State of the Art in Photon Density Estimation

Photon Mapping Basics

Wojciech Jarosz

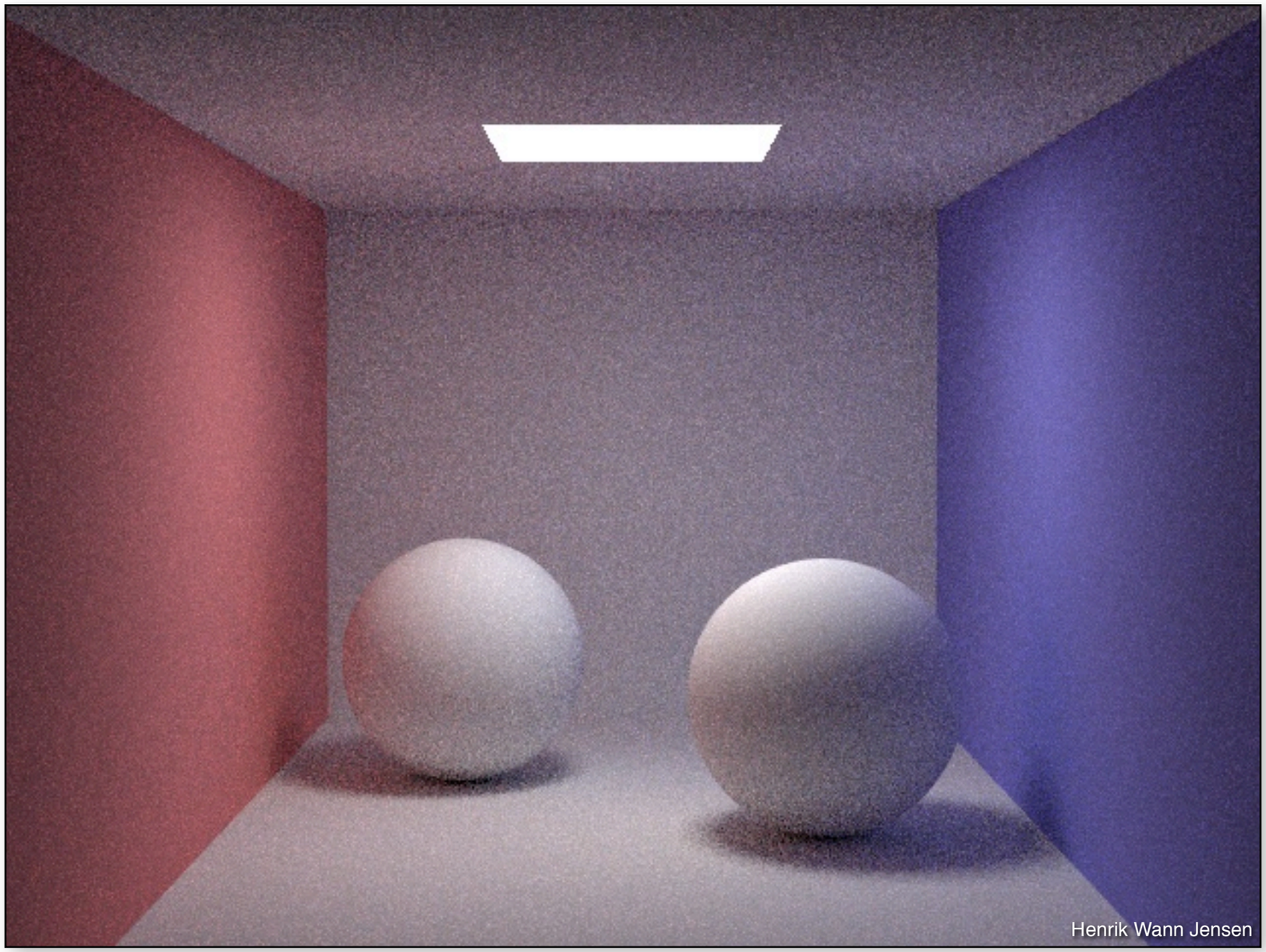
Motivation - Global Illumination

Motivation - Direct Illumination

Motivation - Before photon mapping

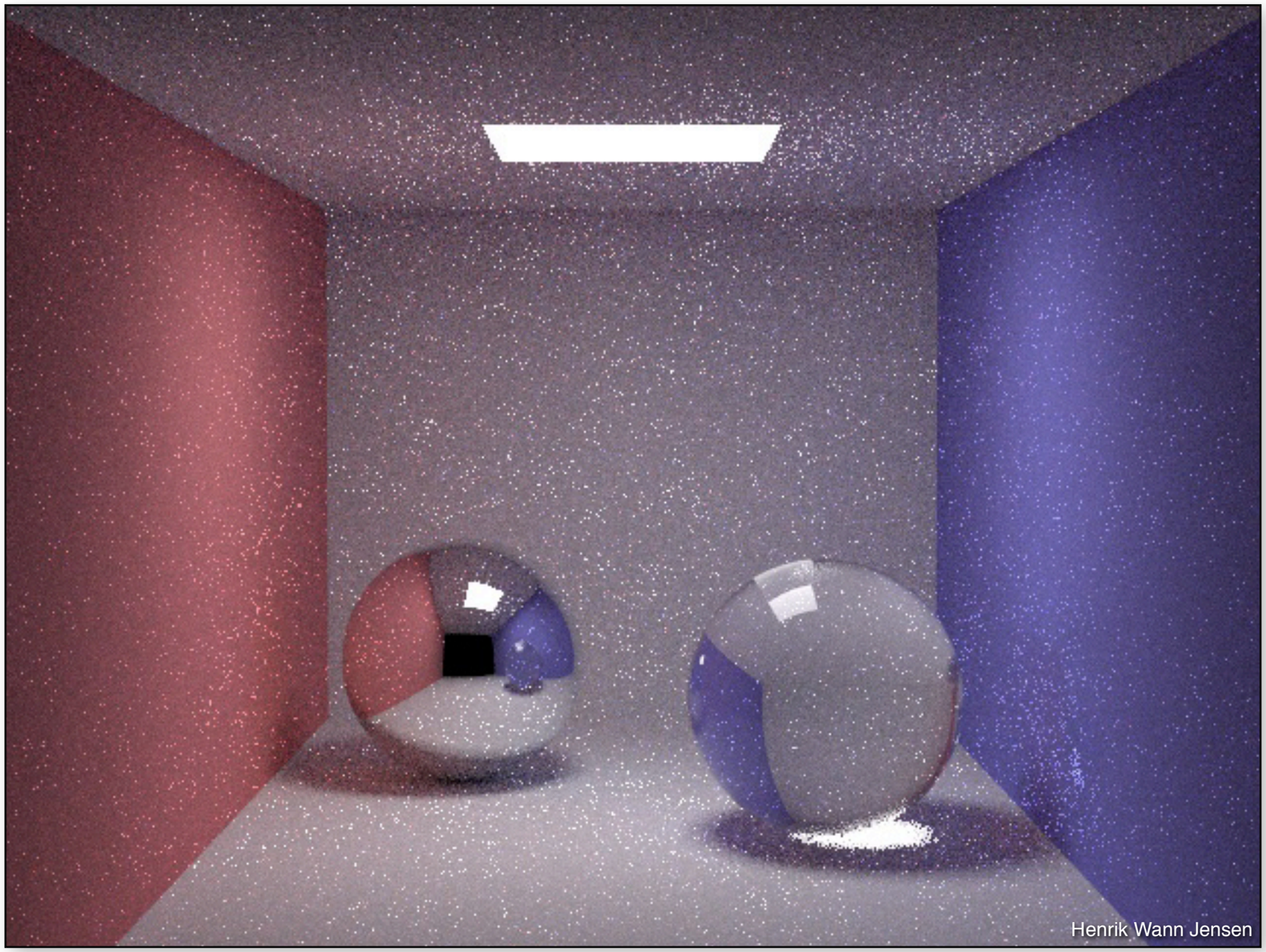
- Radiosity
  - ✘ Mostly diffuse
  - ✘ Mesh based lighting representation
- Monte Carlo path tracing
  - ✓ Very general
  - ✘ Noisy
  - ✘ Computation time/slow convergence

Path Tracing



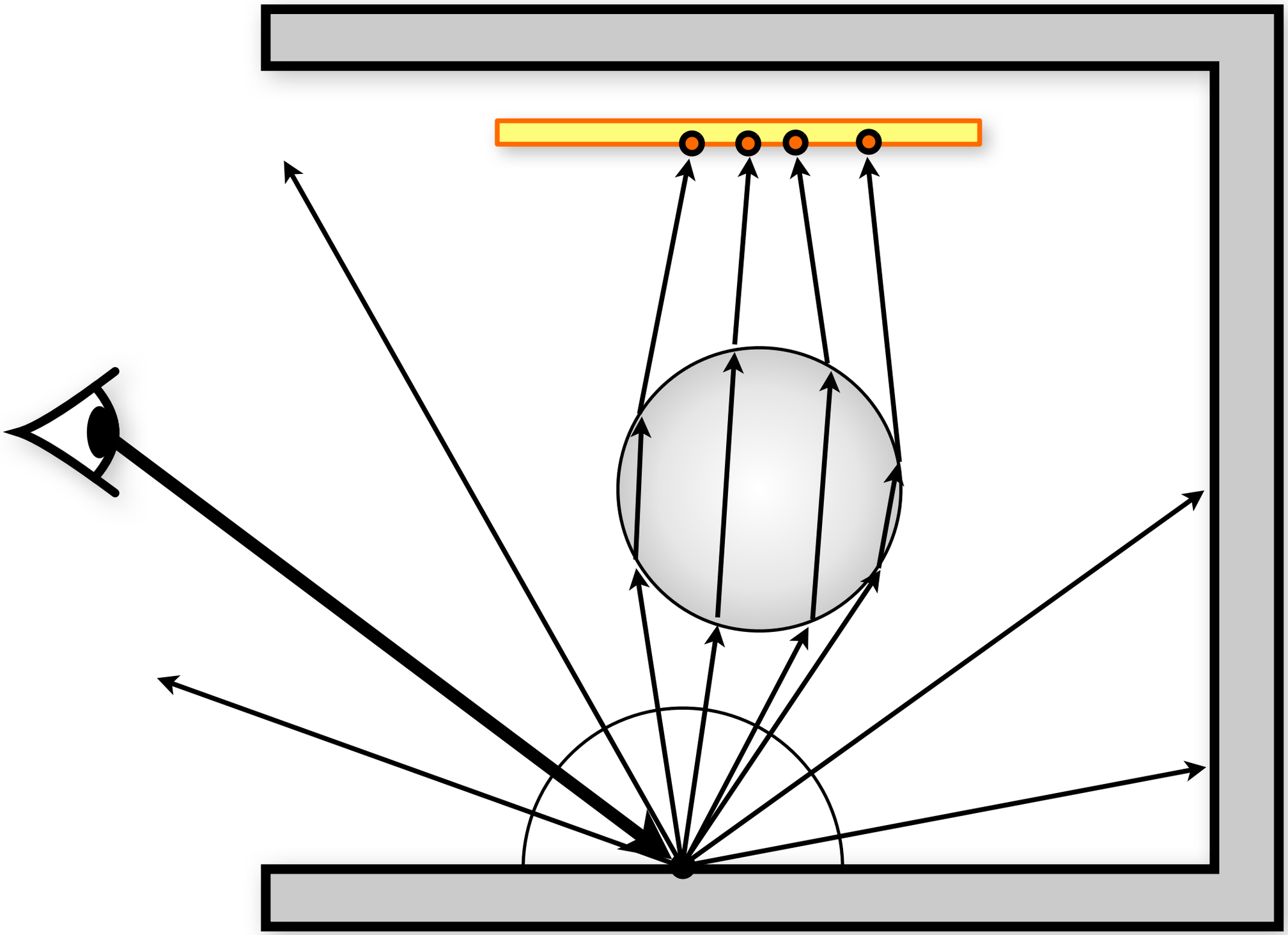
10 paths/pixel

Path Tracing Caustics

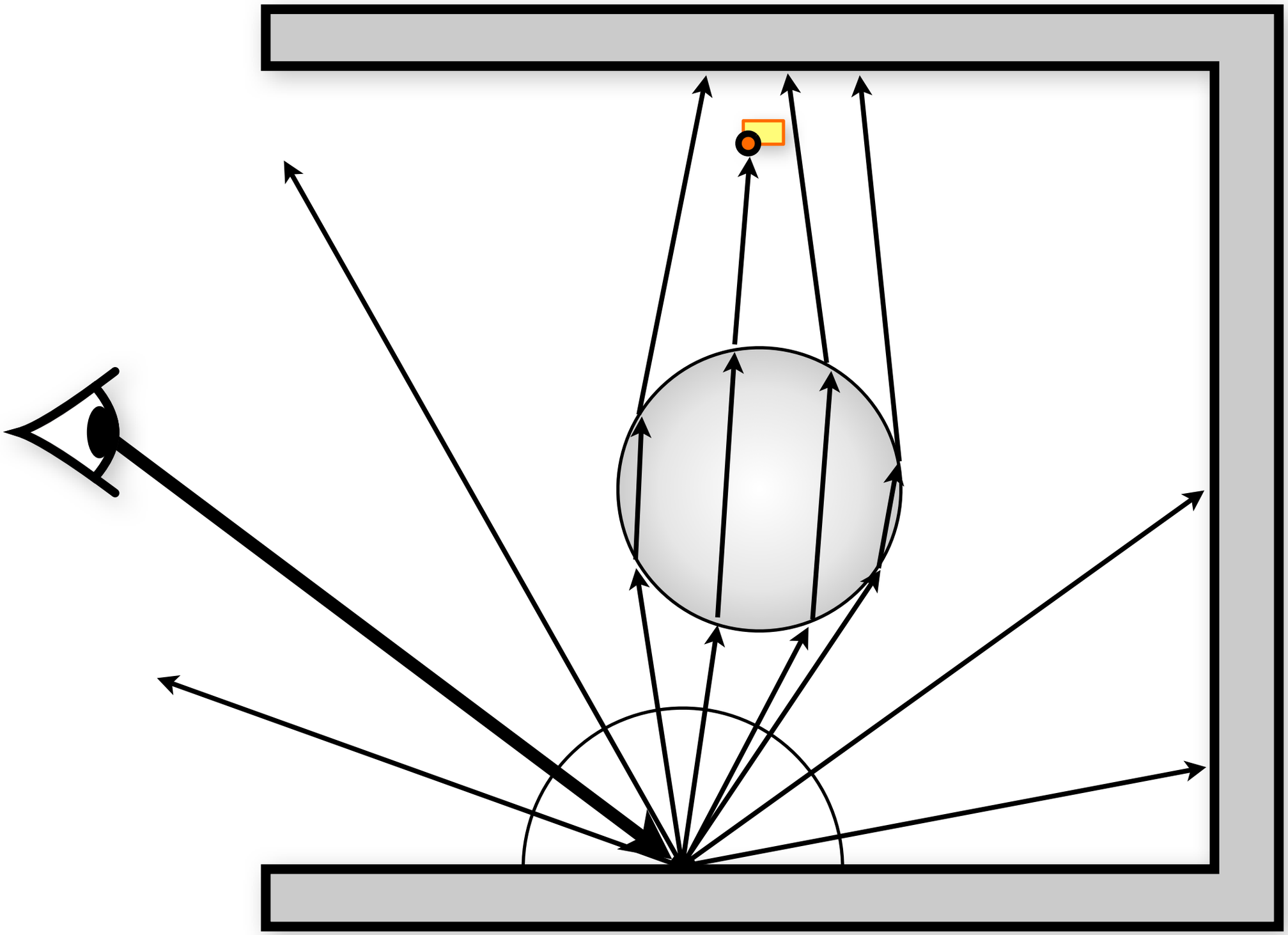


10 paths/pixel

Path Tracing Caustics

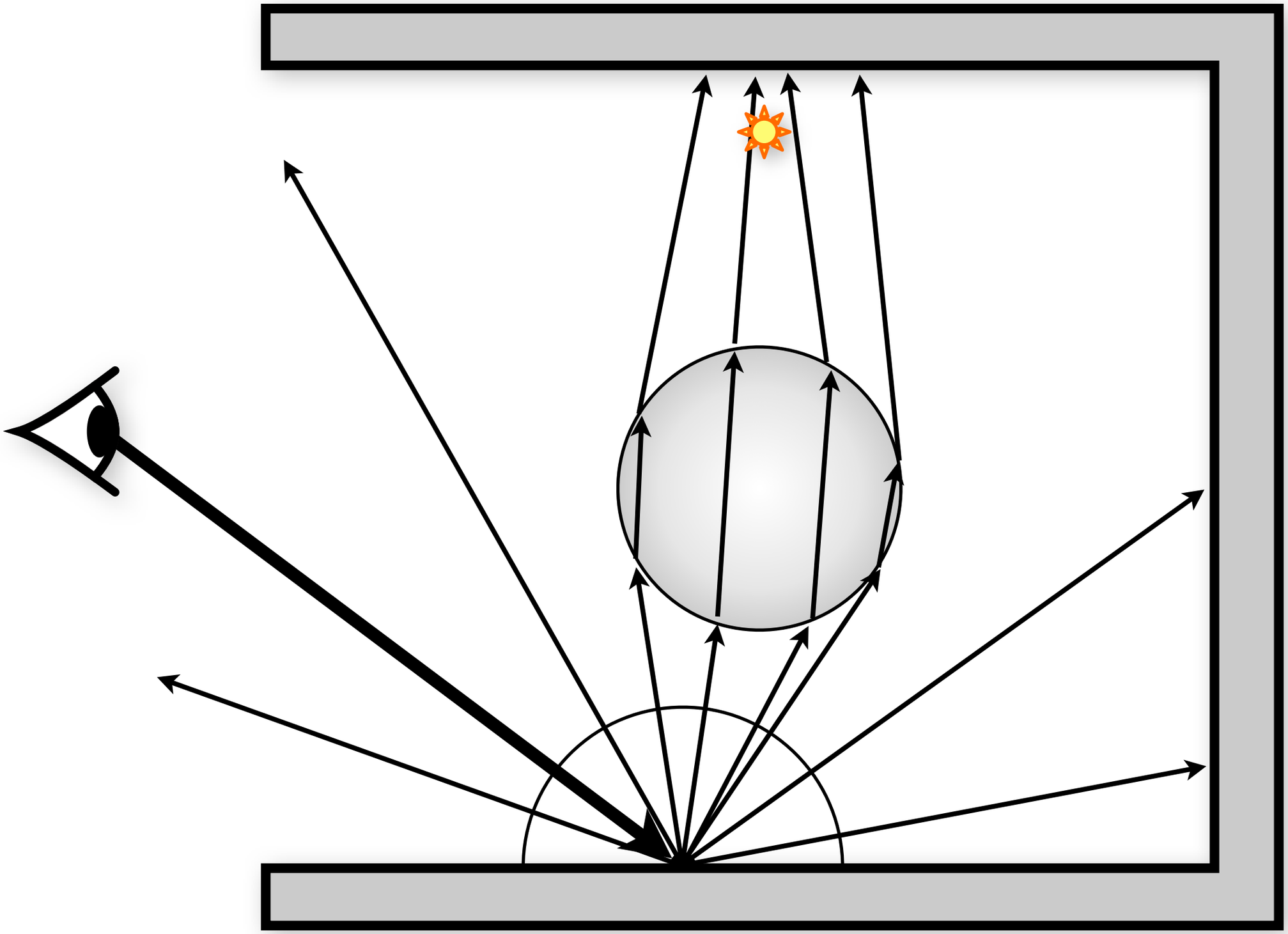


Path Tracing Caustics



Path Tracing Caustics

SIGGRAPHASIA2013



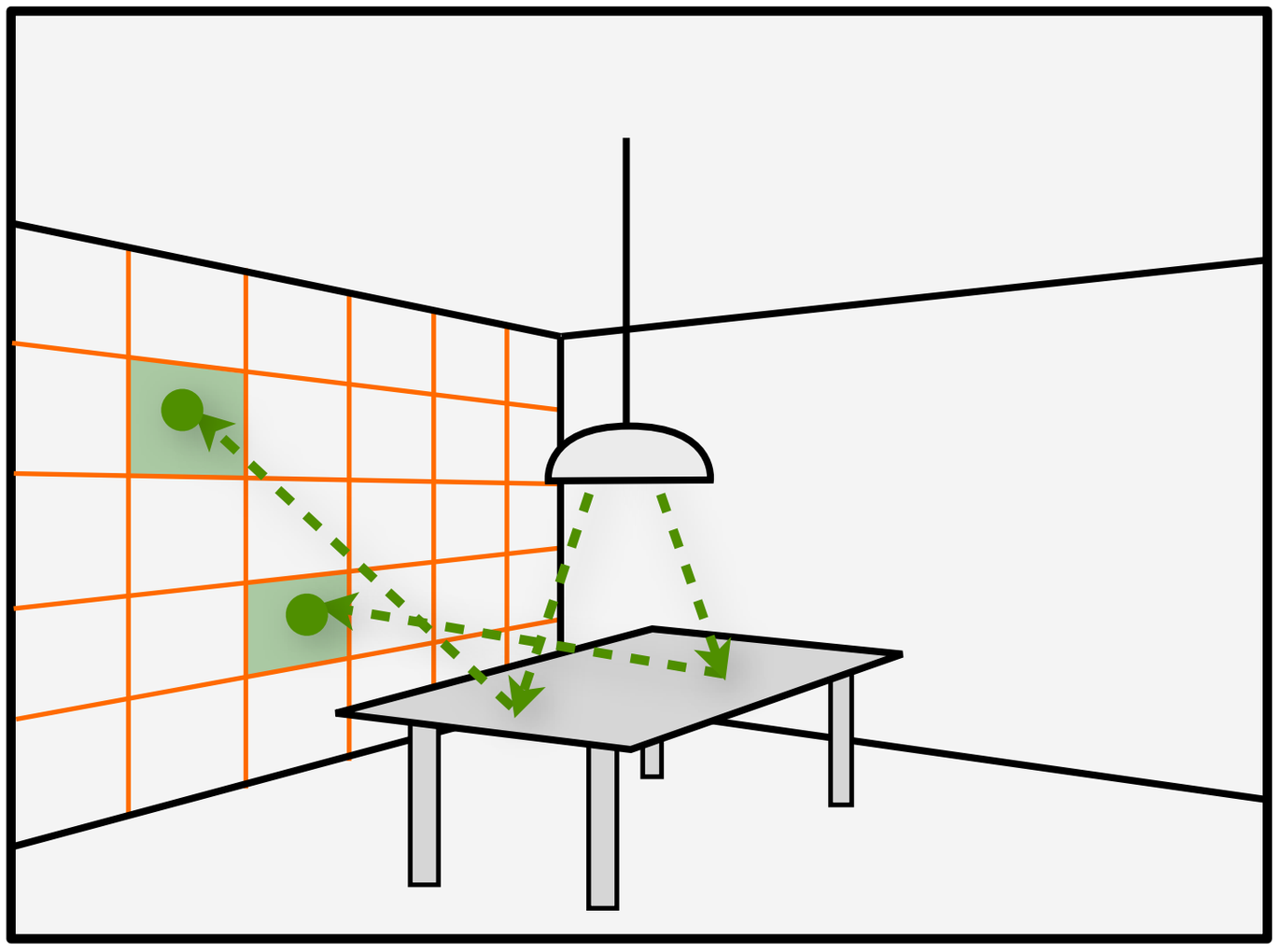
Density Estimation

- Light tracing / “Backward ray tracing”
- James Arvo. In *Developments in Ray Tracing*, SIGGRAPH ‘**86** Course Notes

Light Tracing ("Backward" Ray Tracing)

SIGGRAPHASIA2013

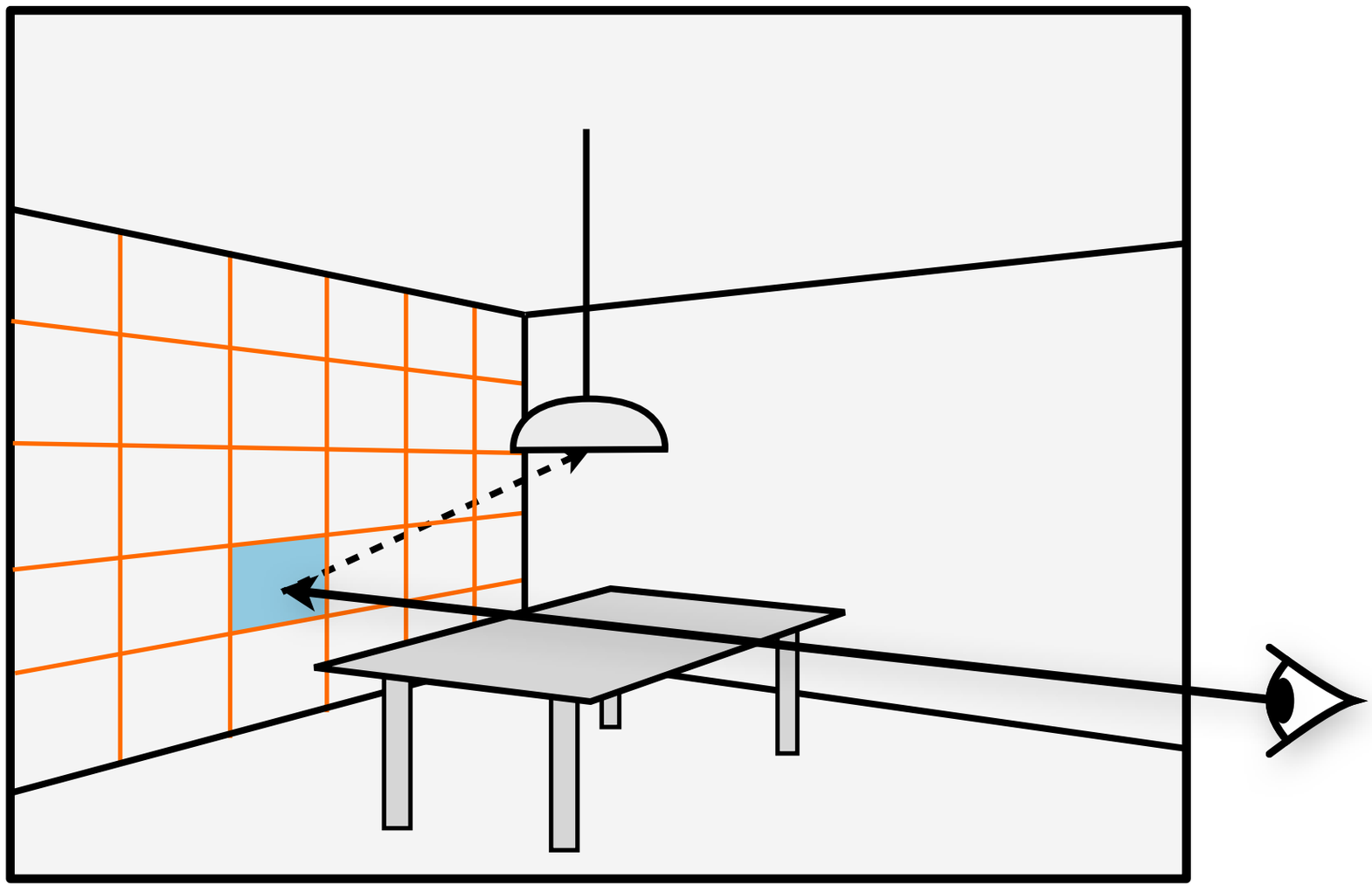
- Preprocess:
  - parametrize surfaces and create "illumination maps"
  - shoot light from light sources
  - deposit photon energy in illumination maps



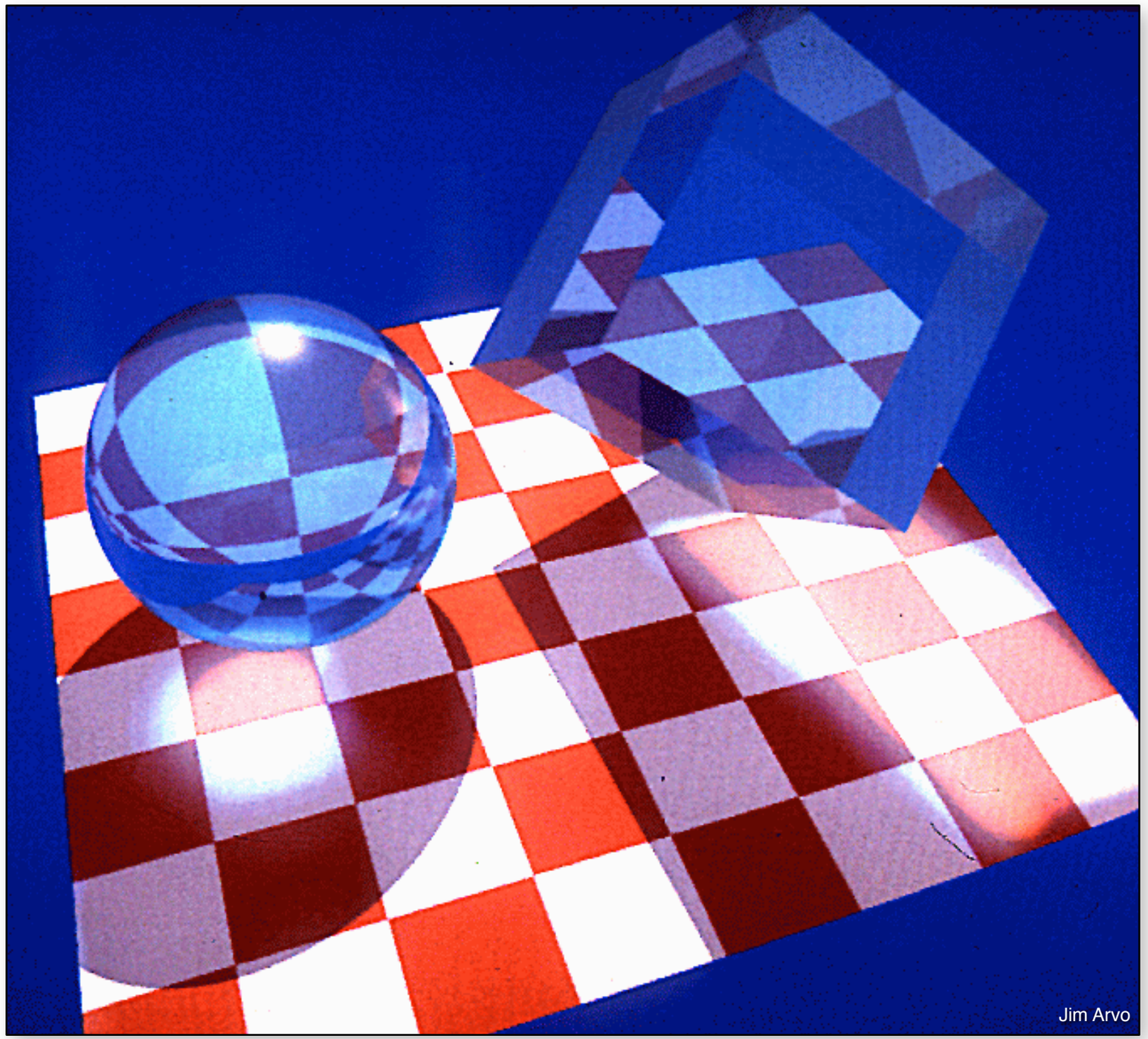
Rendering

SIGGRAPHASIA2013

- For each shading point
  - compute direct lighting
  - lookup indirect lighting + caustics from illumination maps



Light Tracing



"Backward Ray Tracing", by James Arvo. In *Developments in Ray Tracing*, SIGGRAPH `86 Course Notes, Volume 12, August 1986.

Light Tracing

SIGGRAPHASIA2013

✓ One of first techniques to simulate caustics

✗ Requires parametrizing surface or meshing

- Complex/procedural geometry difficult to handle

✗ Illumination map resolution difficult to choose

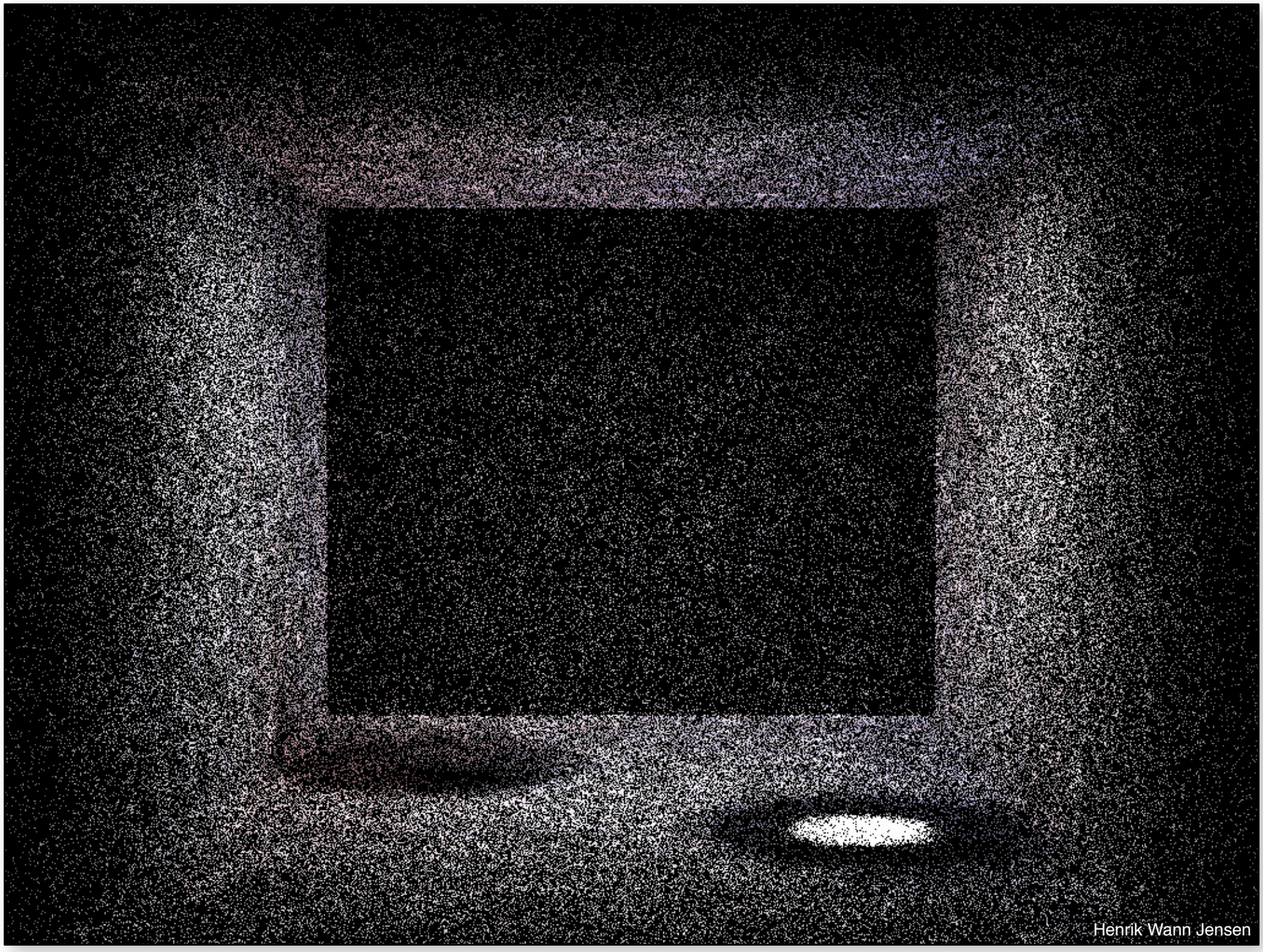
- high resolution/few photons: high-frequency noise
- low resolution/many photons: blurred illumination

Photon Mapping

SIGGRAPHASIA2013

- A two-pass algorithm:
  - Pass 1: Trace virtual photons from the light source, scatter at surfaces, and cache
  - Pass 2: Ray trace the scene and use the photons to compute indirect illumination
- Same as light tracing, but different way of storing/caching photons

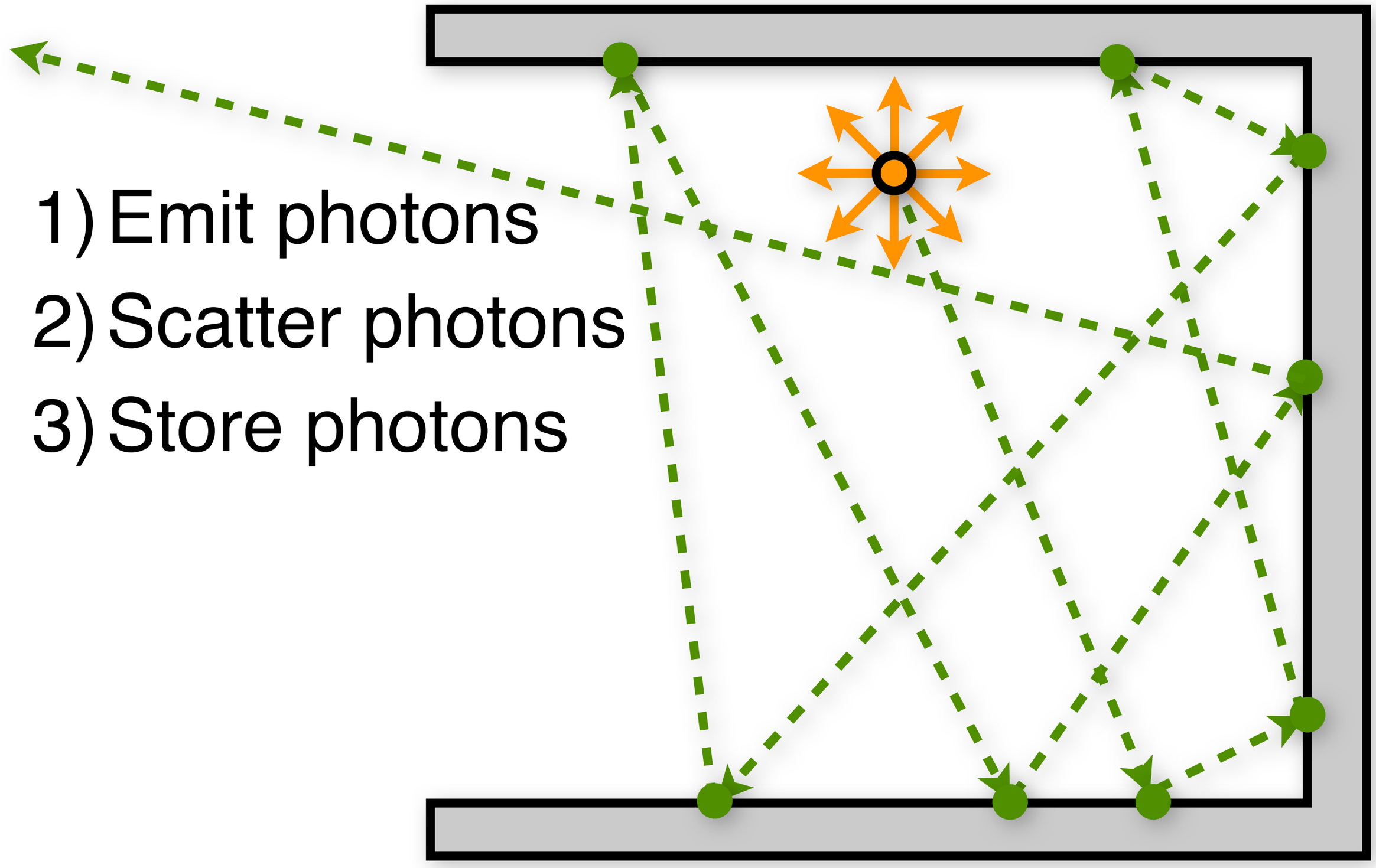
The Photon Map



Photon Mapping (Photon Tracing)

SIGGRAPHASIA2013

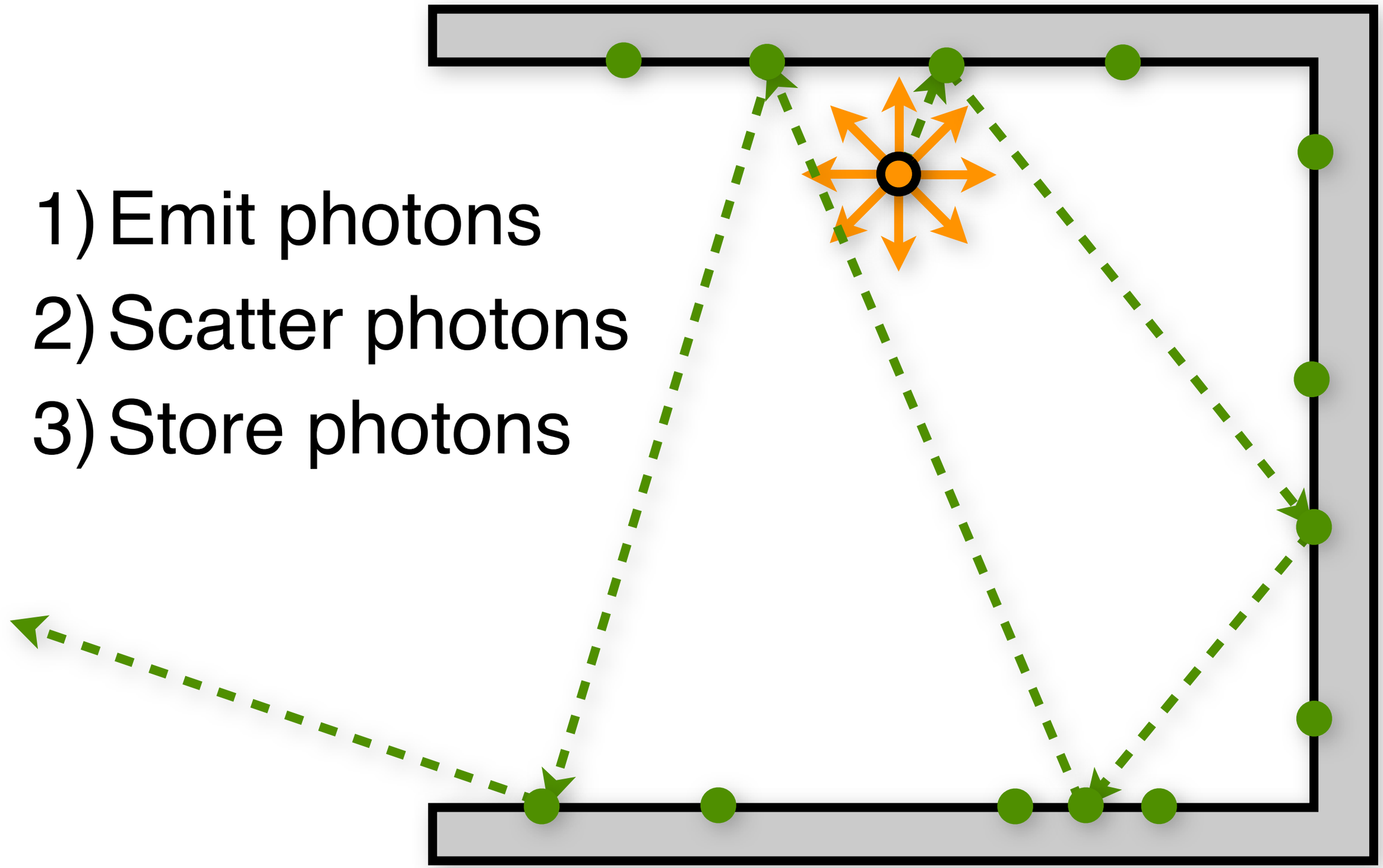
1) Emit photons
2) Scatter photons
3) Store photons



Photon Mapping (Photon Tracing)

SIGGRAPHASIA2013

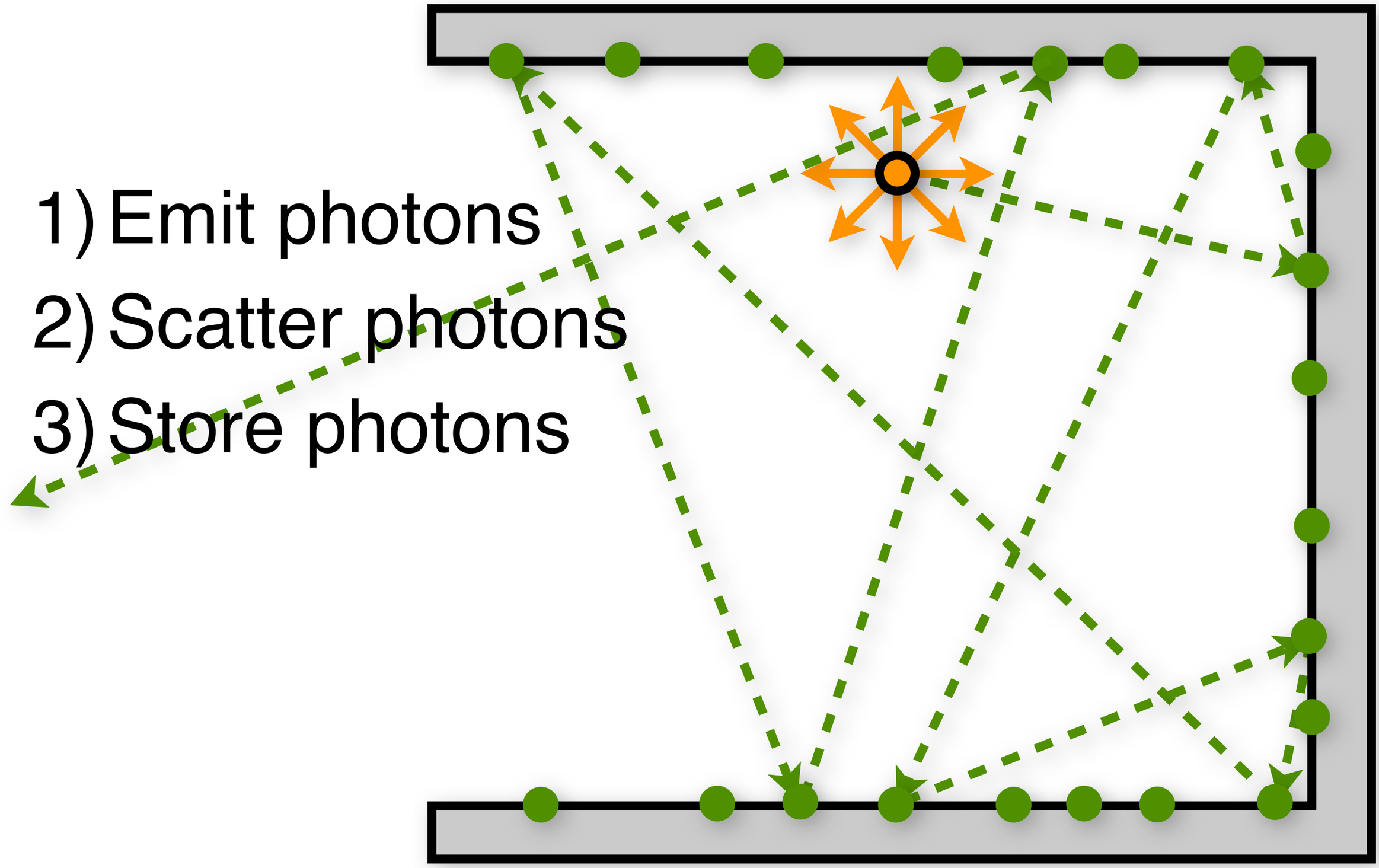
1) Emit photons
2) Scatter photons
3) Store photons



Photon Mapping (Photon Tracing)

SIGGRAPHASIA2013

1) Emit photons
2) Scatter photons
3) Store photons



Photon emission

- Define initial:
  - x_p : position
  - ω_p : direction
  - Φ_p : photon power
- Recipe:
  - Sample position on surface area of light $x_p \sim pdf(x_p)$
  - Sample direction $\omega_p \sim pdf(\omega_p)$
  - $\Phi p = L_e(x_p, \omega_p) / pdf(x_p) / pdf(\omega_p)$

Photon emission examples

- Isotropic point light:
  - Generate uniform random direction over sphere
- Spotlight:
  - Generate uniform random direction within spherical cap
- Diffuse square light
  - Generate uniform random location on square
  - Generate cosine-weighted direction over hemisphere

Pseudocode

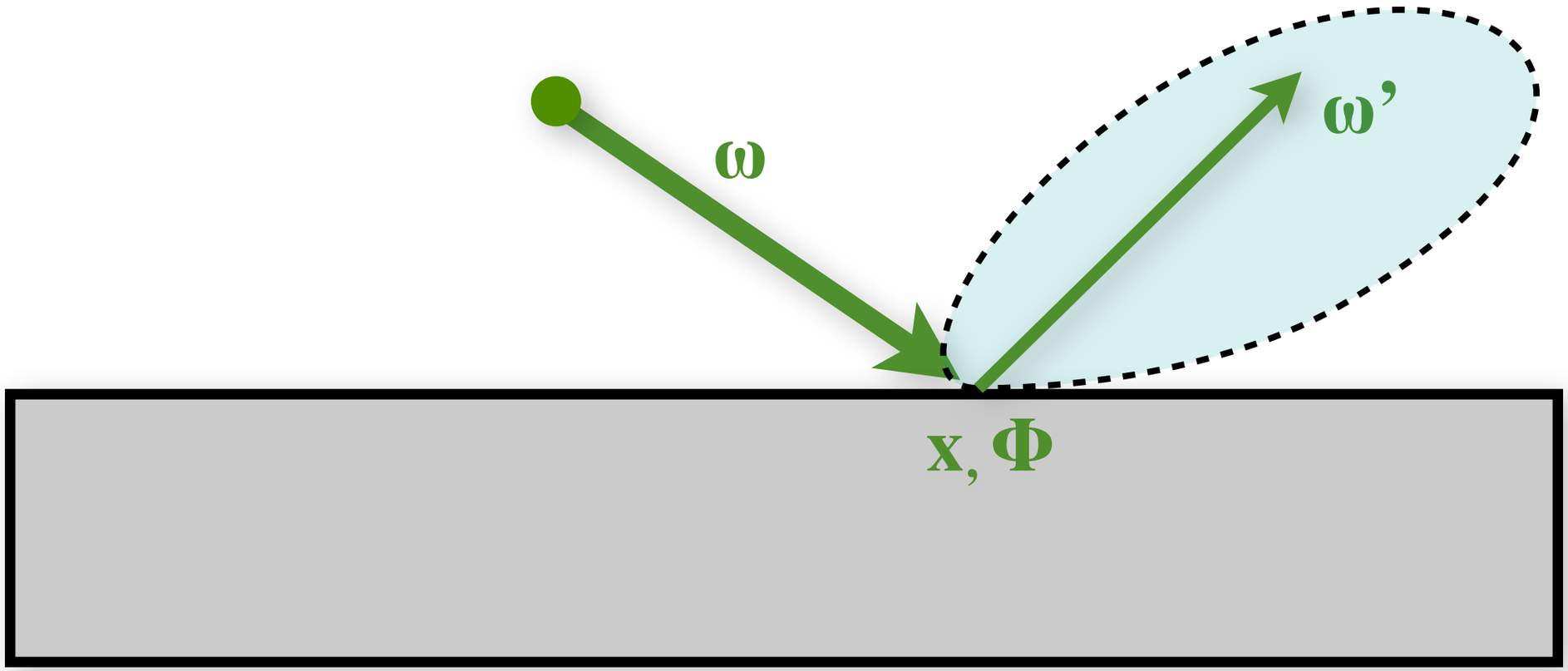
```
void generatePhotonMap()
    repeat:
        (l, Prob_l) = chooseRandomLight()
        (x, ω, Φ) = emitPhotonFromLight(l)
        tracePhoton(x, ω, Φ / Prob_l)
    until we have enough photons;
    divide all photon powers by number of emitted photons
```

Photon Tracing

```
void generatePhotonMap()
    repeat:
        (l, Prob_l) = chooseRandomLight()
        (x, ω, Φ) = emitPhotonFromLight(l)
        tracePhoton(x, ω, Φ / Prob_l)
    until we have enough photons;
    divide all photon powers by number of emitted photons

void tracePhoton(x, ω, Φ)
```

Photon Tracing




```
void tracePhoton(x, ω, Φ)
    s = nearestSurfaceHit(x, ω)
    x += s*ω        // propagate photon
    possiblyStorePhoton(x, ω, Φ)
    (ω’, pdf) = sampleBxDF(x, ω)
    return tracePhoton(o, ω’, Φ * BxDF(x,ω,ω’) / pdf)
```

Use Russian roulette!

Photon Mapping

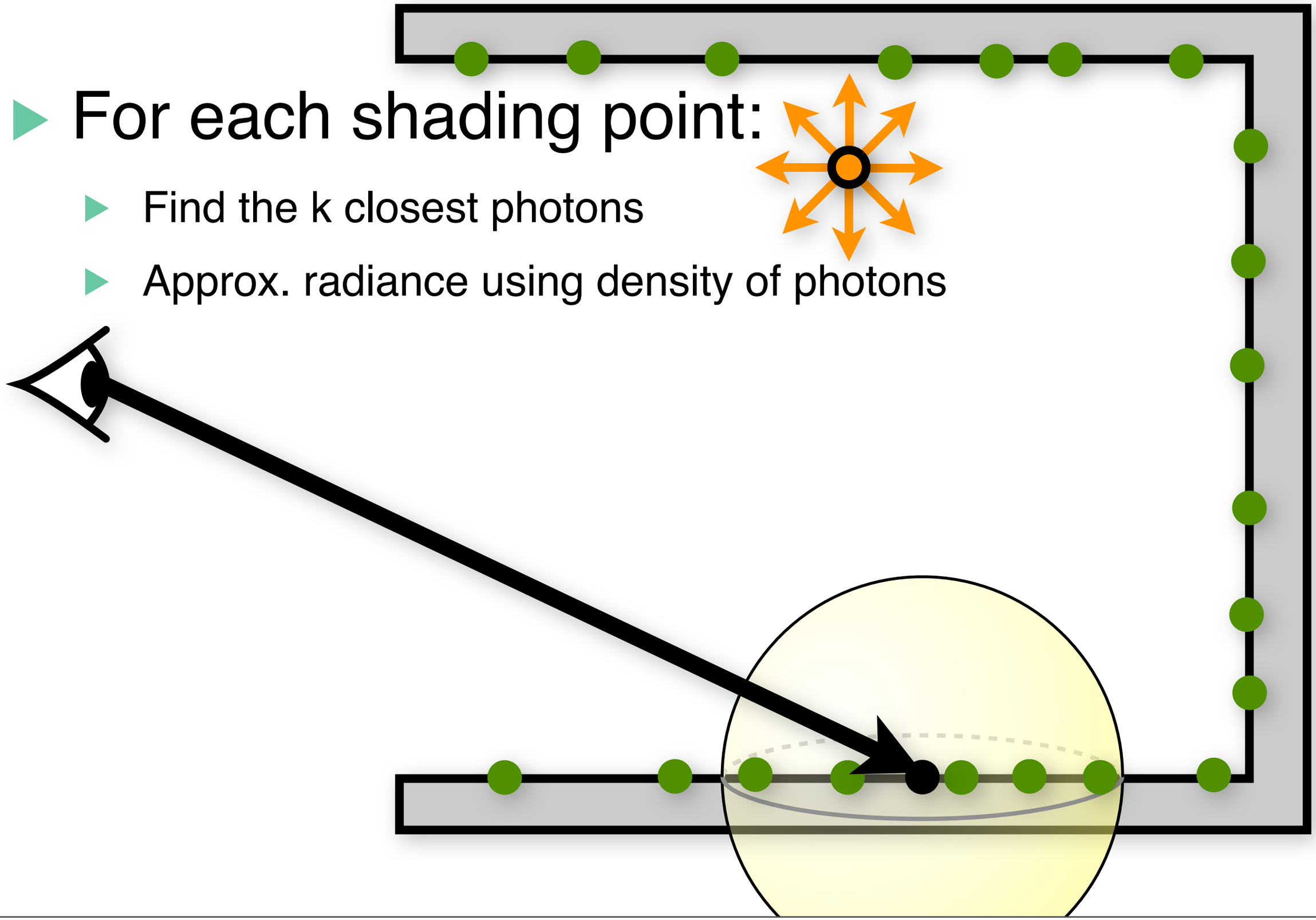
SIGGRAPHASIA2013

- A two-pass algorithm:
  - Pass 1: Trace virtual photons from the light source, scatter at surfaces, and cache
  - Pass 2: Ray trace the scene and use the photons to compute indirect illumination

Photon Mapping (Rendering)

SIGGRAPHASIA2013

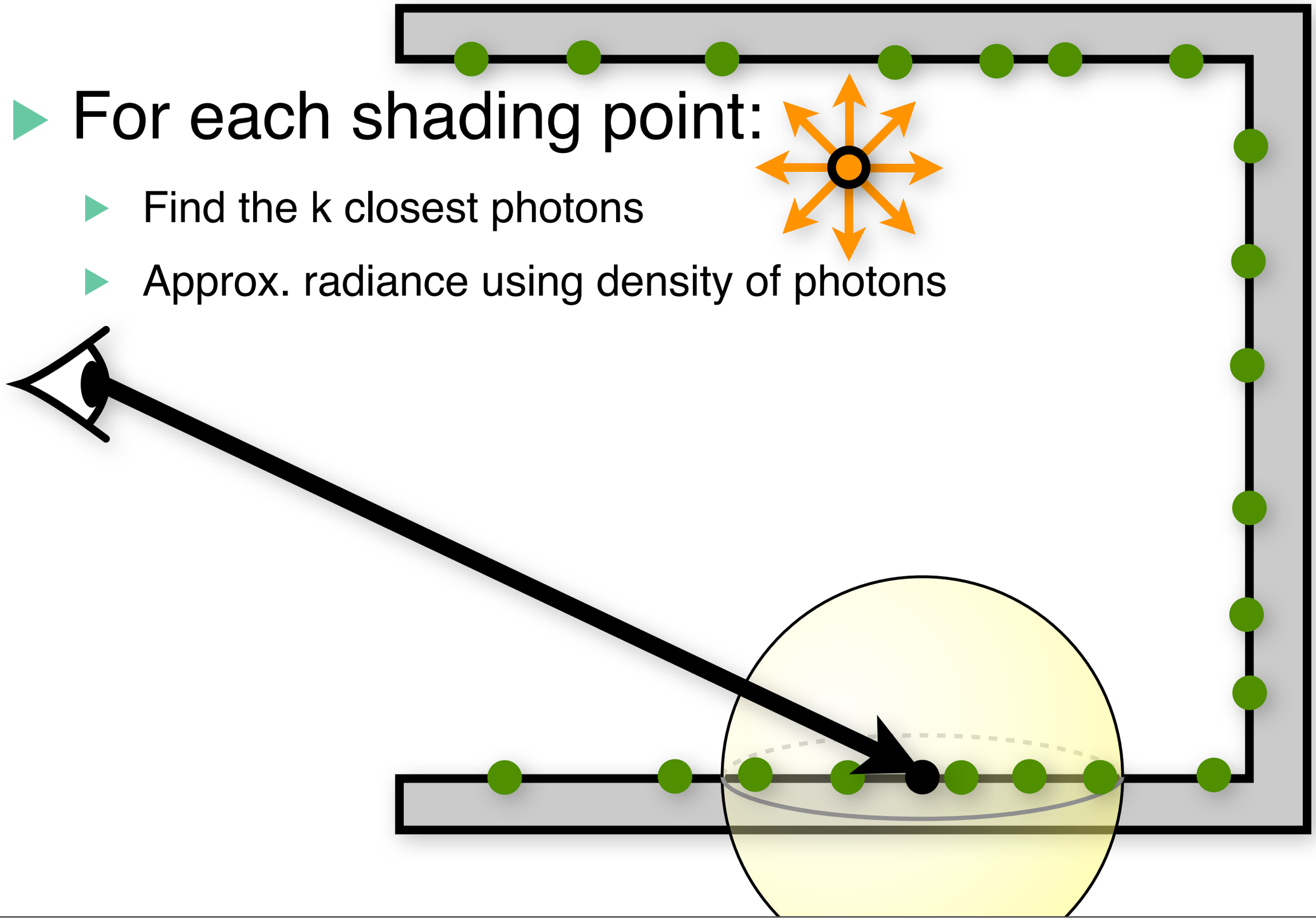
- For each shading point:
  - Find the k closest photons
  - Approx. radiance using density of photons



Photon Mapping (Rendering)

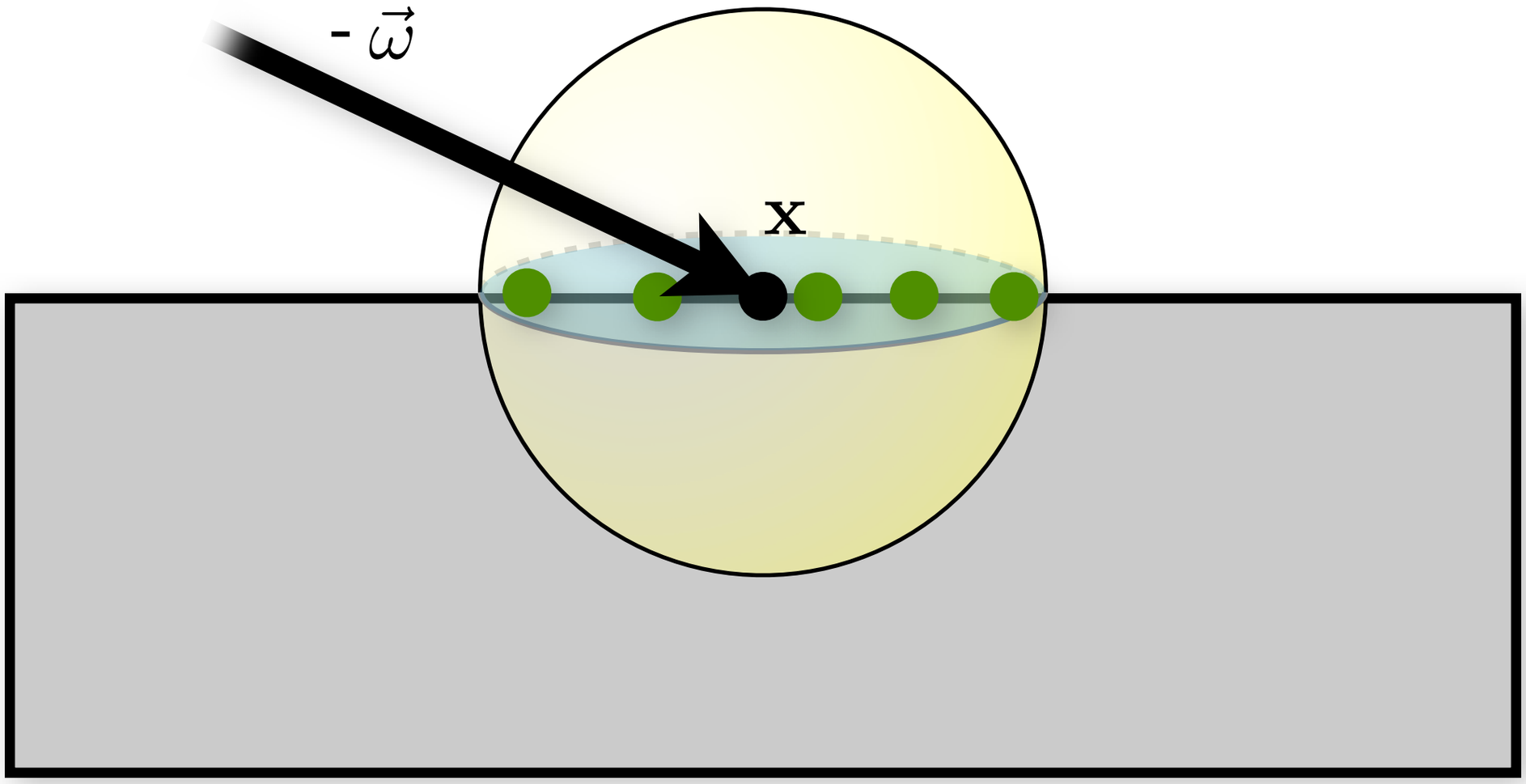
SIGGRAPHASIA2013

- For each shading point:
  - Find the k closest photons
  - Approx. radiance using density of photons



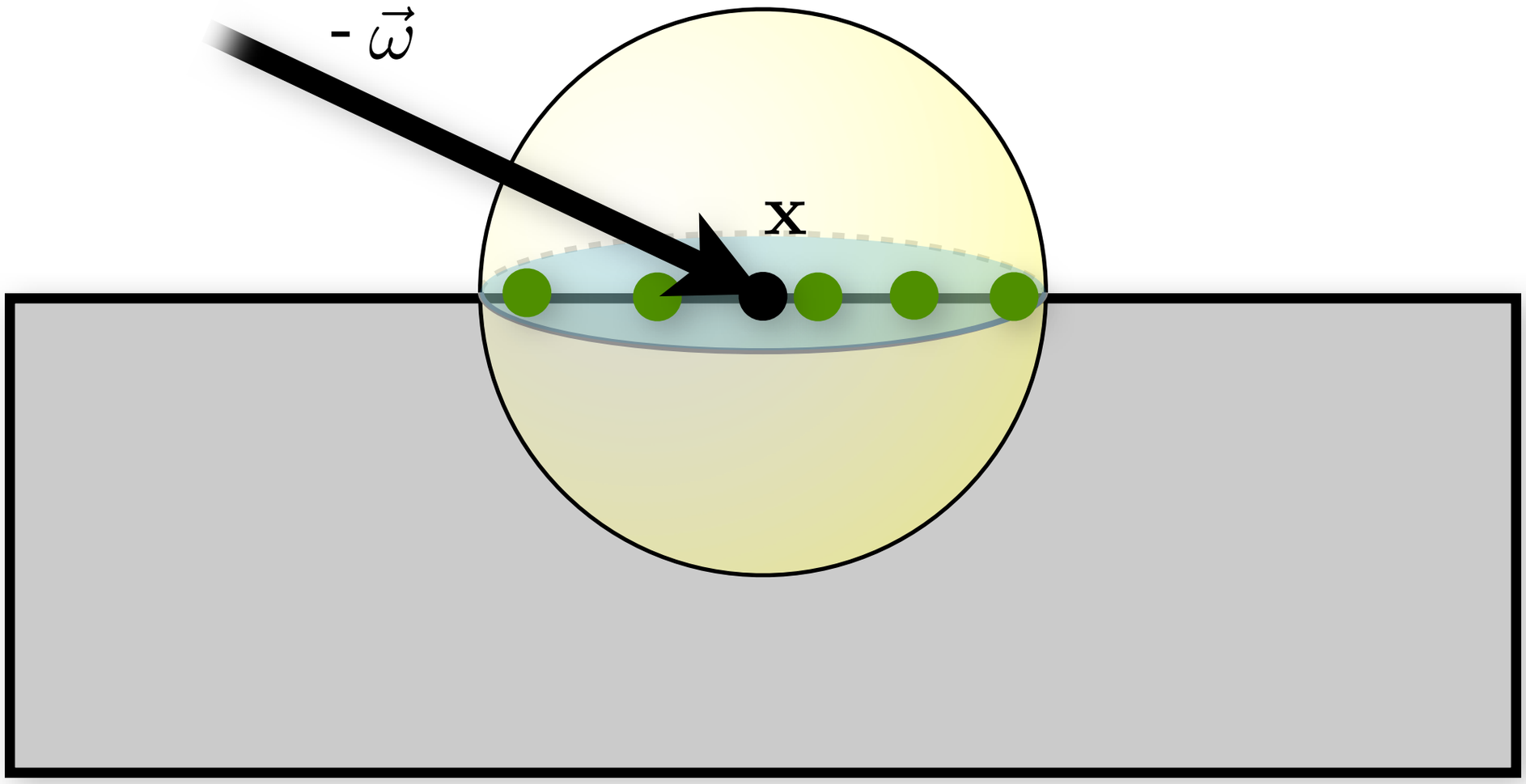
The Radiance Estimate

$$L_r(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{k-1} f_r(\mathbf{x}, \vec{\omega}_p, \vec{\omega}) \frac{\Phi_p}{A_k}$$



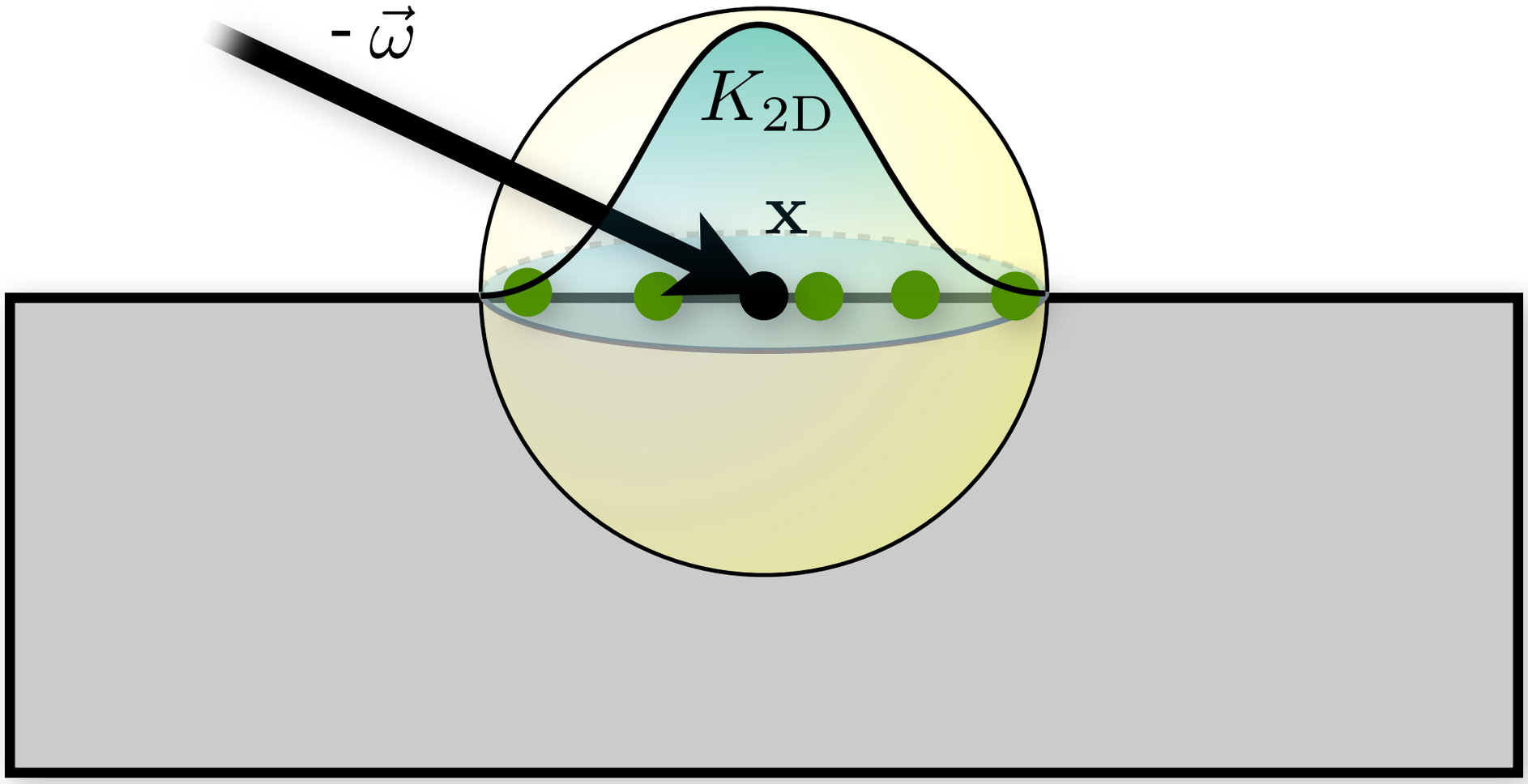
The Radiance Estimate

$$L_r(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{k-1} f_r(\mathbf{x}, \vec{\omega}_p, \vec{\omega}) \frac{\Phi_p}{\pi r_k^2}$$



The Radiance Estimate

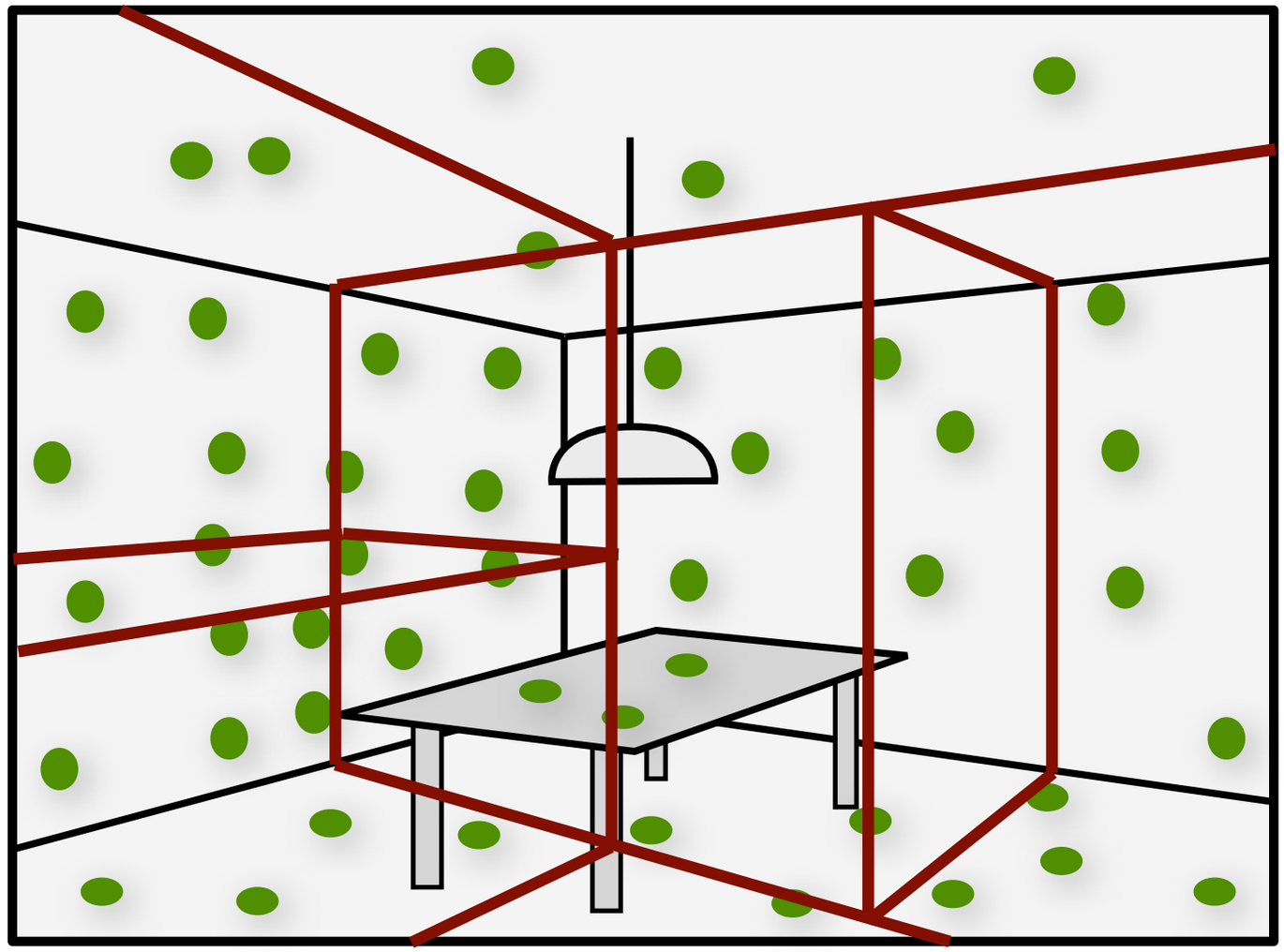
$$L_r(\mathbf{x}, \vec{\omega}) \approx \sum_{p=1}^{k-1} f_r(\mathbf{x}, \vec{\omega}_p, \vec{\omega}) \Phi_p K_{2\mathrm{D}}(r_p, r_k)$$



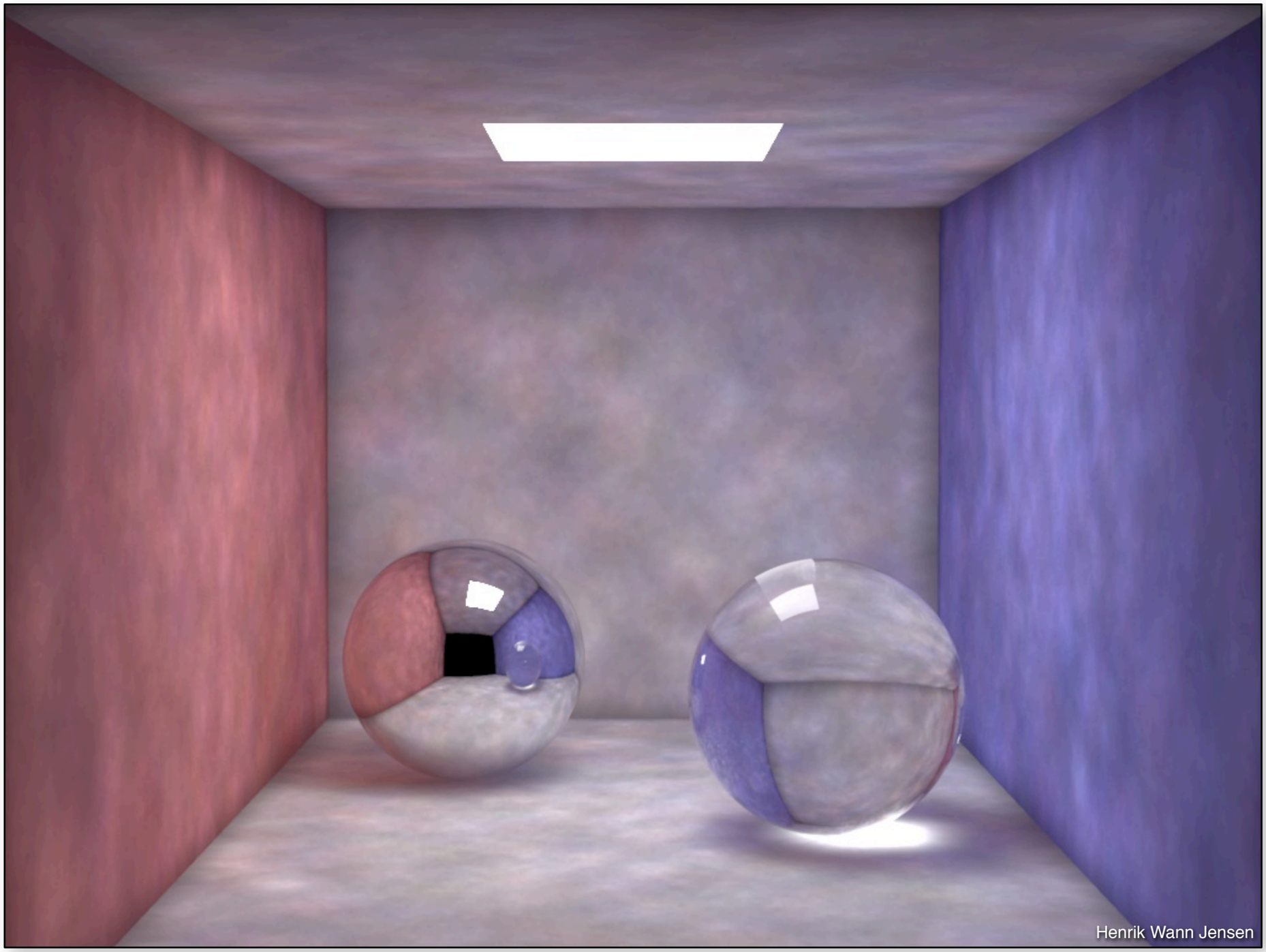
The Photon Map Data Structure

SIGGRAPHASIA2013

- Requirements:
  - Compact (we want many photons)
  - Fast nearest neighbor search
- KD-tree

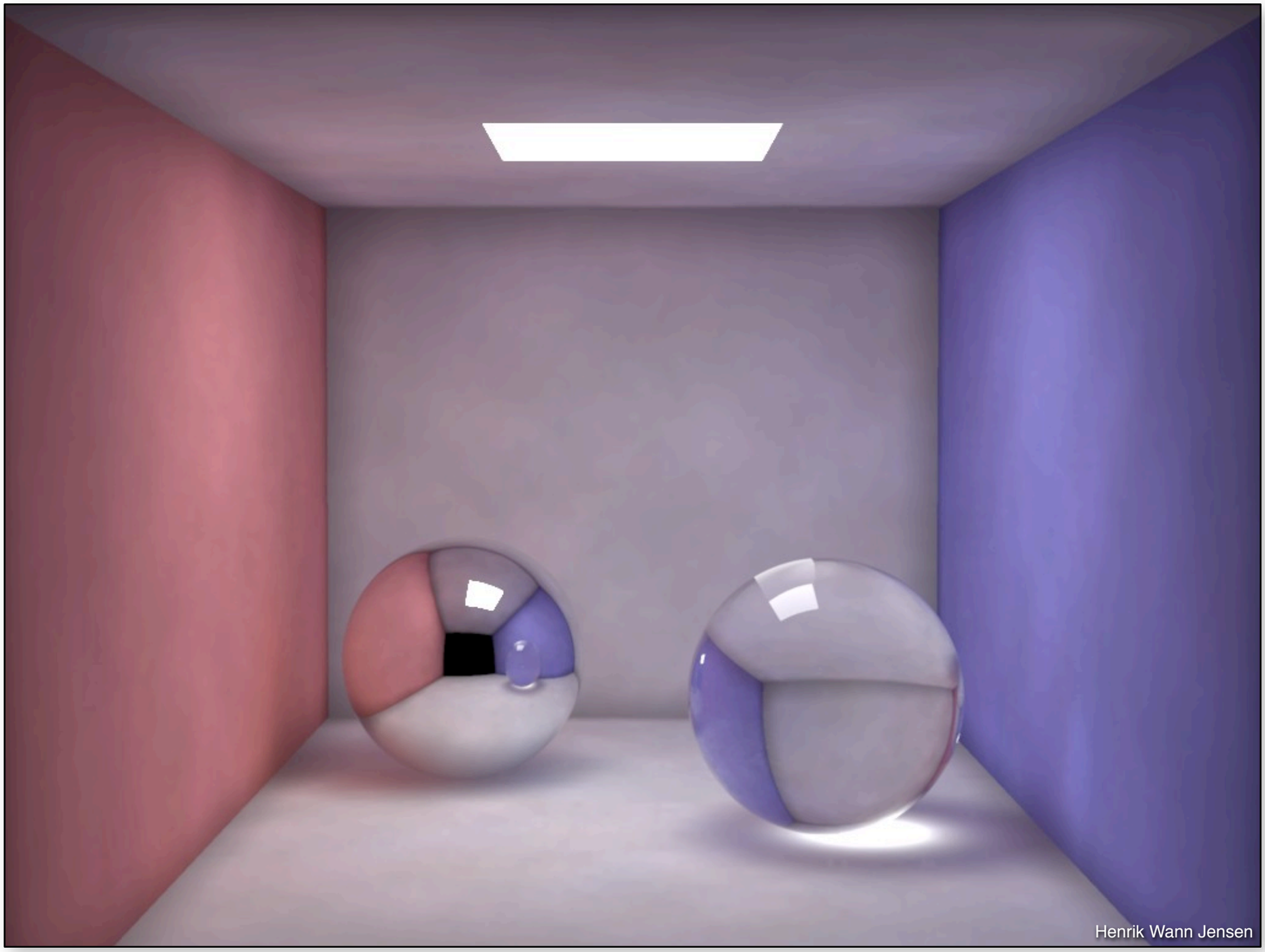


Global Illumination



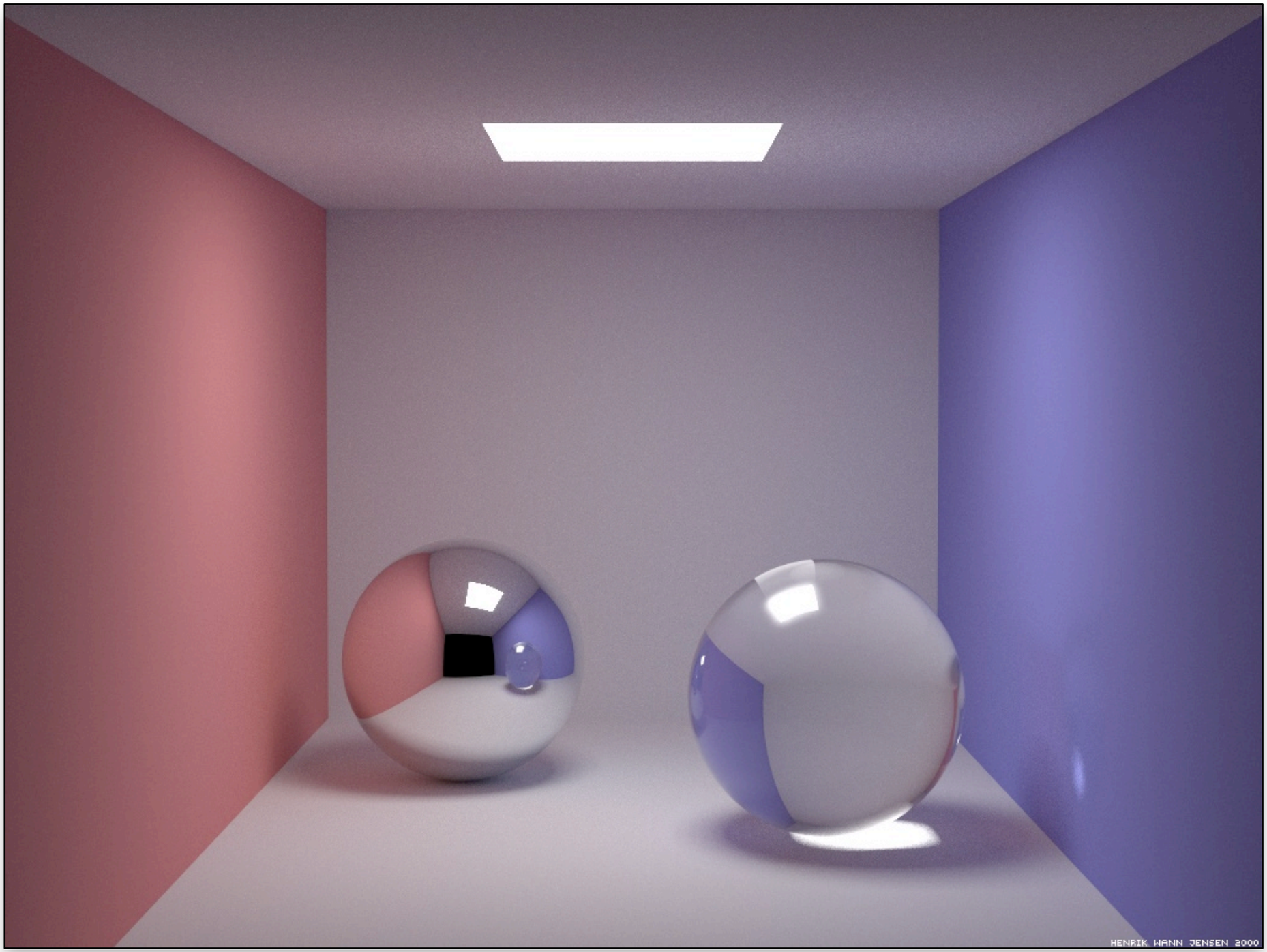
100000 photons / 50 photons in radiance estimate

Global Illumination



500000 photons / 500 photons in radiance estimate

Path Tracing



Practical Photon Mapping

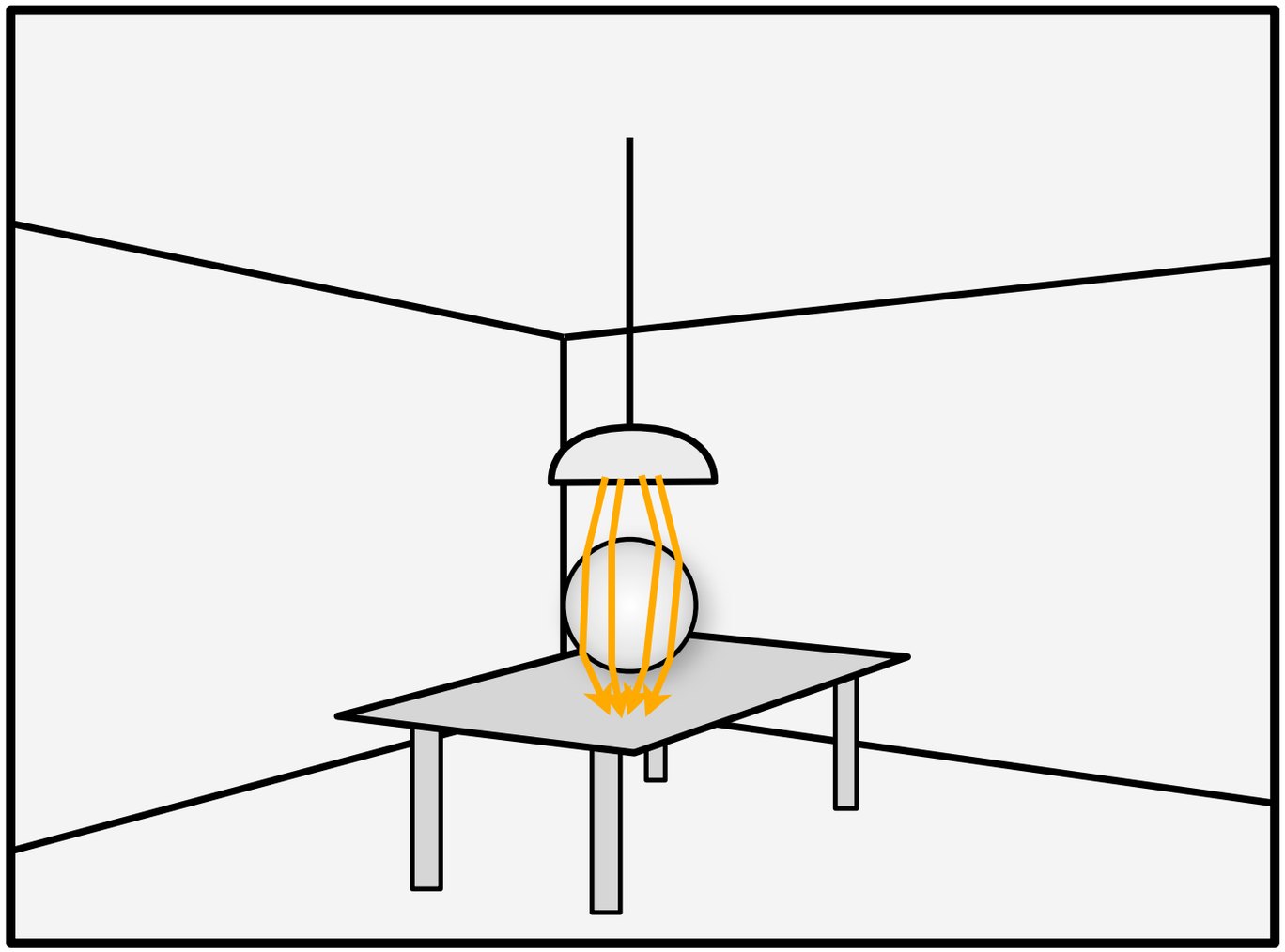
SIGGRAPHASIA2013

- Split illumination

Caustics

SIGGRAPHASIA2013

- Separate, higher quality photon map for caustics
  - Only stores LS^+D paths
  - Many photons shot directly at specular objects

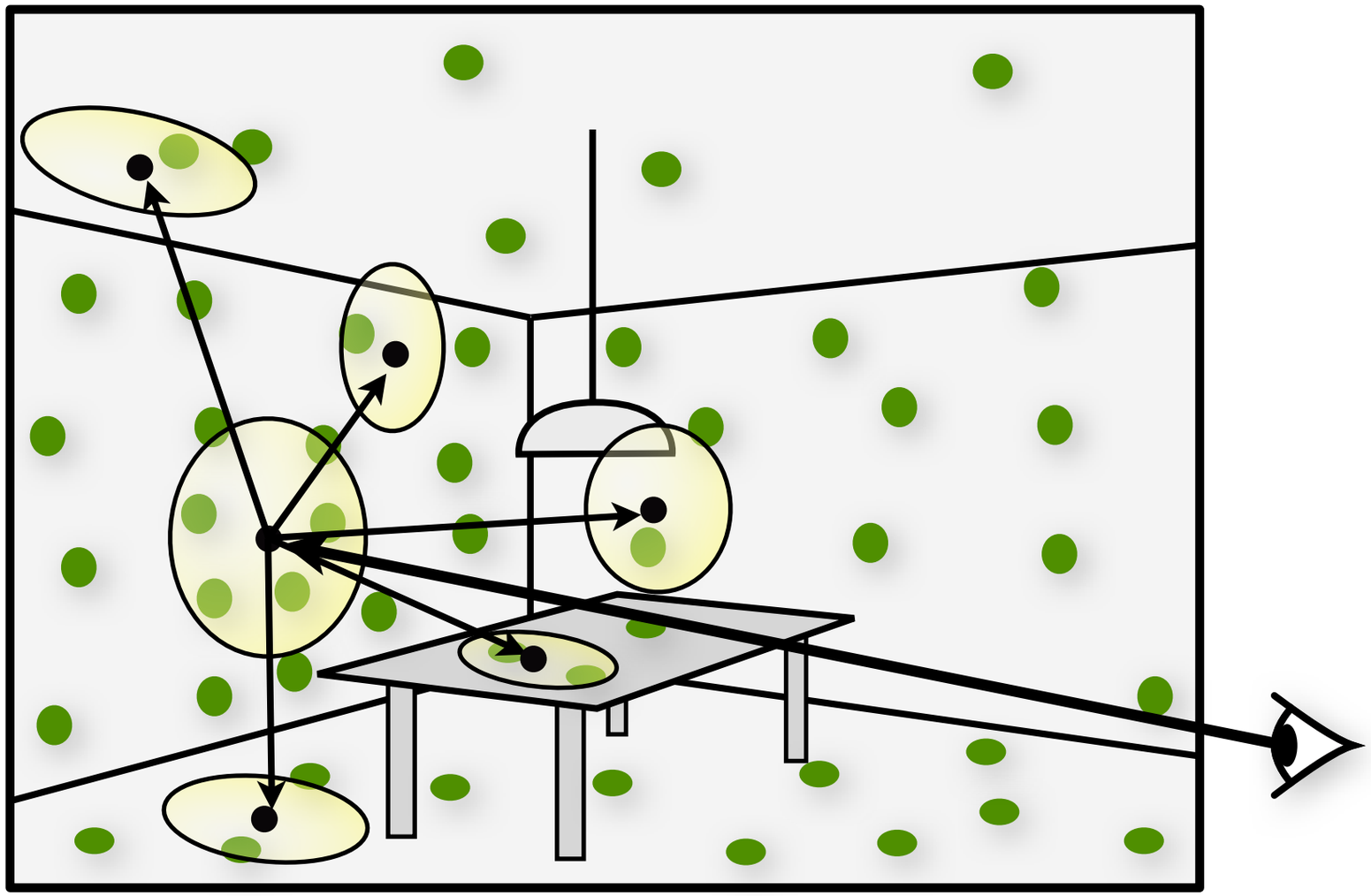


Diffuse Indirect

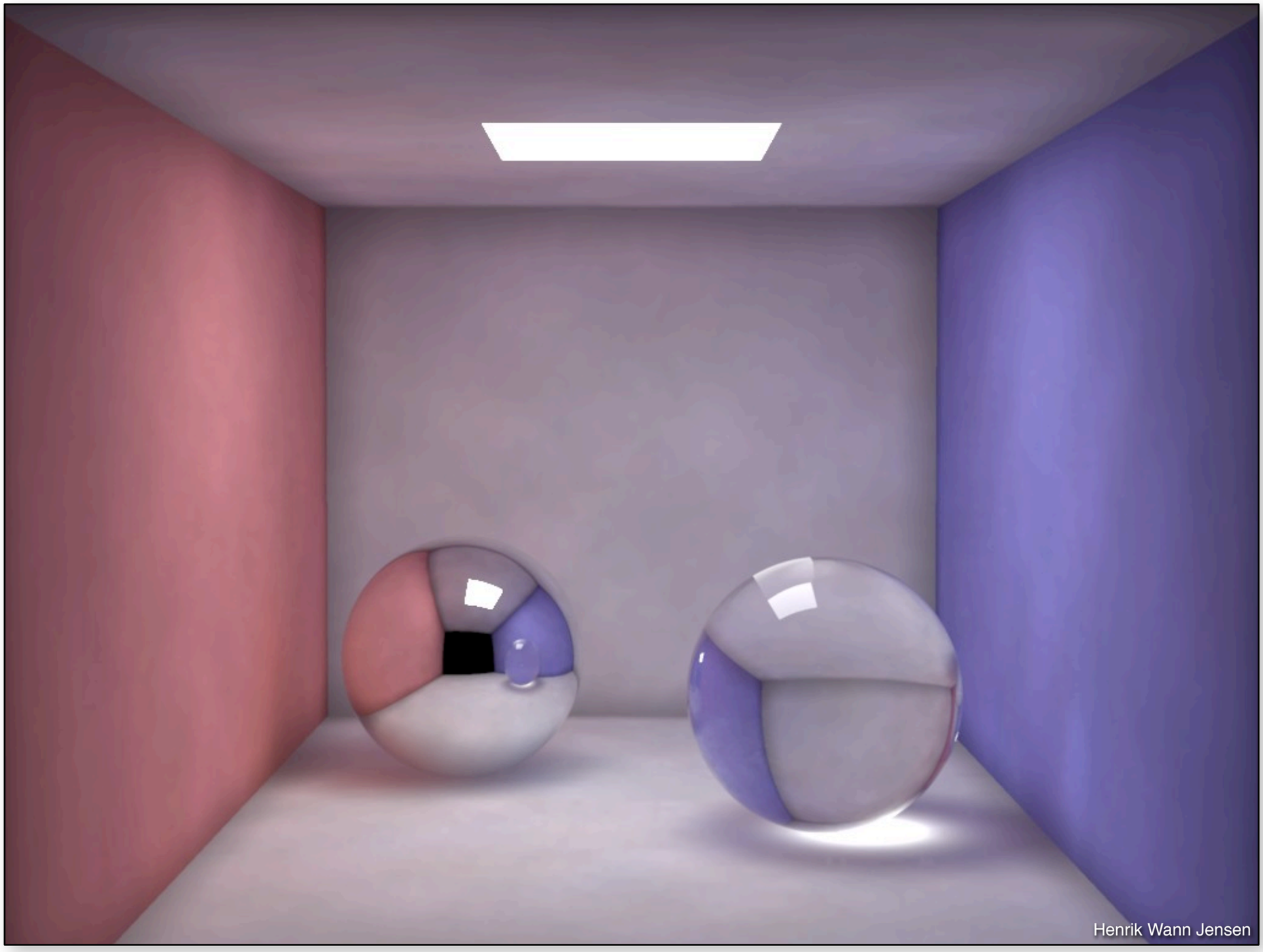
SIGGRAPHASIA2013

✘ Density estimation is blotchy

✓ Use final gather

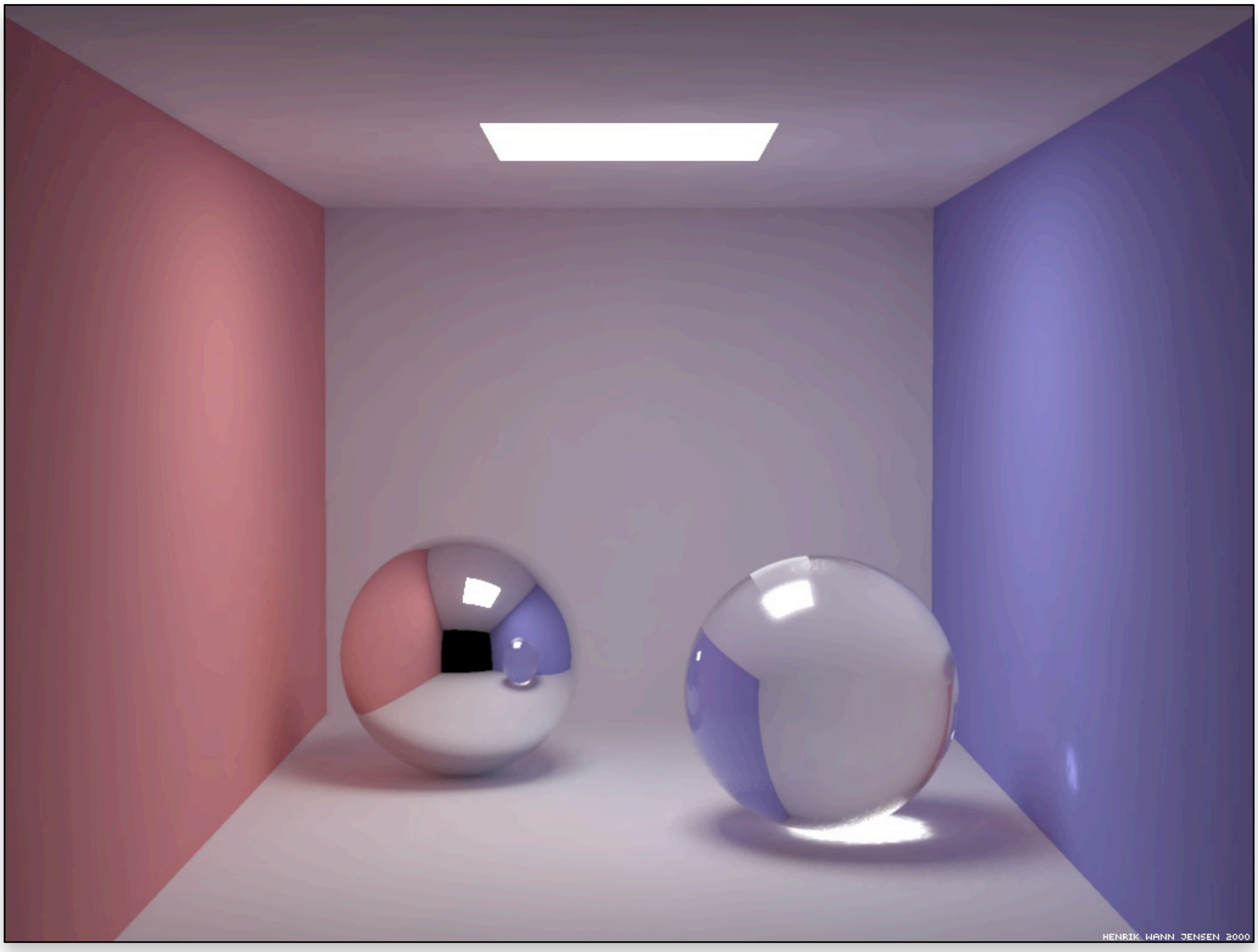


Global Illumination



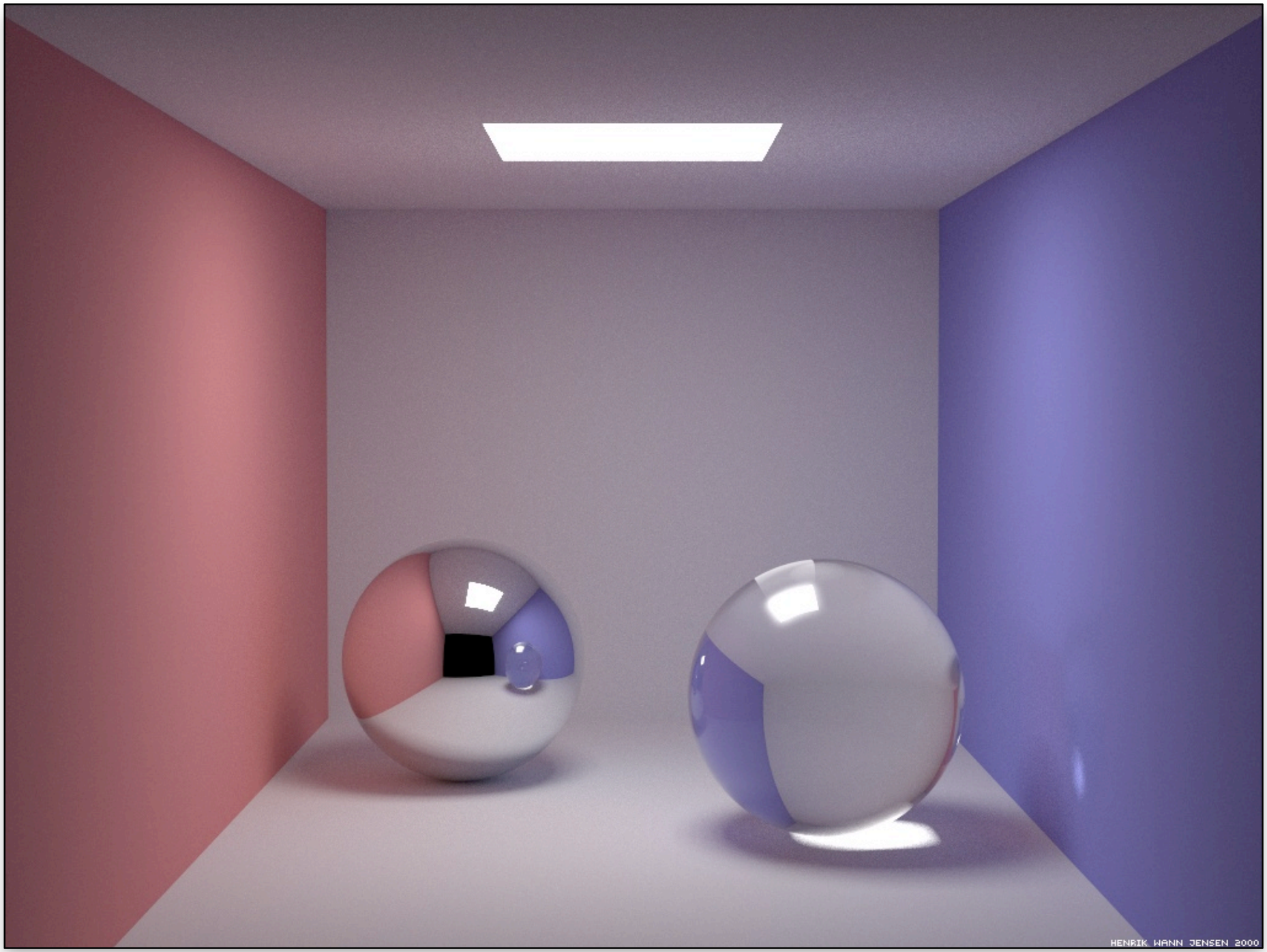
500000 photons / 500 photons in radiance estimate

Improved Photon Mapping



final gather + global photon map (200000) + caustic photon map (50000)

Path Tracing



Next...

- Improve / extend / generalize
- Progressive Photon Mapping