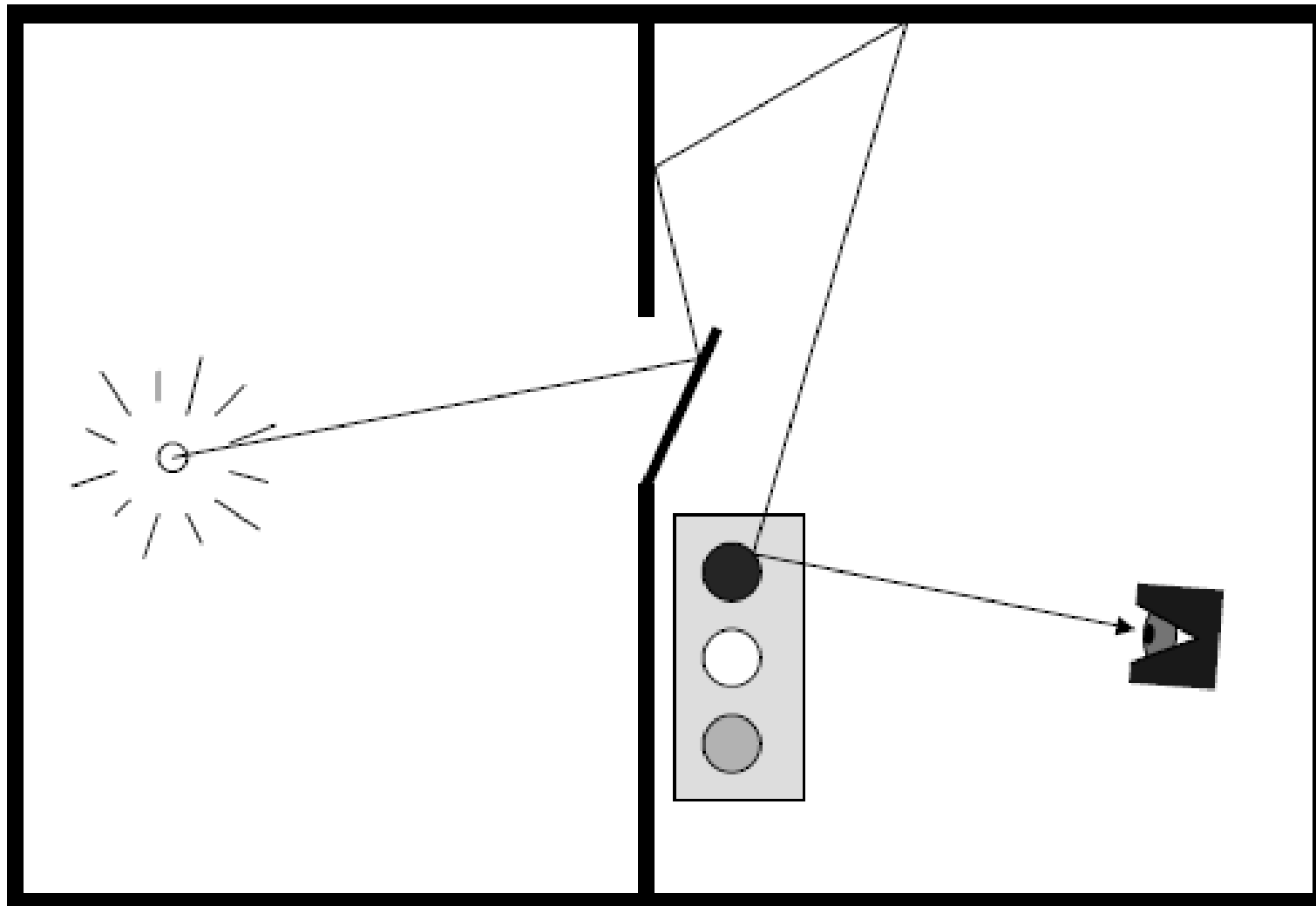
Metropolis
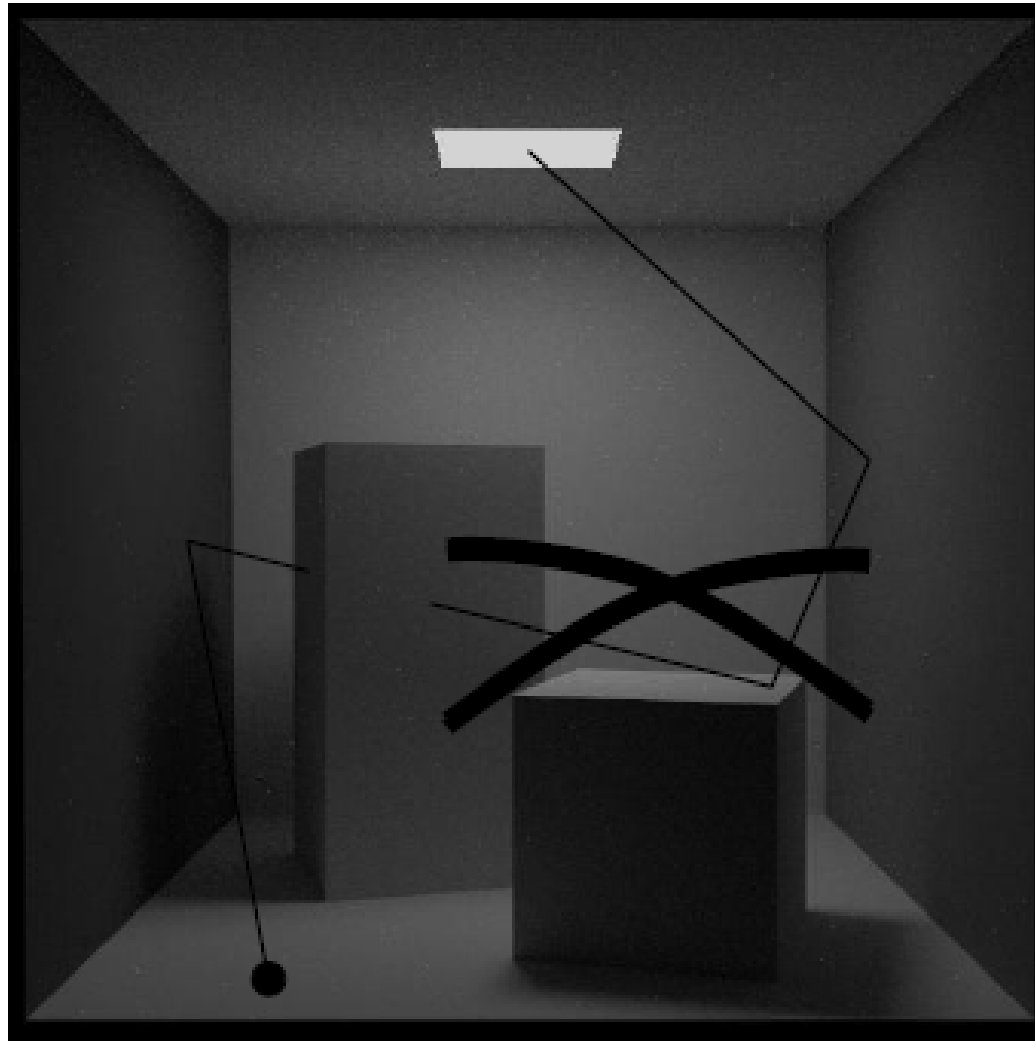
Metropolis

valid path

Metropolis

small
perturbations

Metropolis

Accept
mutations
based on
energy
transport

© Kavita Bala, Computer Science, Cornell University

Metropolis

- **Advantages**
  - **Robust**
  - **Good for hard to find light paths**

- **Disadvantage**
  - **Slow convergence for many important paths**
  - **Tricky to implement and get right**

Unbiased vs. Consistent

- **Unbiased**
  - **No systematic error**
  - $E[I_{estimator}] = I$
  - **Better results with larger N**

- **Consistent**
  - **Converges to correct results with more samples**
  - $E[I_{estimator}] = I + \varepsilon$, where $\lim_{n \to \infty} \varepsilon = 0$

Biased Methods

- **MC methods**
  - **Too noisy and slow**
  - **Nose is objectionable**

- **Biased methods: store information (caching)**
  - **Irradiance caching**
 - **Photon mapping**

KAIST

Irradiance Caching

- **Introduced by Greg Ward 1988**
- **Implemented in RADIANCE**
  - **Public-domain software**

- **Exploits smoothness of irradiance**
  - **Cache and interpolate irradiance estimates**

KAIST

Irradiance Caching
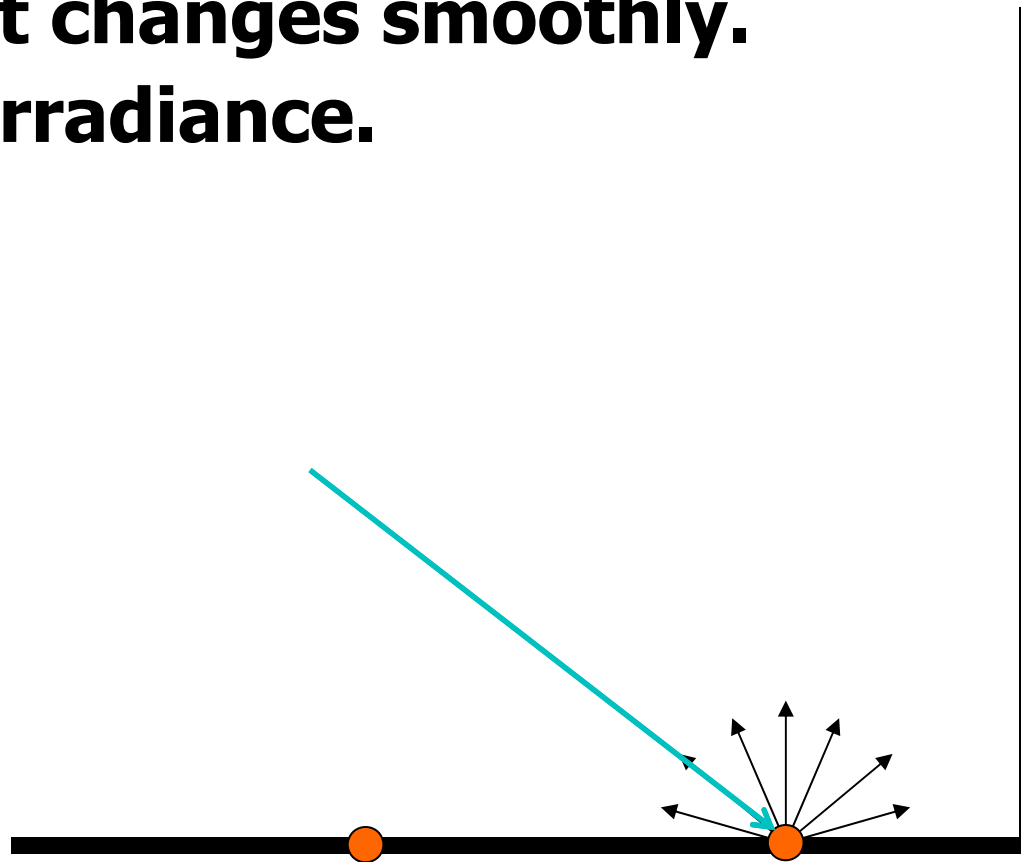
- **Indirect changes smoothly.**

From Wang's slides

KAIST

Irradiance Caching

- **Indirect changes smoothly.**
- **Cache irradiance.**

From Wang's slides

KAIST

Irradiance Caching

- **Indirect changes smoothly.**
- **Cache irradiance.**

From Wang's slides
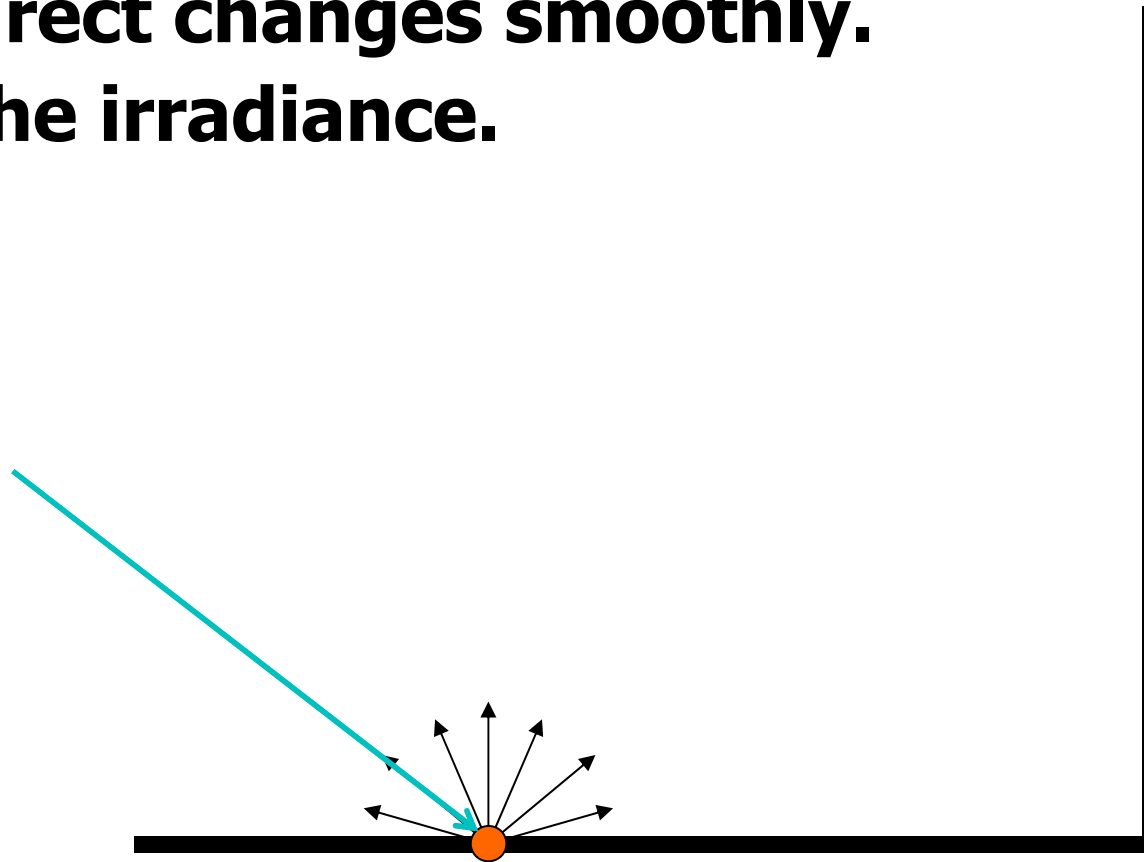
Irradiance Caching

- **Indirect changes smoothly.**
- **Cache irradiance.**
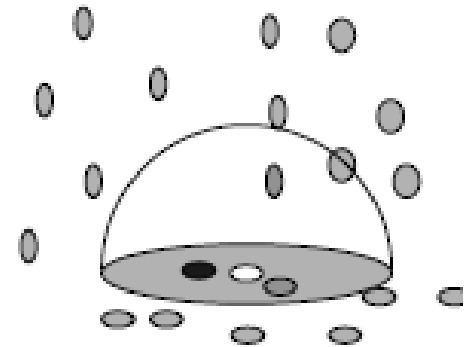- **Interpolate them.**

From Wang's slides
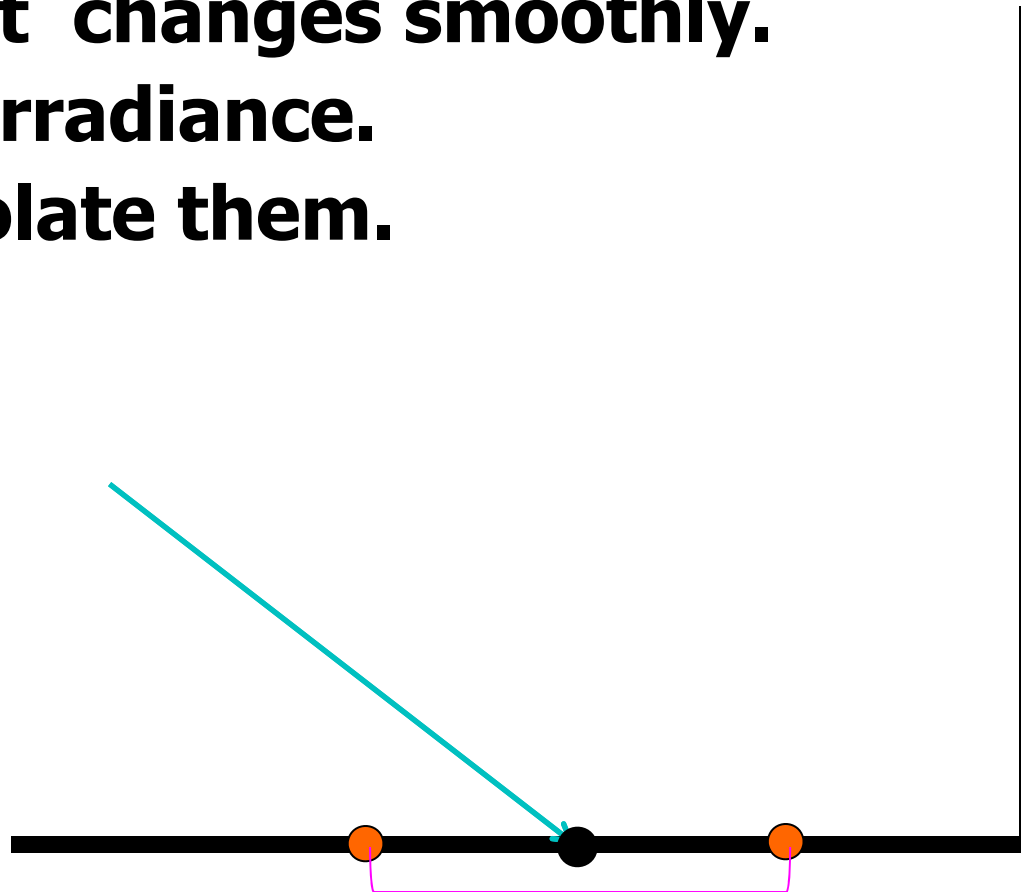
Irradiance Caching Approach

- **Irradiance E(x) estimated using MC**
- **Cache irradiance when possible**
  - **Store in octree for fast access**
- **When do we use this cache of irradiance values?**

KAIST

Smoothness Measure

- When new sample requested
  - Query octree for samples near location
  - Check ε at x, x_i is a nearby sample

  $$\varepsilon_i(x,\vec{n}) = \frac{\|x_i - x\|}{R_i} + \sqrt{1 - \vec{n} \bullet \vec{n}_i}$$
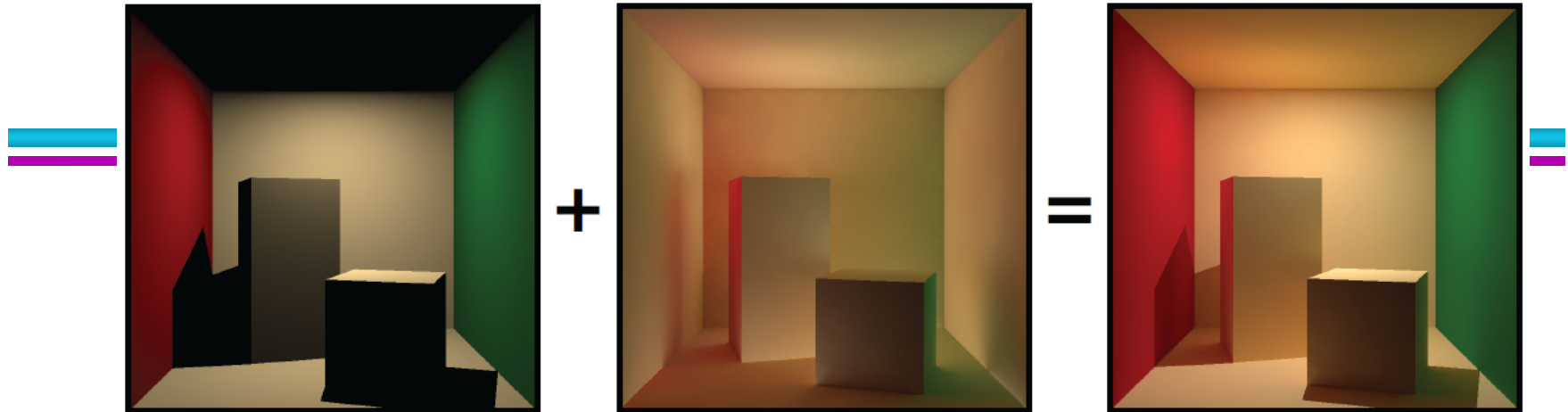
  - Weight samples inversely proportional to ε_i

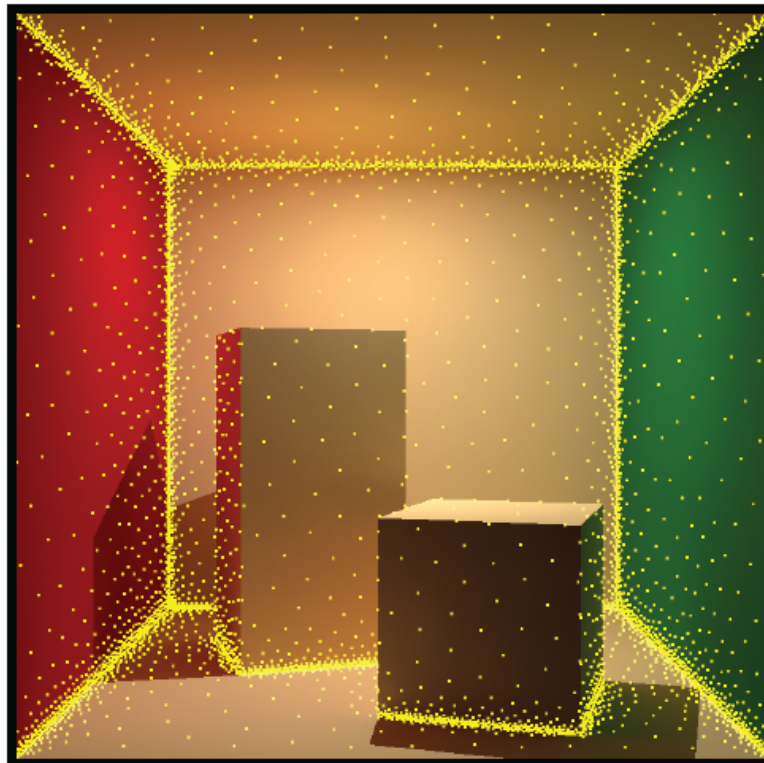 $$E(x,\vec{n}) = \frac{\sum_{i, w_i > 1/a} w_i(x,\vec{n}) E_i(x_i)}{\sum_{i, w_i > 1/a} w_i(x,\vec{n})}$$
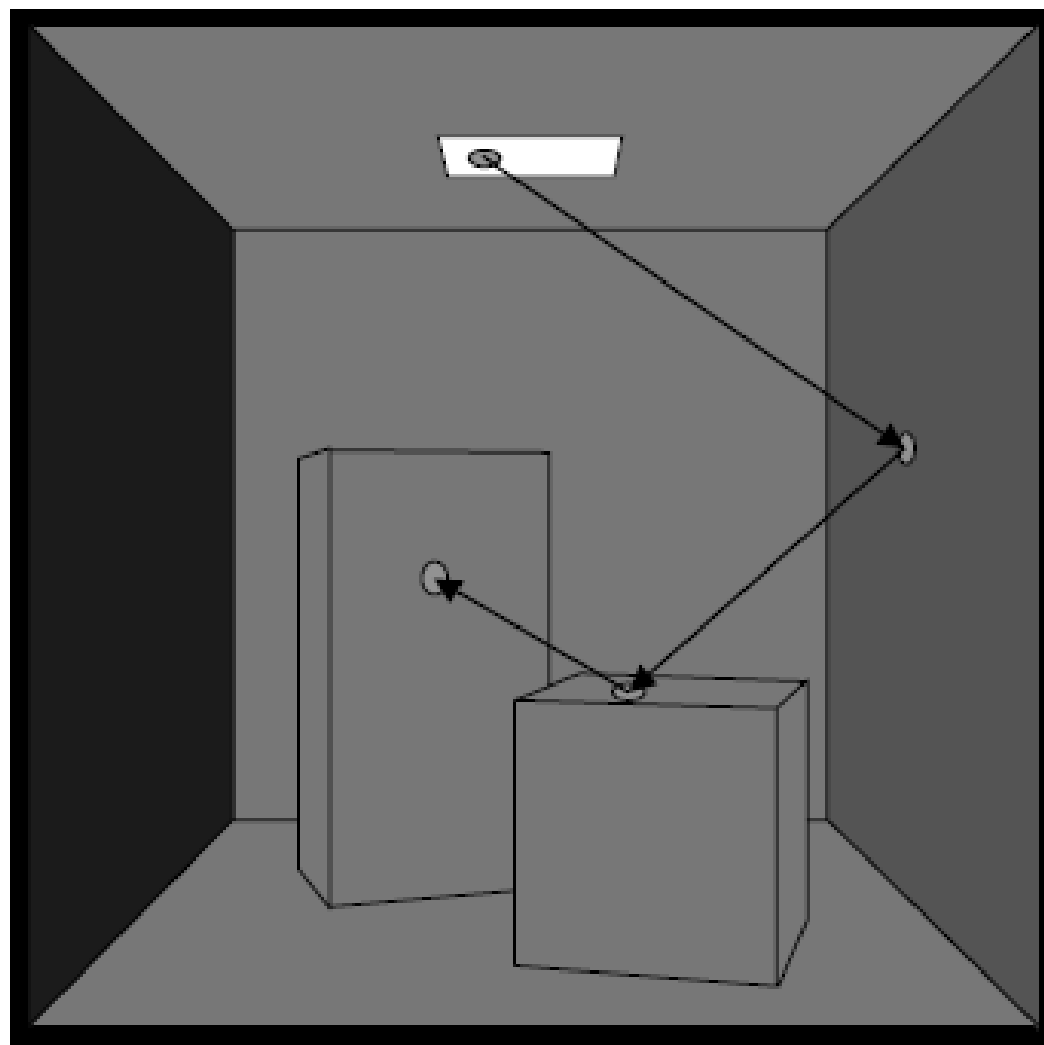
  - Otherwise, compute new sample

Direct

Indirect

+

=

Indirect ≈

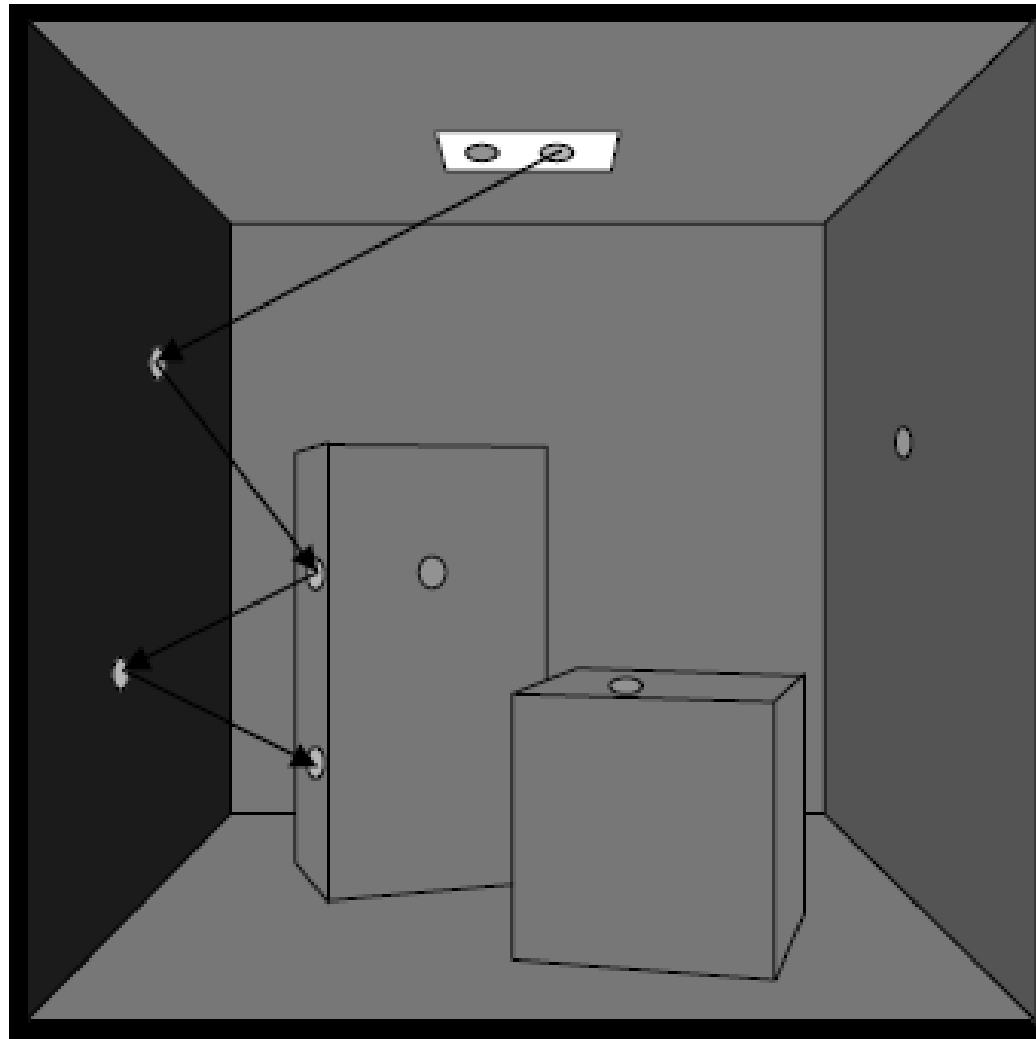
From thesis of Jarosz

KAIST

Photon Mapping

- **2 passes:**
  - **Shoot "photons" (light-rays) and record any hit-points**
  - **Shoot viewing rays and collect information from stored photons**

KAIST
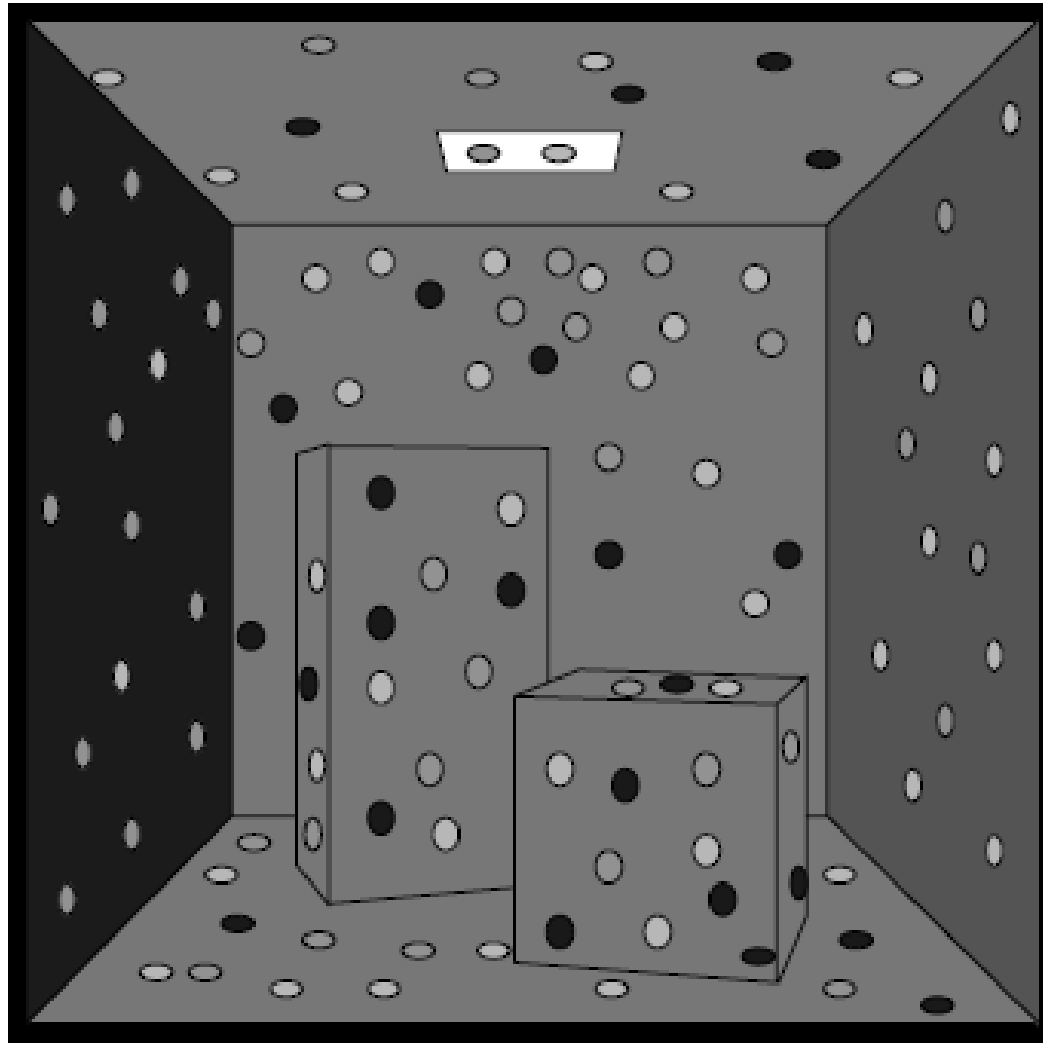
Pass 1: shoot photons

- Light path generated using MC techniques and Russian Roulette

- Store:
  - position
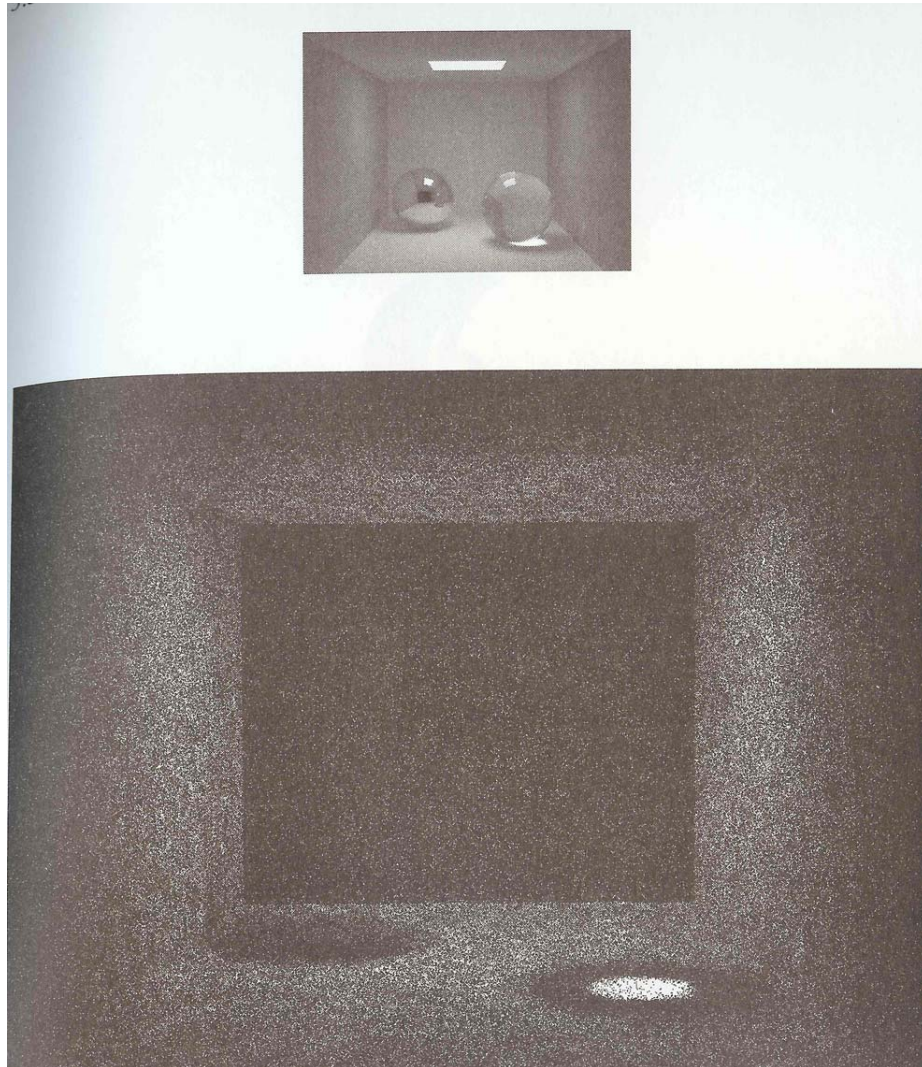 - incoming direction
  - color
  - ...

Pass 1: shoot photons

- Light path generated using MC techniques and Russian Roulette

- Store: **Flux for each photon**
  - position
  - incoming direction
  - color
  - ...

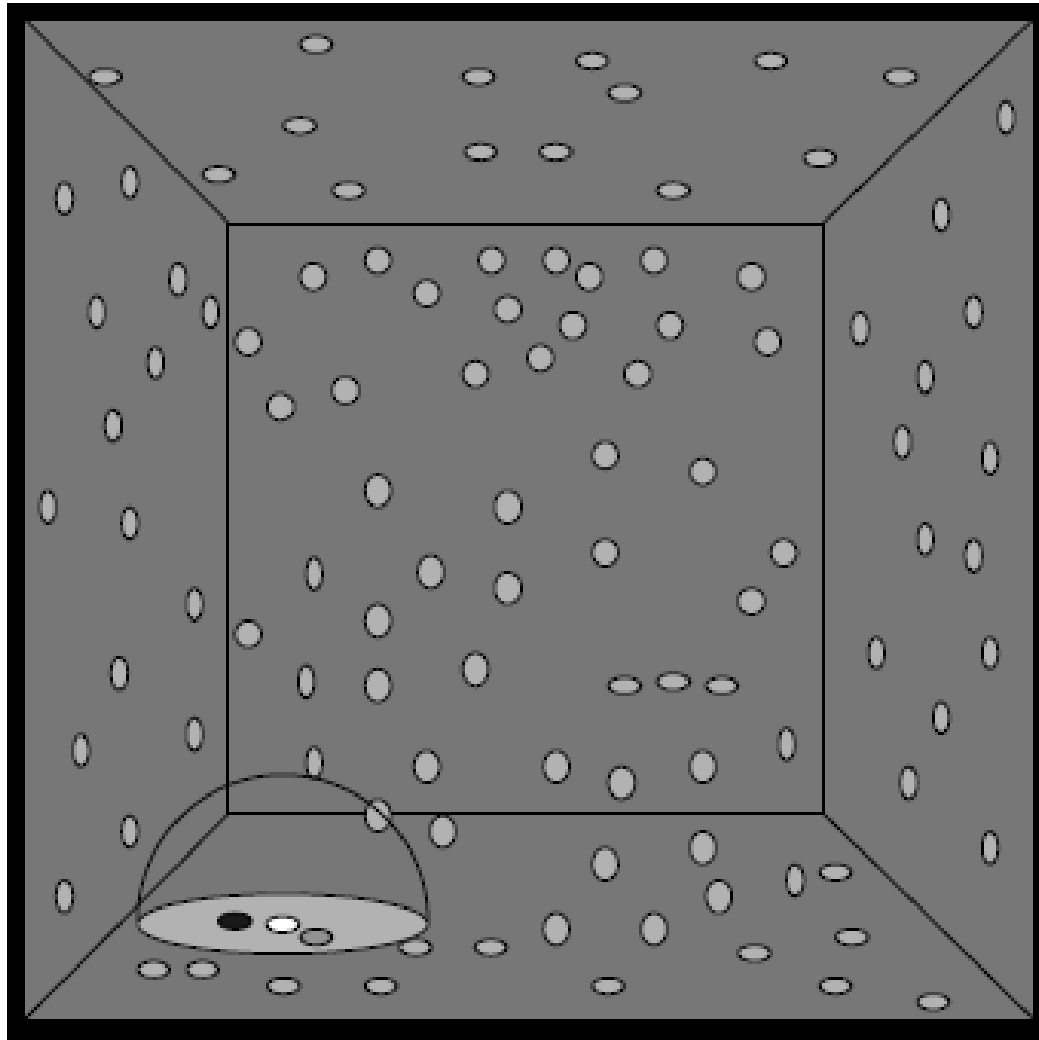# Pass 1: shoot photons

- Light path generated using MC techniques and Russian Roulette

- Store: **for diffuse materials**
  - position
  - incoming direction
  - color
  - ...

Stored Photons

**Generate a few hundreds of thousands of photons**

Pass 2: viewing ray

- Search for N closest photons (+check normal)

- Assume these photons hit the point we're interested in

- Compute average radiance
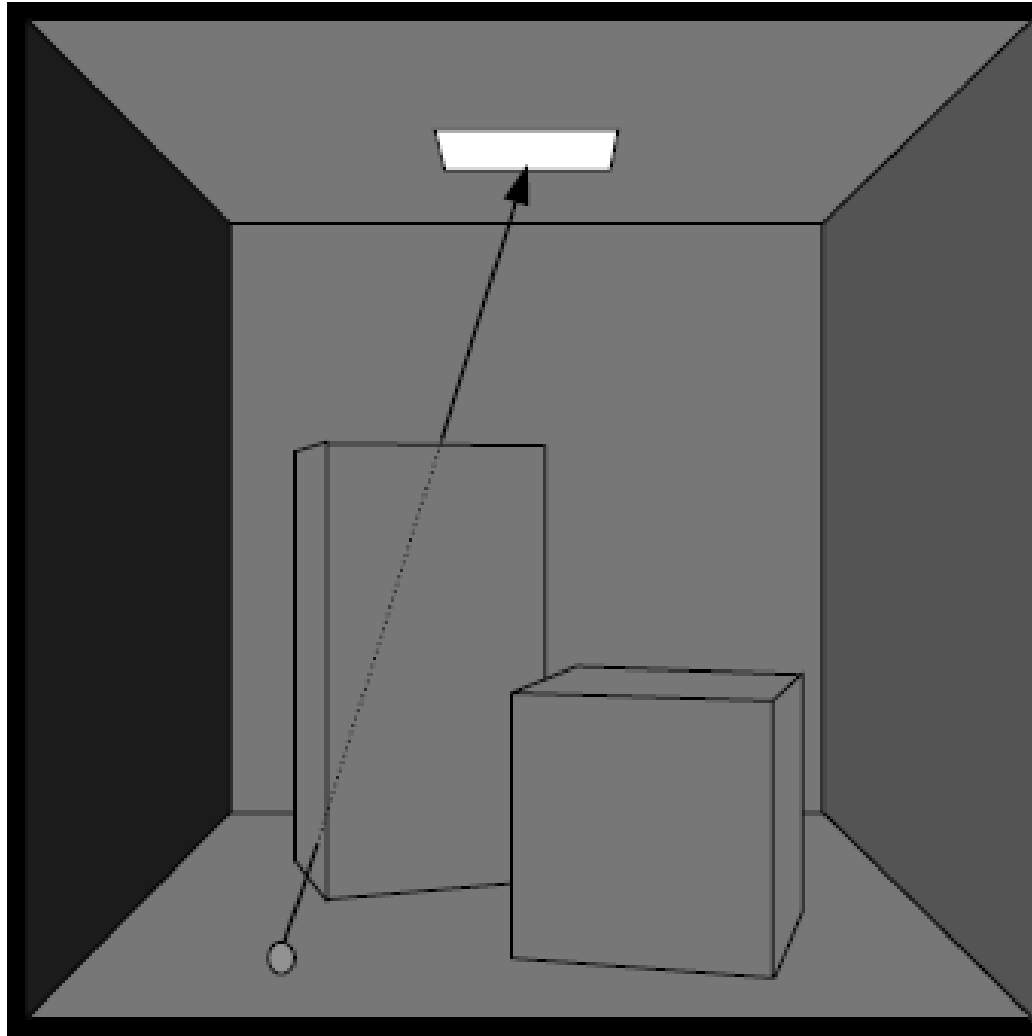
Radiance Estimation

- **Compute N nearest photons**
  - **Consider a few hundreds of photons**
  - **Compute the radiance for each photon to outgoing direction**
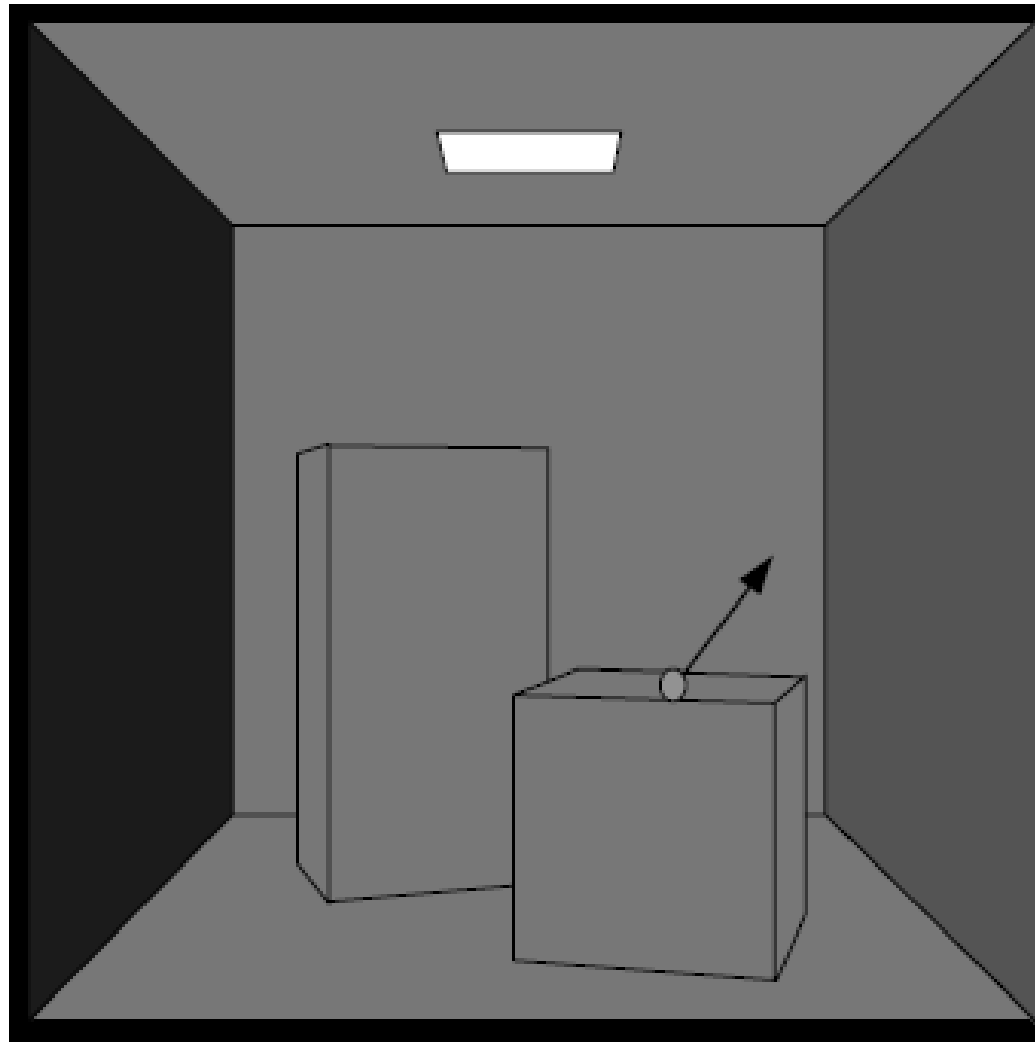 - **Consider BRDF and**
  - **Divided by area**
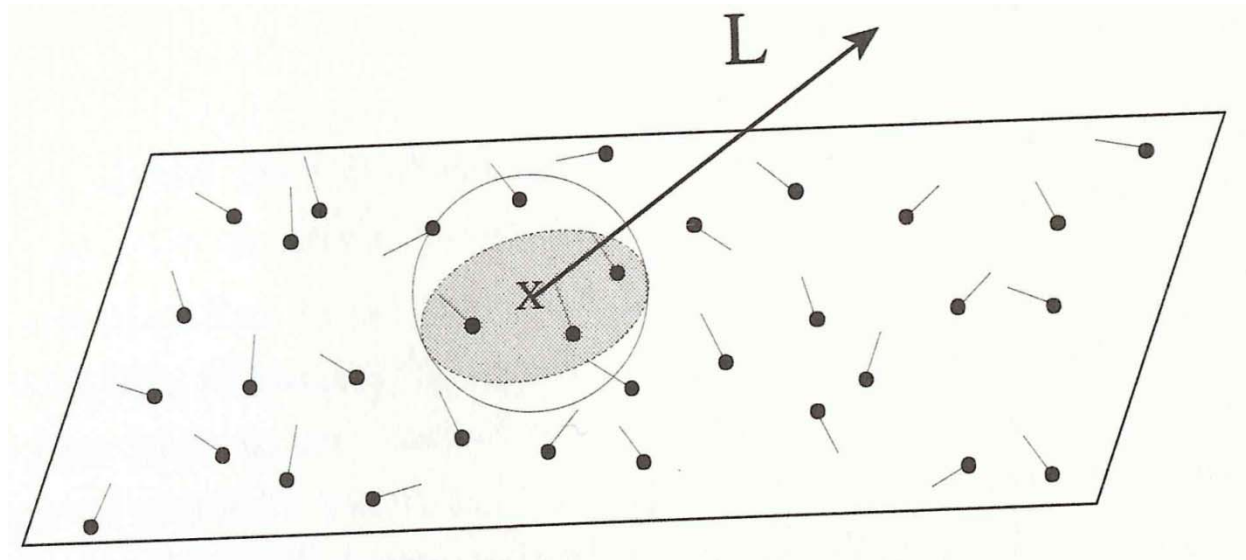
Efficiency

- **Want k nearest photons**
  - **Use kd-tree**

- **Using photon maps as it create noisy images**
  - **Need extremely large amount of photons**

KAIST

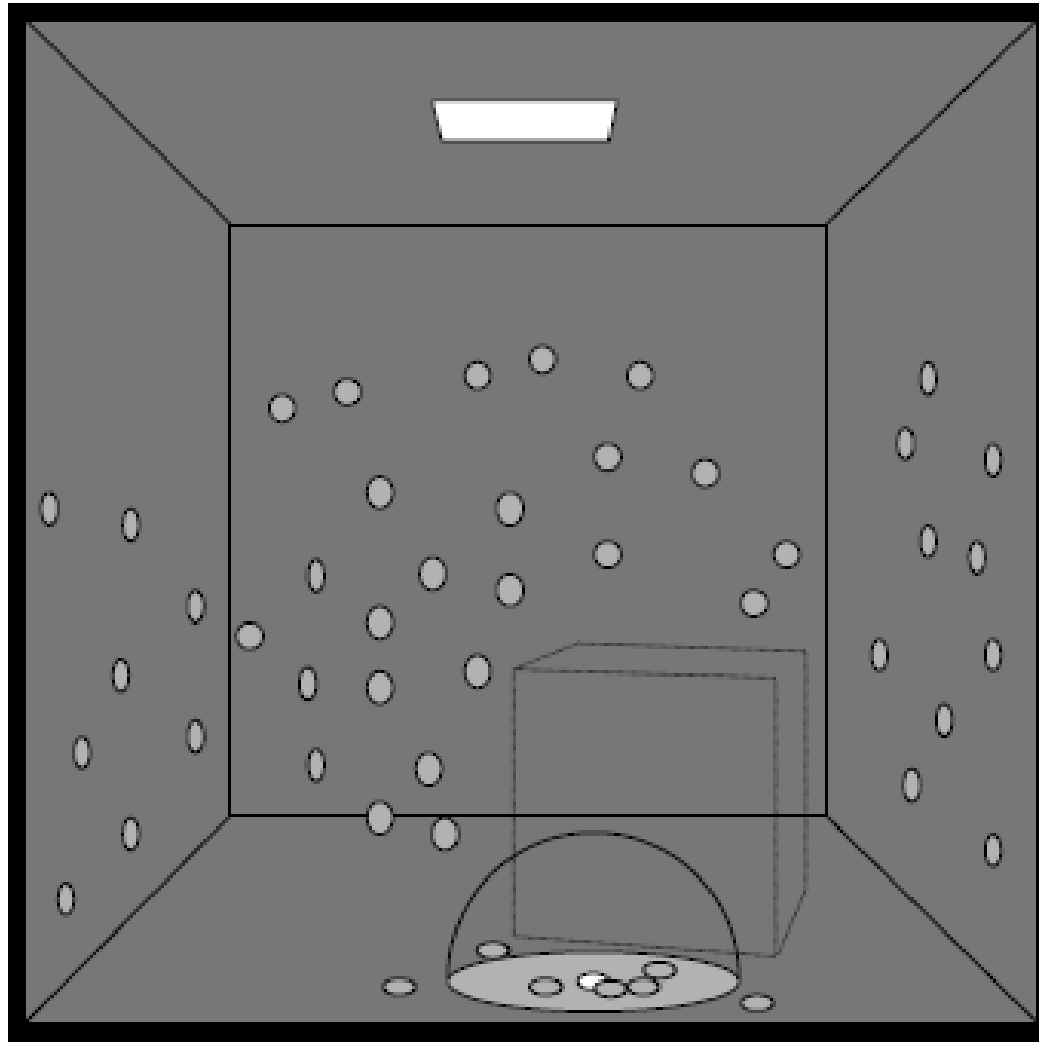# Pass 2: Direct Illumination

Perform direct illumination for visible surface using regular MC sampling

© Kavita Bala, Computer Science, Cornell University

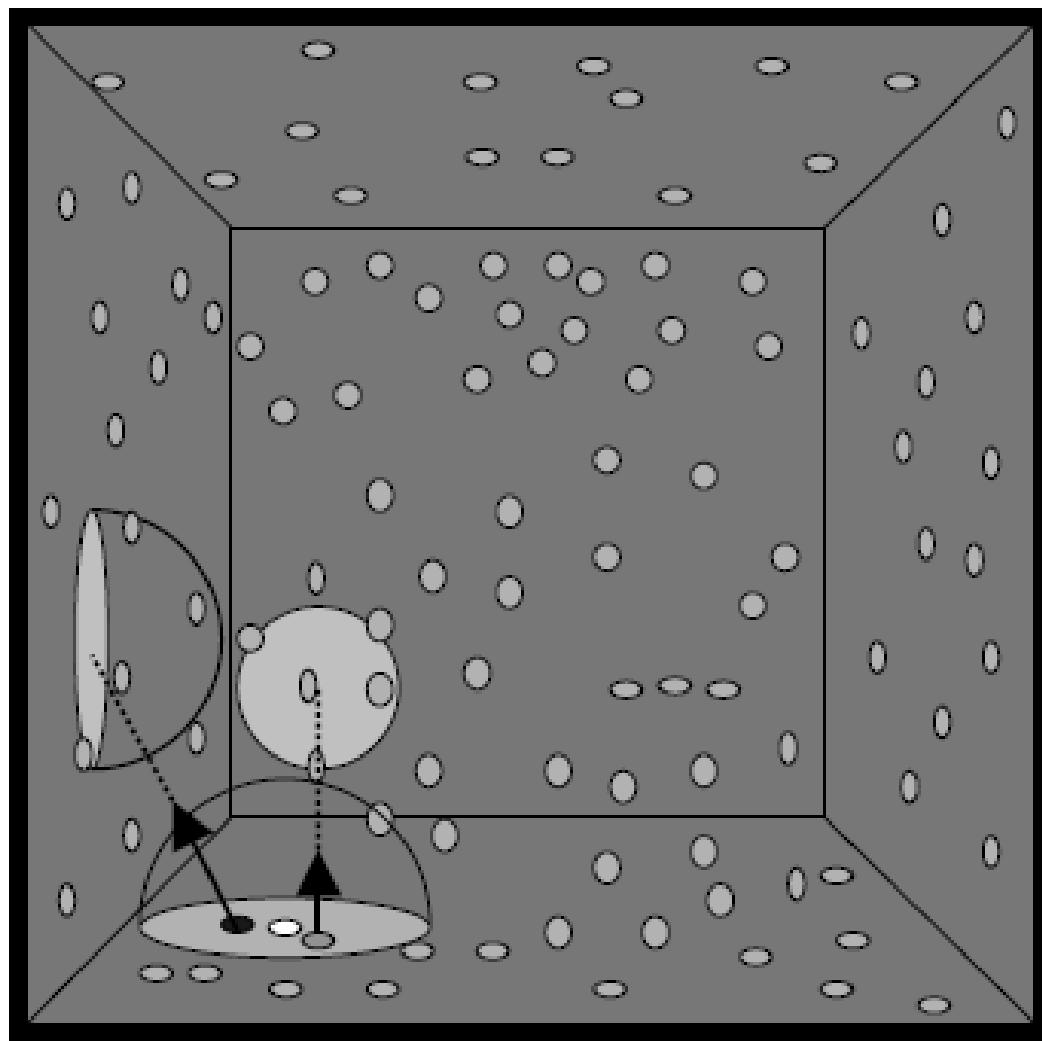# Pass 2: Specular reflections

Specular reflection and transmission are ray traced

Pass 2: Caustics

- Direct use of "caustic" maps

- The "caustic" map is similar to a photon map but treats LS*D path

- Density of photons in caustic map usually high enough to use as is

Pass 2:Indirect Diffuse

- Search for N closest photons

- Assume these photons hit the point

- Compute average radiance by importance sampling of hemisphere

© Kavita Bala, Computer Science, Cornell University

Result

350K photons for the caustic map

Progressive Photon Mapping [Hachisuka et al., SIG. A. 08]

- **Photon mapping**
  - A consistent algorithm and good at caustics and SDS paths
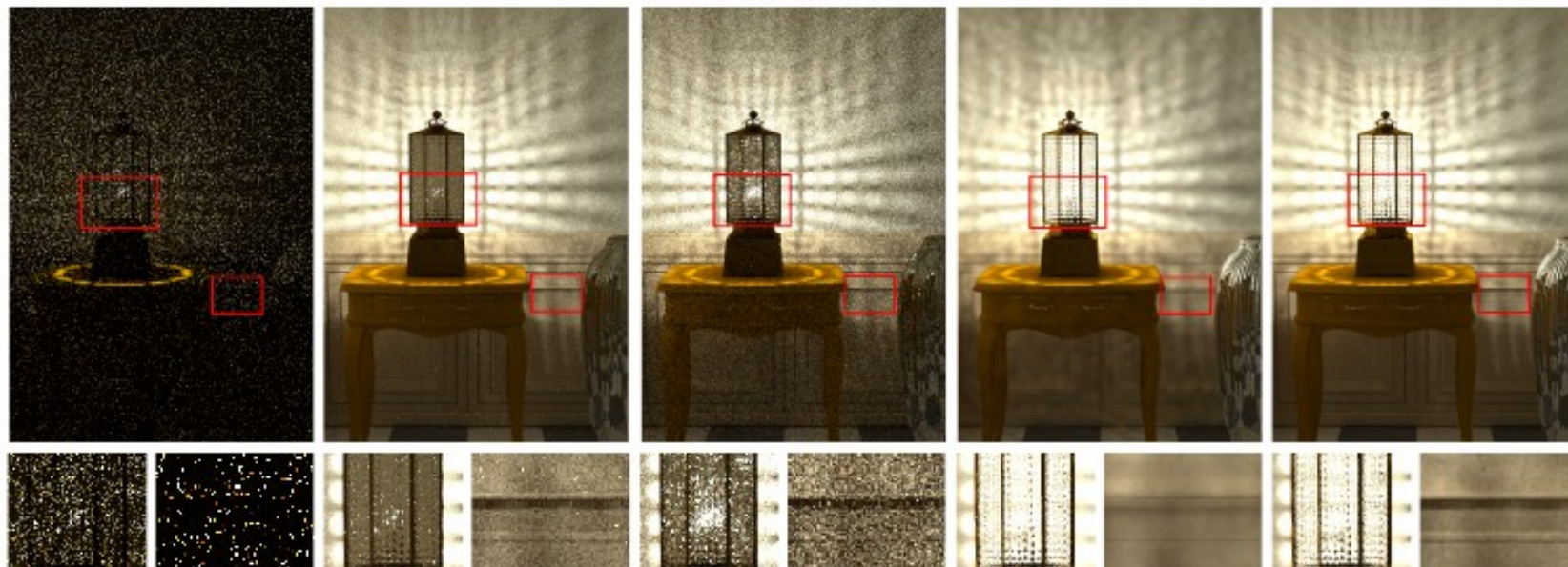 - Requires huge # of photons to avoid noises

Img.blog.yahoo.co.kr/ybi

KAIST

Progressive Photon Mapping [Hachisuka et al., SIG. A. 08]

- **Photon mapping**
  - **Requires huge # of photons to avoid noises**
    - Its quality is limited by the available memory

Path tracing Bidirectional path tracing Metropolis light transport Photon mapping Progressive photon mapping
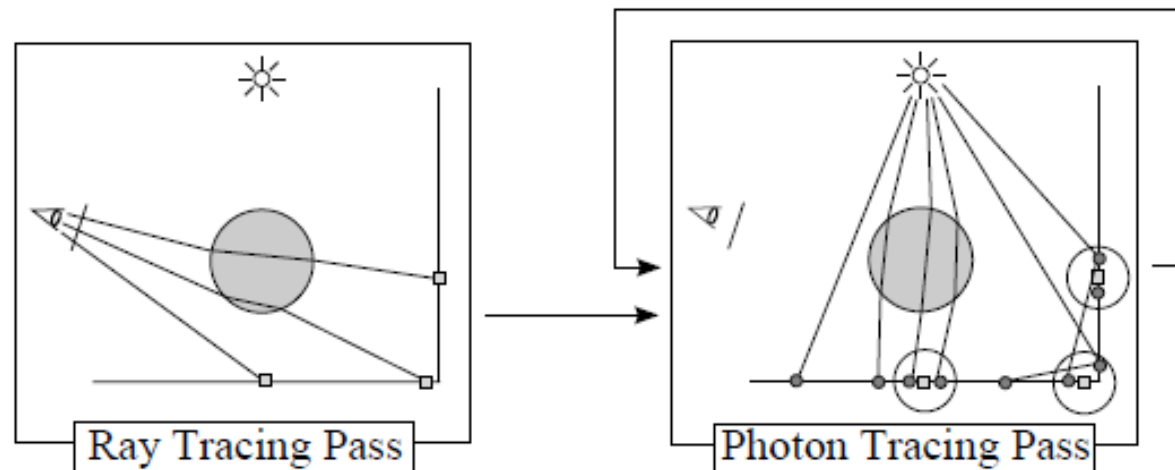
22 hours   20M photons   165M photons

KAIST

Overall Framework

- **Achieve arbitrary accuracy without requiring infinite memory**

- **Uses multiple phases**

Ray Tracing Pass

Photon Tracing Pass

- **Store extra information for all the hit points along all the ray paths**

  - **E.g., accumulated # of photons, flux, and current radius**

KAIST

Key Idea

Radius: R(x)
Photons: N(x)

Radius: R(x)
Photons: N(x) + M(x)

Radius: R(x) - dR(x)
Photons: N(x) + αM(x)
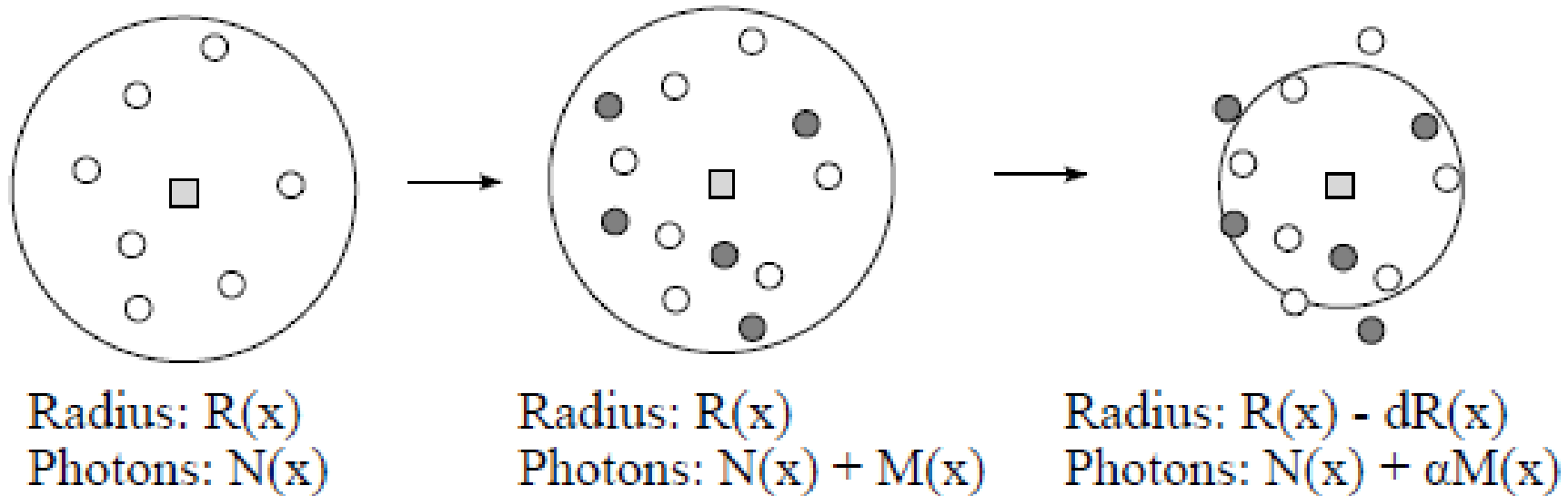
- **We want to increase # of photons and reduce radius while keeping photon density**

- **Key assumption:**
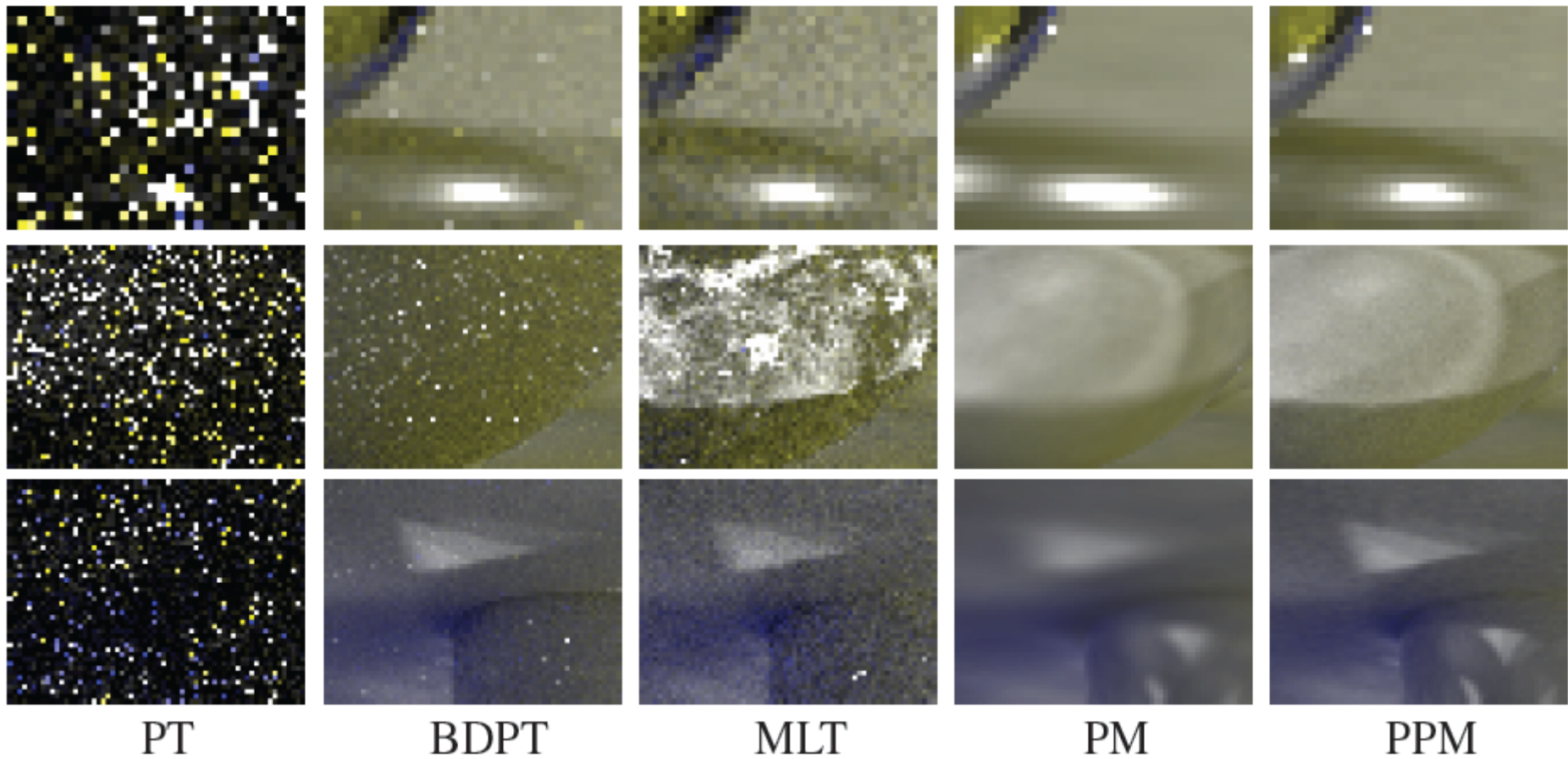 - **Uniform photon density and illumination within each radius**

Results
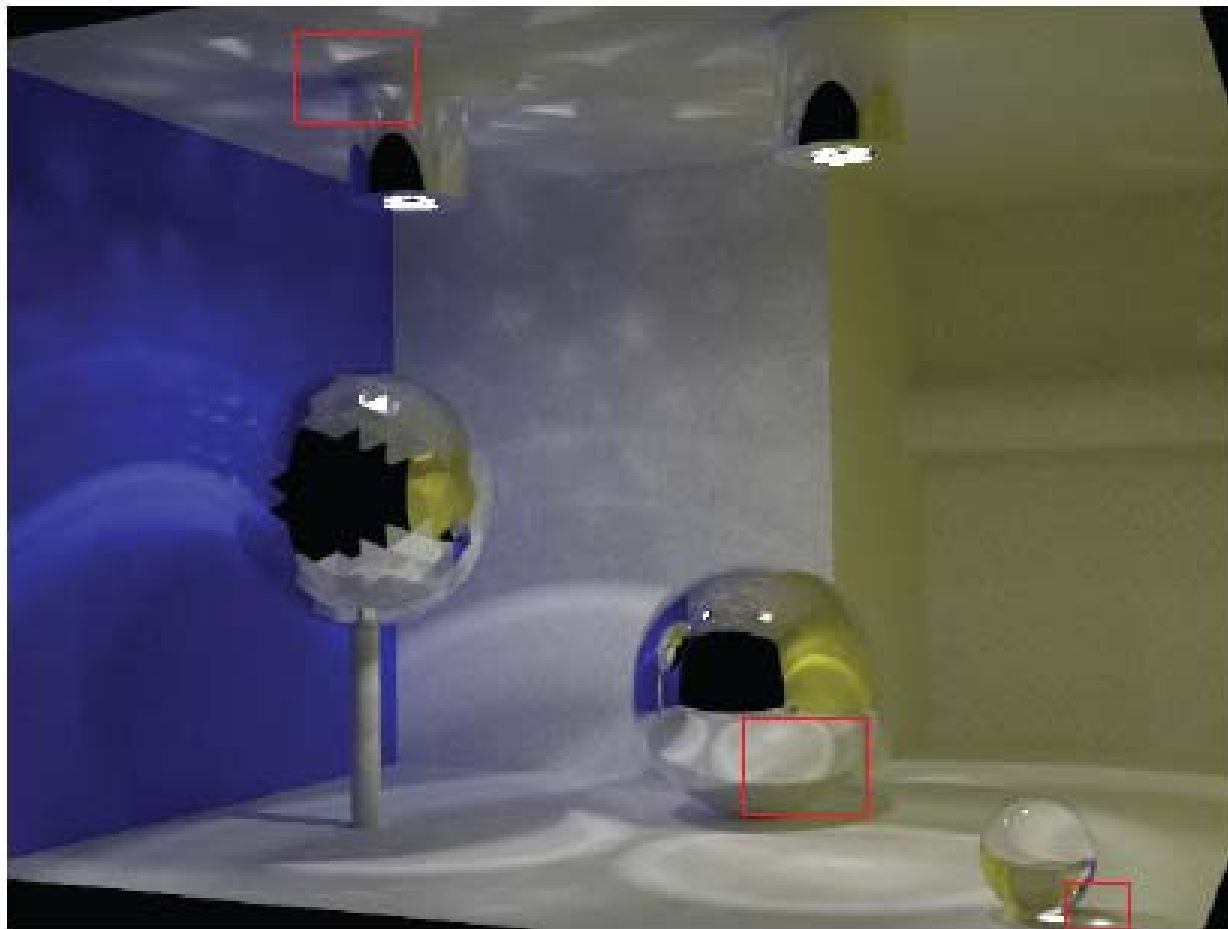
213M photons

Progressive photon mapping

Comparison

PT   BDPT   MLT   PM   PPM

KAIST

Future Work

- **Stopping criteria and error estimate**
  - **How many photons do we need?**

- **Adaptive photon tracing**
  - **We know how many photons are used in each hit point in the PPM framework**

KAIST

Class Objectives were:

- **Extensions to the basic MC path tracer**
  - **Bidirectional path tracer**
  - **Metropolis sampling**

- **Biased techniques**
  - **Irradiance caching**
  - **Photon mapping**

KAIST

Summary

- **Two basic building blocks**
- **Radiometry**
- **Rendering equation**
- **MC integration**
- **MC ray tracing**
  - **Unbiased methods**
  - **Biased methods**

KAIST

Summary