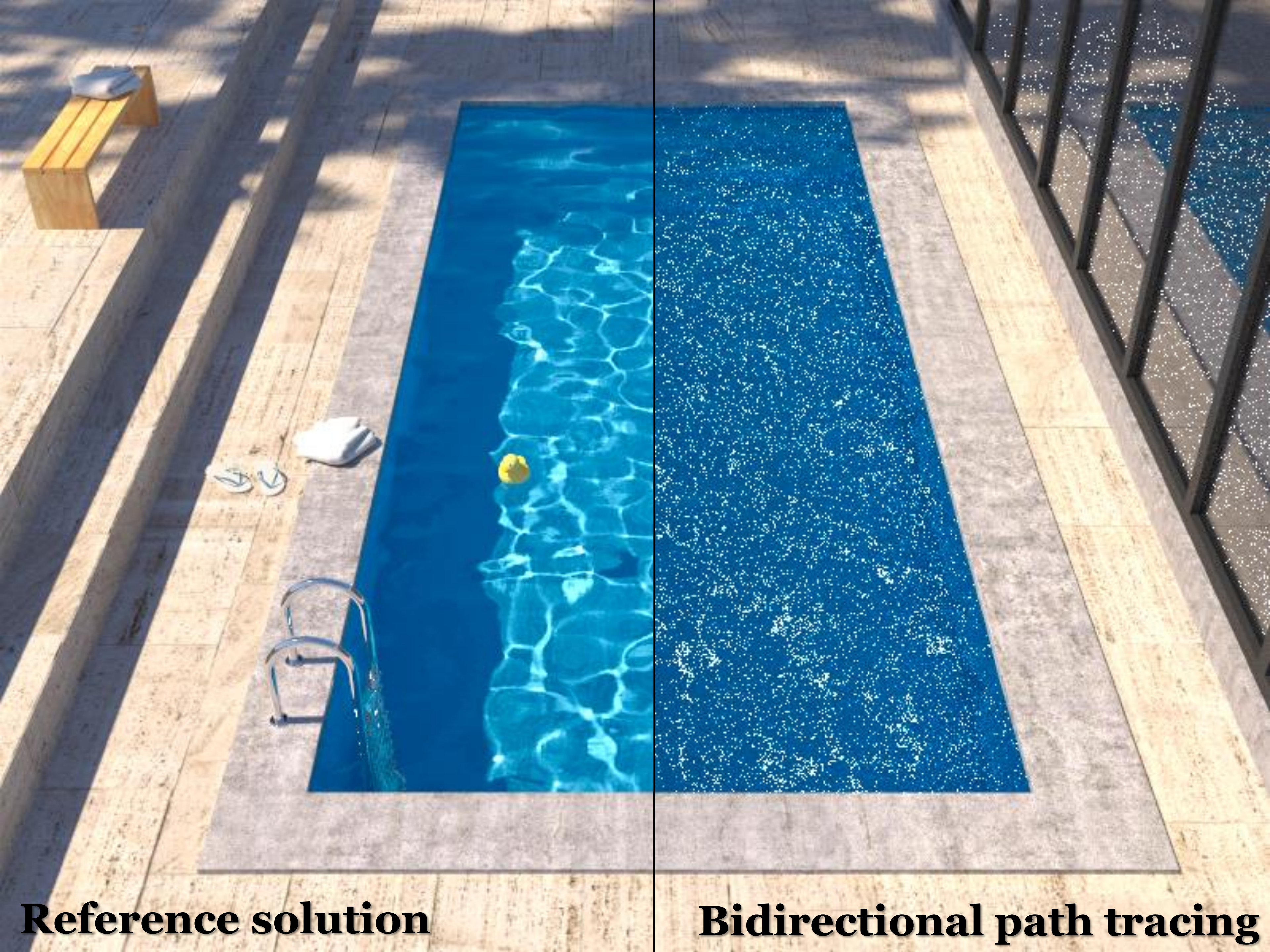
# Computer graphics III – Photon mapping

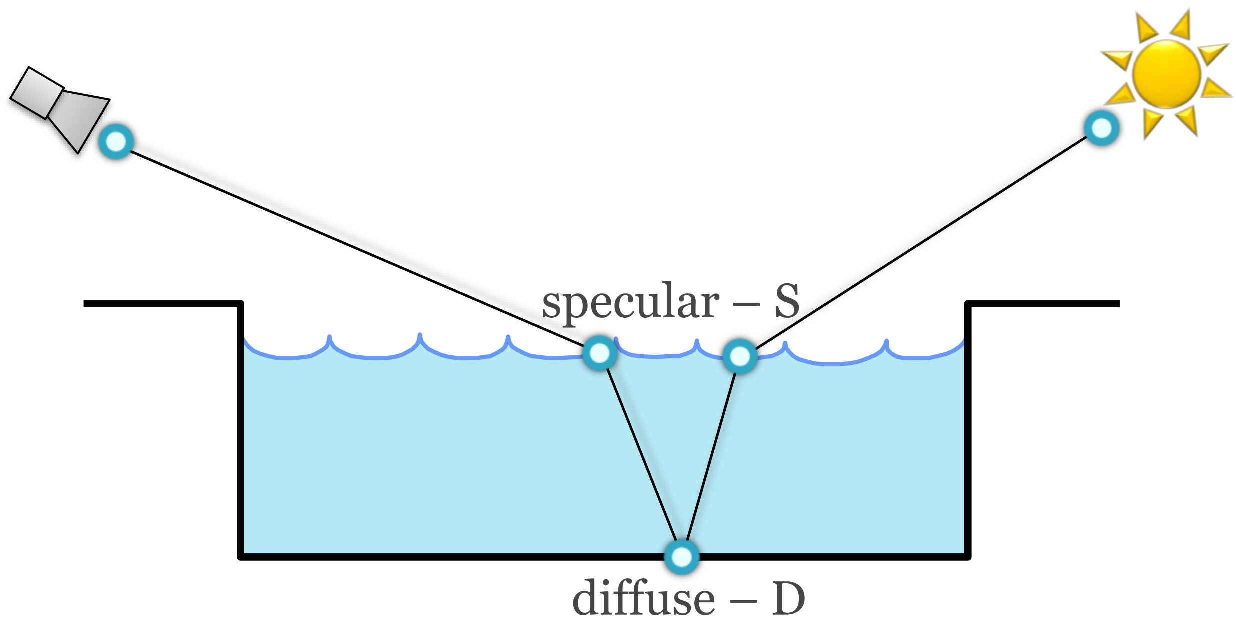
Jaroslav Křivánek, MFF UK

Jaroslav.Krivanek@mff.cuni.cz


Reference solution
Bidirectional path tracing

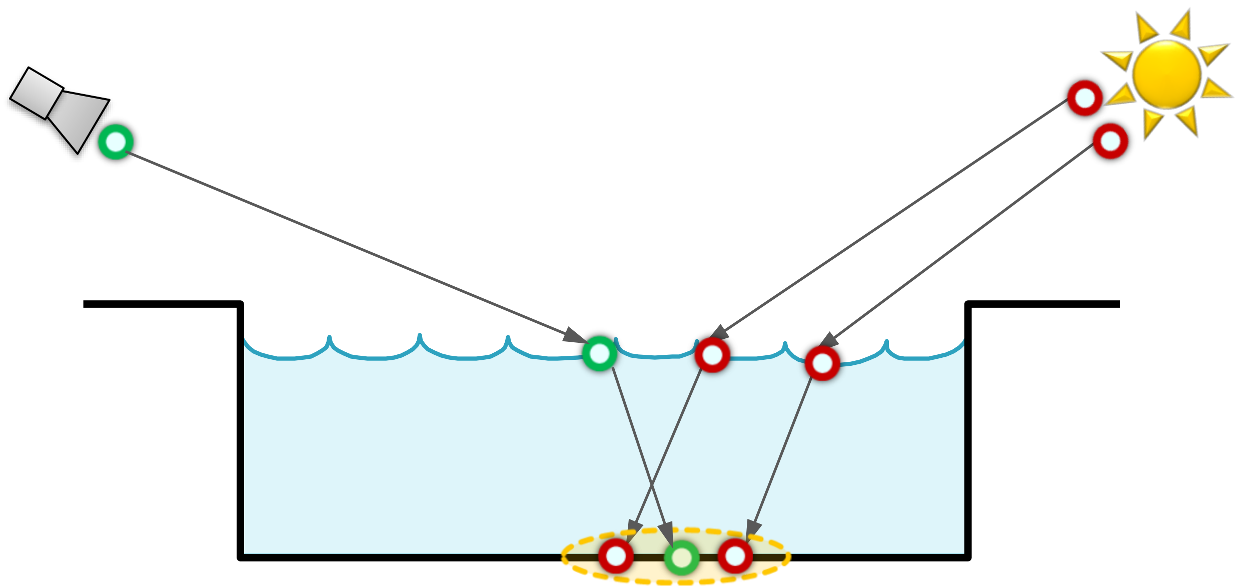
# Insufficient path sampling techniques

- Some paths sampled with zero (or very small) probability



# Photon mapping (Density estimation)

1. Many fwd walks + store particles ("photon map")
2. Radiance estimate: (Kernel) **density estimation**



# Photon mapping – SDS paths

© H.W.Jensen

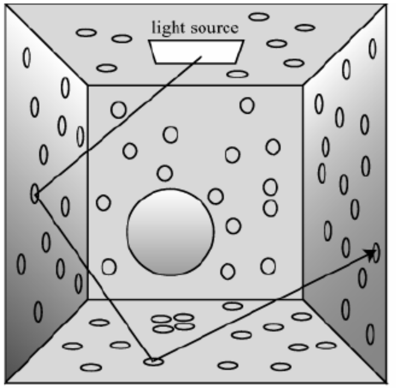
© Wojciech Jarosz

# Photon mapping overview

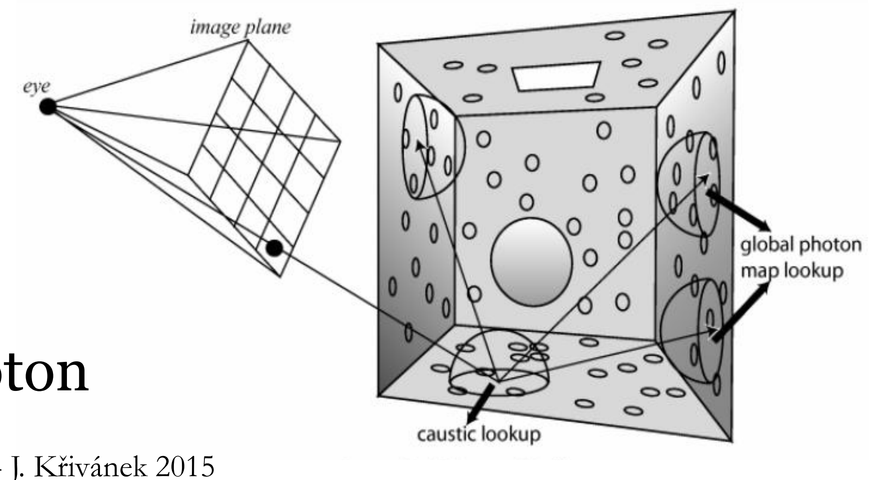
- Paths are followed both **from the light sources** and **from the camera**
- Similar to bidirectional path tracing
  - But the sub-path connection strategy significantly differs
- **Reuse of light sub-paths** for all pixels
  - Photon map = "**light sub-path cache**"
  - Essential for good performance
- For the same quality often faster than pure MC techniques
- **Biased!**
  - But **can be made consistent** (i.e. converges as the photon count increases, cf. **progressive photon mapping**)

# Calculation steps

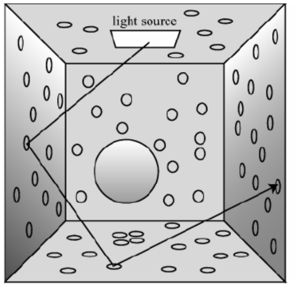
1. **Photon tracing**
   - "Photons" emitted from light sources,
   - traced through the scene (a la light tracing),
   - and stored in a photon map



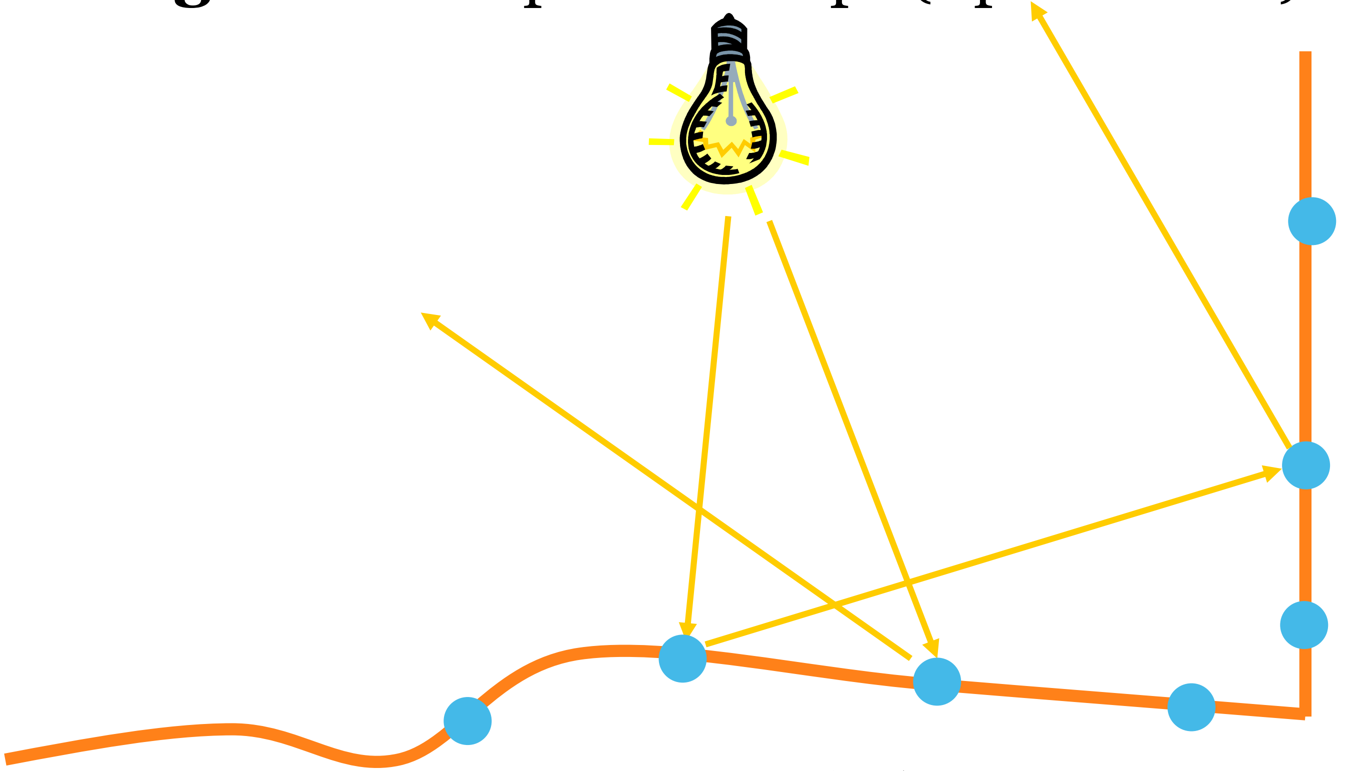

2. **Rendering with photon maps**
   - Similar to distributed path tracing
   - Recursion replaced by a photon map lookup



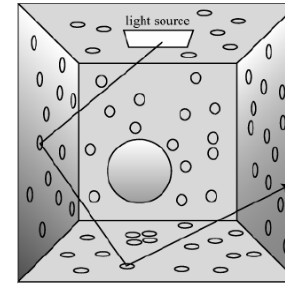
# Phase 1: Photon tracing




1. **Emission** of photons from light sources
2. **Tracing** of photon paths
3. **Storage** into the “photon map” (=photon list)



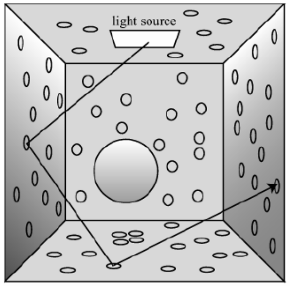
# Photon emission




- **Goal**
  - All emitted photons carry the same (or similar) flux (so that the variance of photon map radiance estimates is low)

1. **Emission** of a single photon (i.e. of a single sub-path)
   1. **Choose the light source**
      - Randomly with a probability proportional to its total flux
   2. **Choose the photon origin**
      - The light position for point sources
      - Randomly chosen position for area sources
   3. **Choose the photon direction**
      - Randomly according to the emission distribution of the source

# Photon emission




- Flux of the emitted photon:

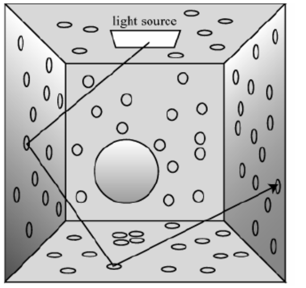
$$\Phi_{p,0} = \frac{1}{N} \frac{L_e(\mathbf{x}_0, \omega_0) \, |\cos \theta_0|}{P_l \; p(\mathbf{x}_0, \omega_0)}$$

Total number of emitted photon paths

(Discrete) probability of choosing the light source $l$

Pdf for sampling position $\mathbf{x}_o$ and direction $\omega_o$

# Photon emission



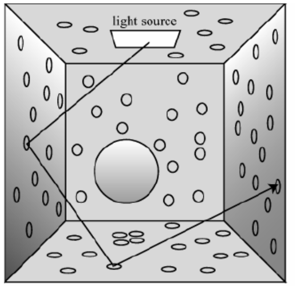

- "Ideal" sampling

$$p(\mathbf{x},\omega) = \frac{L_e(\mathbf{x},\omega)\,|\cos\theta|}{\int_{A_{light}}\int_{H(\mathbf{x})} L_e(\mathbf{x},\omega)\,|\cos\theta|\,\mathrm{d}\omega\,\mathrm{d}A} = \frac{L_e(\mathbf{x},\omega)\,|\cos\theta|}{\Phi_l}$$

$$P_l = \frac{\Phi_l}{\sum_{i\in lights}\Phi_i} = \frac{\Phi_l}{\Phi_{total}}$$

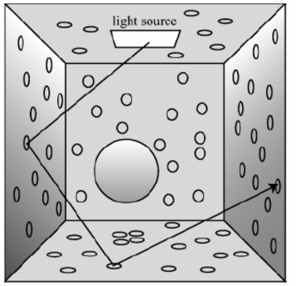
  - All emitted photons carry the same flux: $\Phi_{p,0} = \frac{\Phi_{\text{total}}}{N}$

# Tracing of photon paths




- Similar to light tracing
- Photon-surface intersection:
  1. **Store "photon"** into a photon map
     - photon = (position, incident direction, flux)
  2. **Generate reflected direction**
     - BRDF importance sampling
  3. **Update photon flux**
     - (next slide)
  4. **Russian roulette** – randomized absorption (termination)
     - (next slide)
- **Objective**
  - Keep the photon flux close to its original value

# Photon tracing




3. **Update photon flux**

$$\Phi_{p,j+1}^{tentative} = \Phi_{p,\,j} \frac{f_r(\mathbf{x}, \omega_o \to \omega_i) \left|\cos\theta_o\right|}{p(\omega_o)}$$

4. **Russian roulette** – randomized photon absorption

$$q_{p,j+1} = \min\left\{1, \frac{\max_{r,g,b}[\Phi_{p,j+1}^{tentative}]}{\max_{r,g,b}[\Phi_{p,j}]}\right\}$$

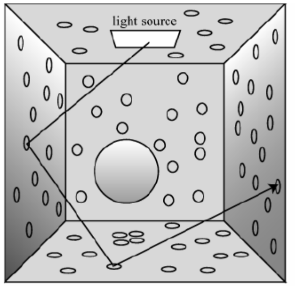
Survival probability

$$\Phi_{p,j+1} = \frac{\Phi_{p,j+1}^{tentative}}{q_{p,j+1}}$$

Updated photon flux on survival

- The above strategy keeps the photon flux roughly constant

# Photon tracing

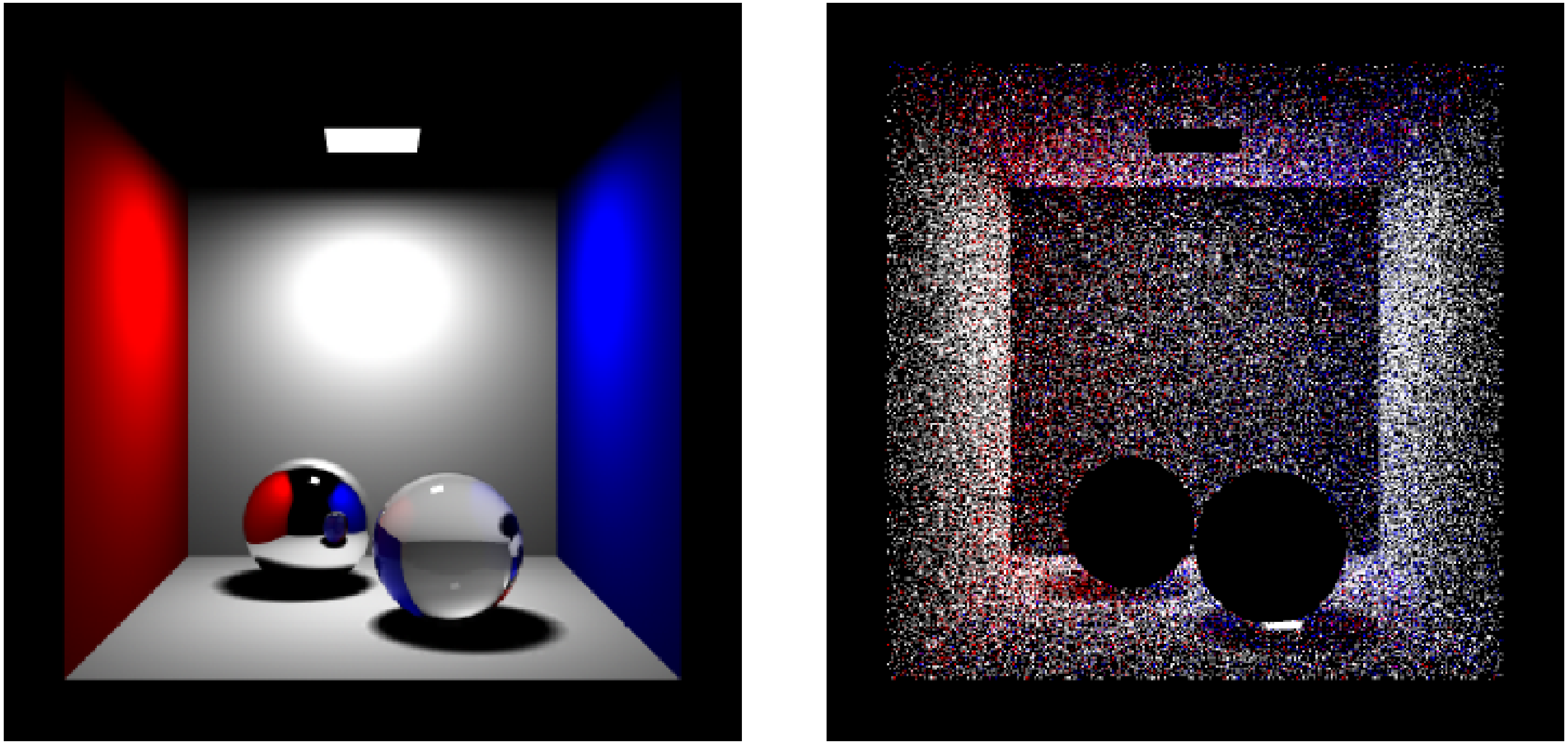



- Attention to **light refraction**
  - Recall: When tracing **paths from the camera**, we need to **update radiance** according to the 2nd power of the relative IOR
  - But photon do not carry radiance but flux – **no flux change upon refraction**

# Photon map

- **Storage of photons** into a photon map
  - Upon each interaction of a photon with a diffuse (or moderately glossy, but not mirror) surface (even on absorption)
- **Photon map**
  - A simple linear list of photons during photon tracing
  - After photon tracing, we build a *kD*-tree for faster search
- **Photon**
  - position: $\mathbf{x}_p = (x, y, z)$
  - incident direction: $\omega_p = (\theta, \phi)$
  - energy (flux): $\Phi_p = (r, g, b)$
- Number of photons: $10^6 - 10^7$ sufficient in many scenes

# Photon map

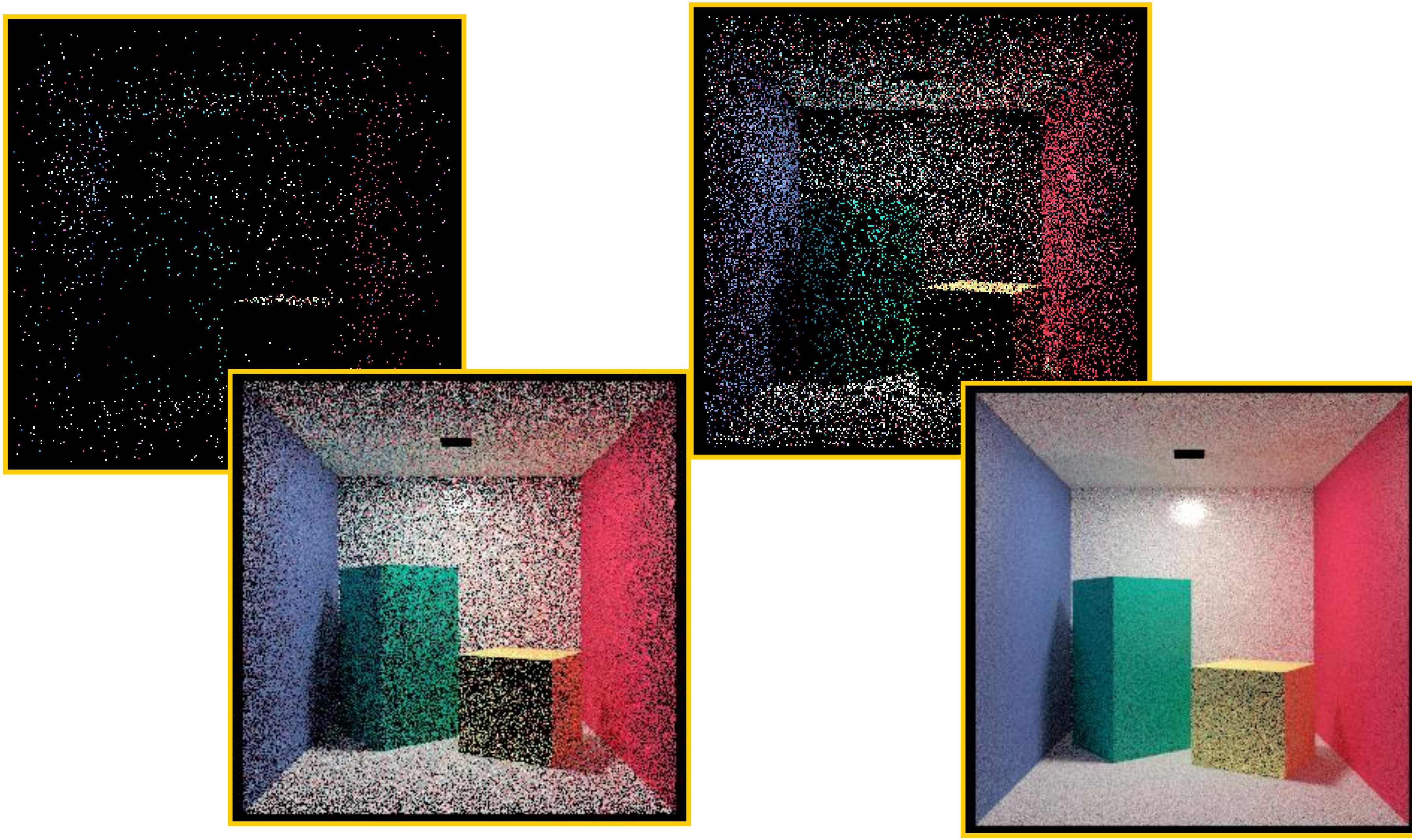


(a) (b)

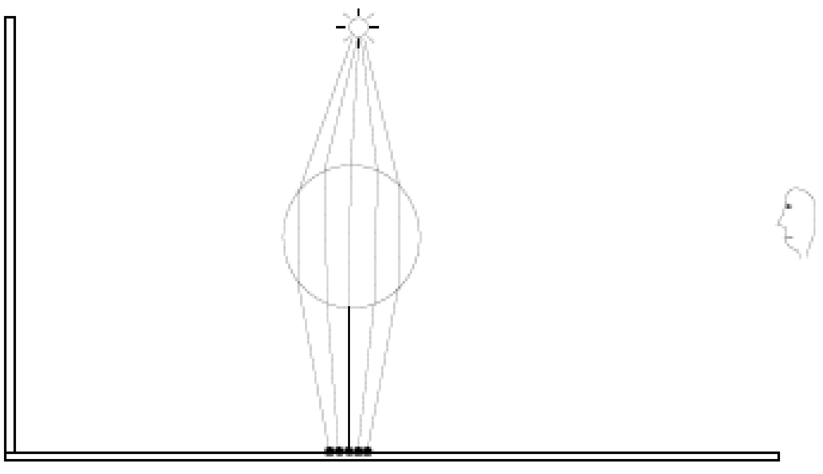
Figure 2.4: "Cornell box" with glass and chrome spheres: (a) ray traced image (direct illumination and specular reflection and transmission), (b) the photons in the corresponding photon map.

Image: Henrik Wann Jensen

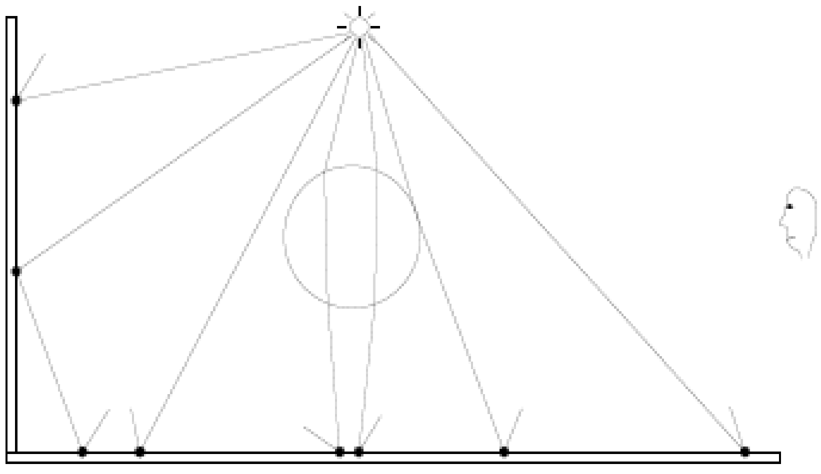
# Photons represent the equilibrium radiance in the scene



# Two photon maps



Caustics map



Global map

# Two photon maps

1. **Global map: L[S|D]*D**
   - Contains even direct illumination
2. **Caustics map: $LS^{+}D$**
   - Contains indirect illumination only
   - Is a subset of the global map

- Different use of the two maps in image rendering
  - It's more advantageous to keep them separate

- **Light path grammar**

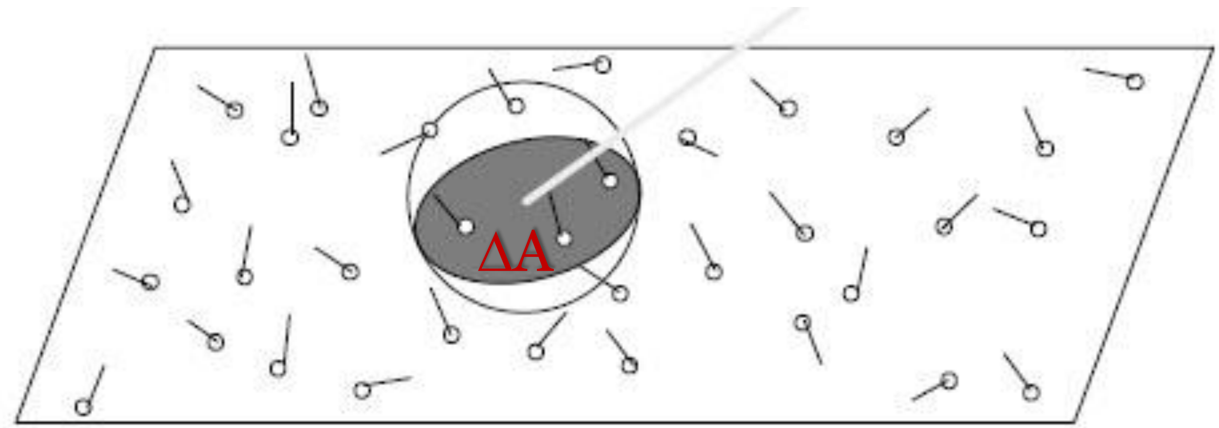
  E … eye, L … light, D … diffuse, S … specular

  G … glossy (often included in D)

# Getting the photon maps ready for rendering

- During photon tracing, photons are simply appended into a linear list
- After that, we build a **spatial search acceleration structure**
  - In rendering we need to quickly locate *k* nearest photons
  - **kD-tree** or **hashed uniform grid**

# Radiance estimate from a photon map




*k* .. #photons around **x**

incident direction of photon p

$$L_r(\mathbf{x},\omega_o) \approx \sum_{p=1}^{k} f_r(\mathbf{x},\omega_p,\omega_o)\frac{\Phi_p(\mathbf{x},\omega_p)}{\Delta A}$$

$$\approx \frac{1}{\pi r^2}\sum_{p=1}^{k} f_r(\mathbf{x},\omega_p,\omega_o)\Phi_p(\mathbf{x},\omega_p)$$

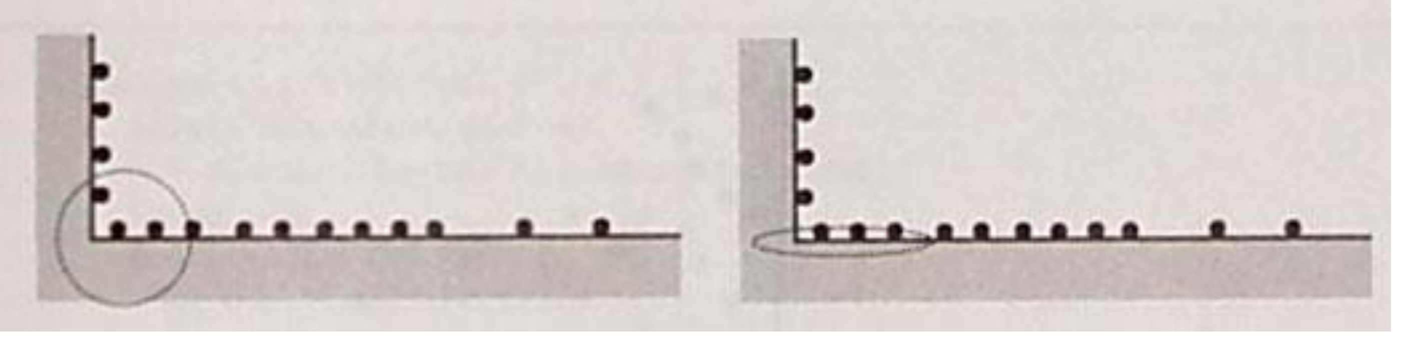
Surface area of ΔA

# Radiance estimate from a photon map

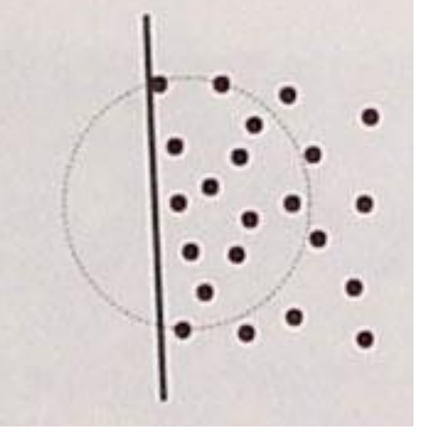
```
RadianceEstimate(x, wo):
    Color L = (0,0,0);
    int k = locateNearestPhotons(x, wo, n_max, nearest, r);
    // 'nearest' is an array of k nearest photons to x
    // r is the distance from x to the farthest of them
    if ( k < 5 ) return L;
    for p = 1 to k do
    {
        if( dot ( nearest[p].wi, N) <= 0 ) continue;
        L += fr(x, wo, nearest[p].wi) * nearest[p].flux;
    }
    return L / (M_PI * r*r);
```

# Radiance estimate – issues

- Incorrect photons included in the search



- Incorrect estimate of the surface area ΔA
  - Next to a wall,
    a caustic or geometry edge



# Fast search of nearest phtons

- Needed for the radiance estimate
- Search of the nearest photons is an instance of

**k-nearest neighbor search (k-NN)**

# *k*-D tree – Construction

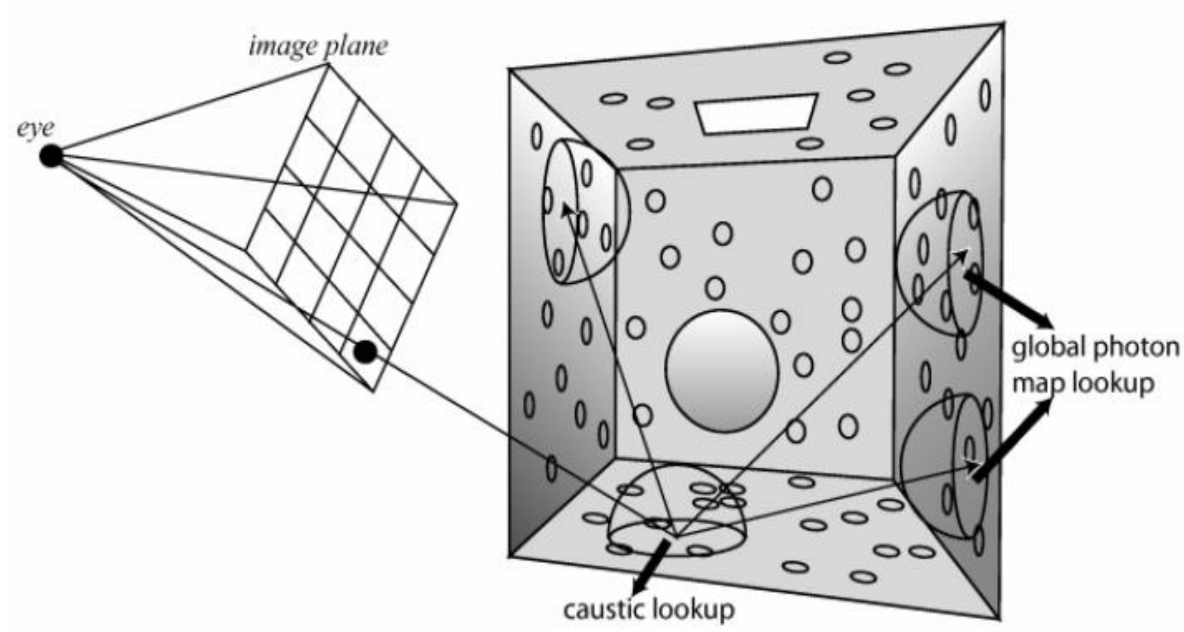
- Recursive space subdivision along the axis with maximum span
- Subdivision
  - Splitting plane can be in the **spatial center** (faster, ok) or through the **median of photons**
- When using the median split rule, the resulting tree is perfectly balanced and can be stored in a linear array
  - Descendants of the photon at index $i$ are at indexes $2i$ a $2i+1$

# *k*-D tree – Nearest neighbor search

- Pruning of the search
  - Either: According to the distance to the already located *k*-th nearest photon (when searching *k* nearest)
    - Photons located so far are maintained in a max-heap
  - Or: According to the search radius *r* (when locating particles witin a fixed radius – „range query“)

# Phase 2: Rendering with photon maps

- **Distributed ray tracing** from the camera
  - Recursion replaced by a photon map lookup
  - For highly specular surfaces we still use recursion as in classic path tracing



# Reflected radiance calculation

- Reflected radiance: this is what we want to calculate

$$L_r(\mathbf{x}, \omega_o) = \int_\Omega L_i(\mathbf{x}, \omega_i)\, f_r(\mathbf{x}, \omega_i \to \omega_o) \cos\theta_i \,\mathrm{d}\omega_i$$

- Split the incoming radiance

$$L_i = L_{i,d} + L_{i,c} + L_{i,l}$$

- Split the BRDF

$$f_r = f_{r,D} + f_{r,S}$$

| | $f_{r,D}$ | $f_{r,S}$ |
|---|---|---|
| $L_{i,d}$ | direct illumination | highly specular reflections / refractions |
| $L_{i,c}$ | caustics **(PM)** | |
| $L_{i,l}$ | diffuse indirect **(FG +PM)** | |

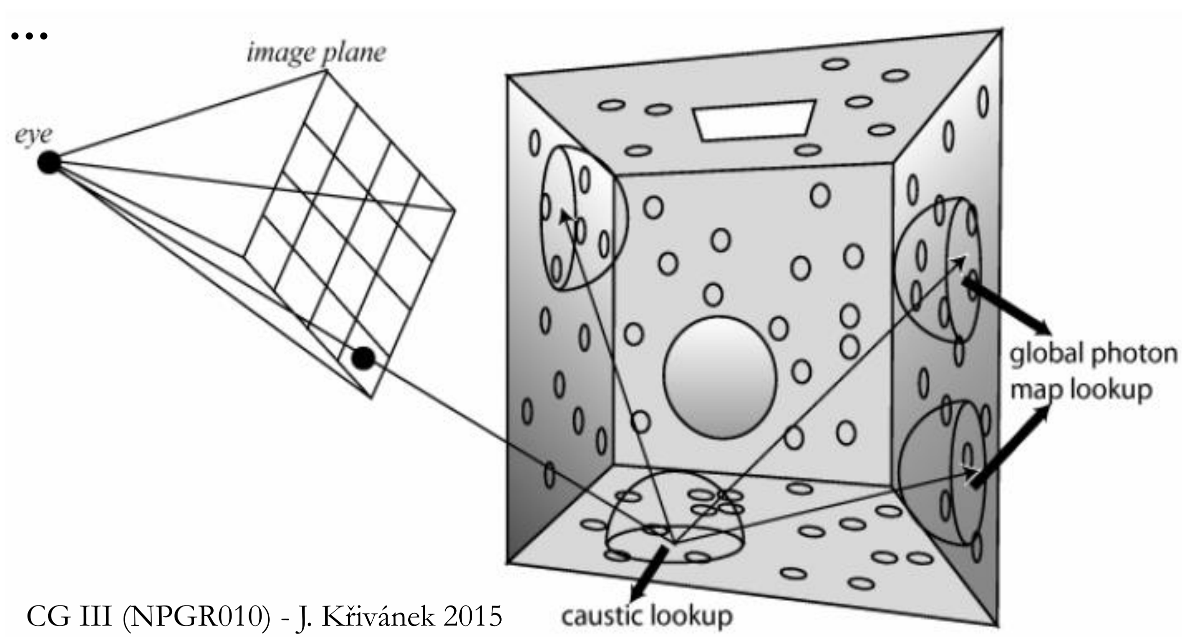
PM … photon map
FG … final gathering

# Reflected radiance calculation

- When not using photon maps
  - Direct illumination
    - As usual: light source area sampling + shadow rays
  - Ideal mirror reflections / refractions
    - As usual: deterministic secondary rays
- With photon maps
  - …

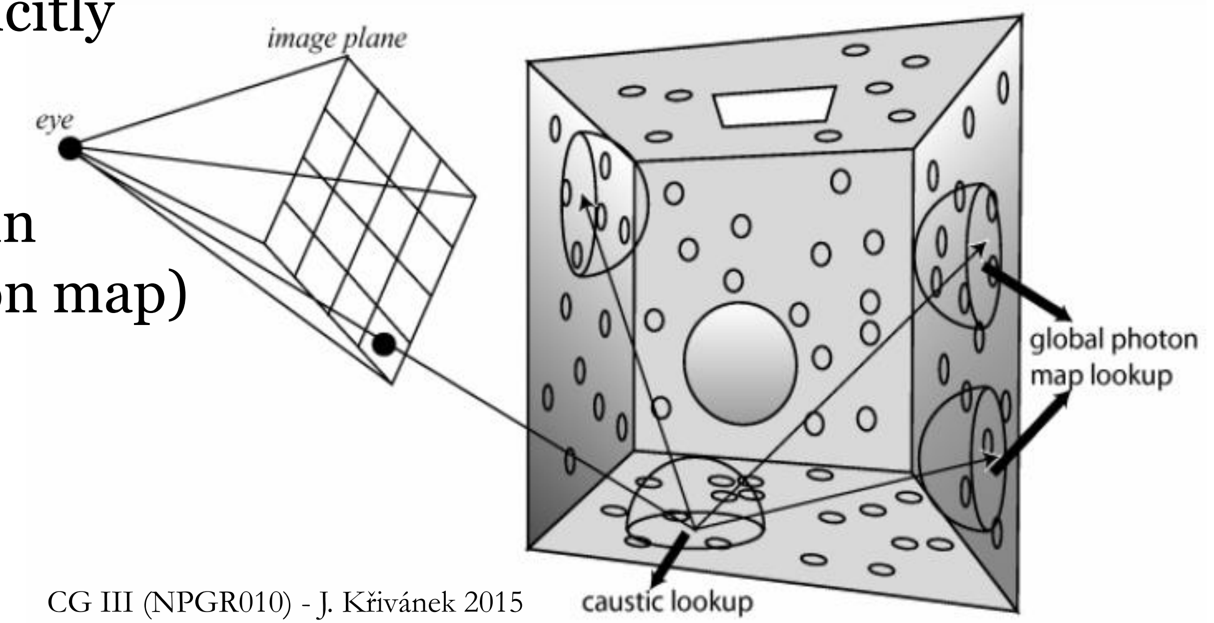
# Illumination calculation for a primary ray (or after a mirror reflection)

- Using the photon map
  - Caustics
    - Radiance estimate from the **caustic photon map**
  - Indirect illumination on diffuse or moderately glossy surfaces
    - Final gathering …



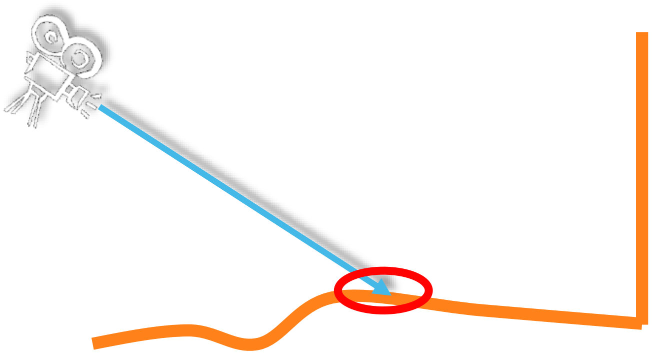
# Final gathering (FG)

- Indirect illumination on diffuse and moderately glossy surfaces
- One level of recursion as in distributed ray tracing (i.e. path tracing with massive splitting)
- For the intersection of secondary rays, use radiance estimate from the **global photon map**
  - No need to explicitly calculate direct illumination (it is contained in the global photon map)

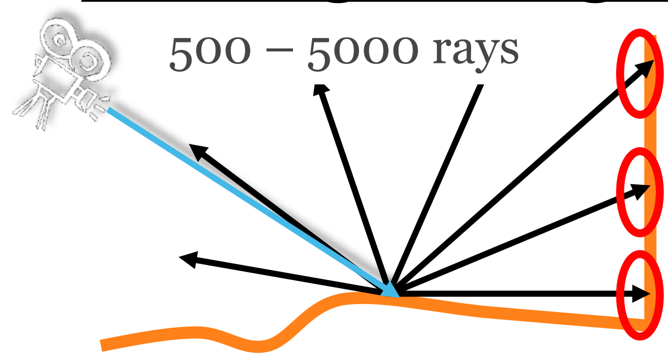


# Why do we need final gathering?





Information in the global photon map is too noisy for a direct use



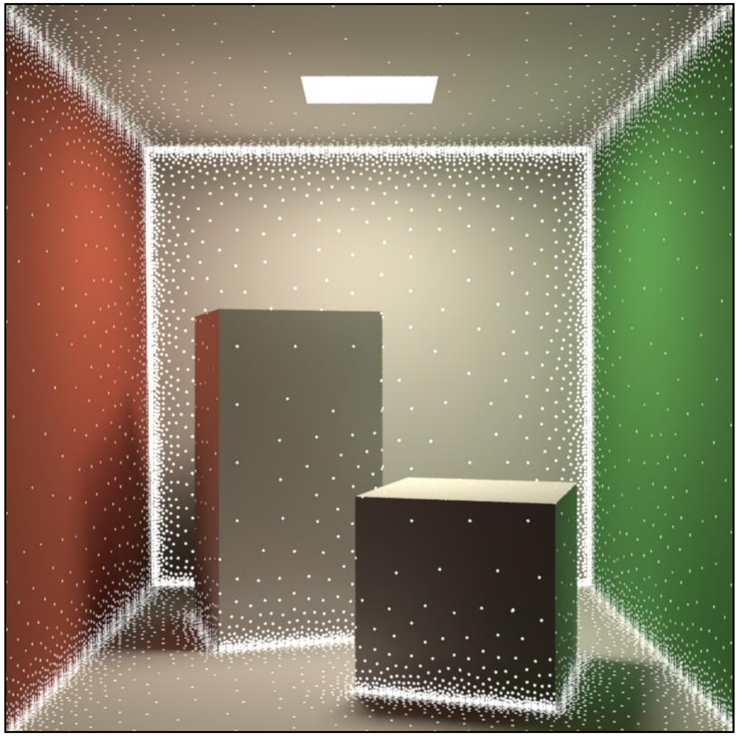

Inaccuracy in the global photon map is “averaged out”

# Why is there no final gathering for caustics?

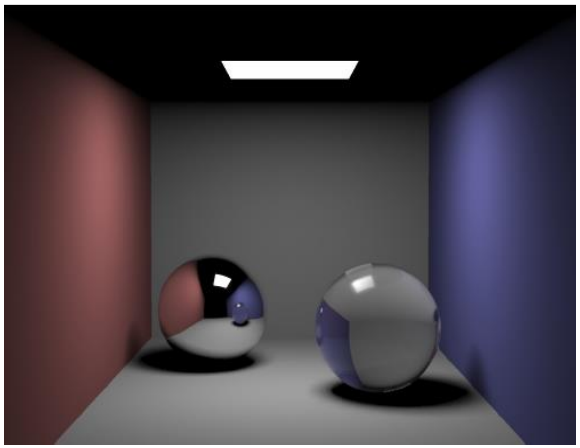
- Caustics = light focusing => sufficient photon density (beware, it's just a heuristic, may not always work well)

# Accelerating final gathering

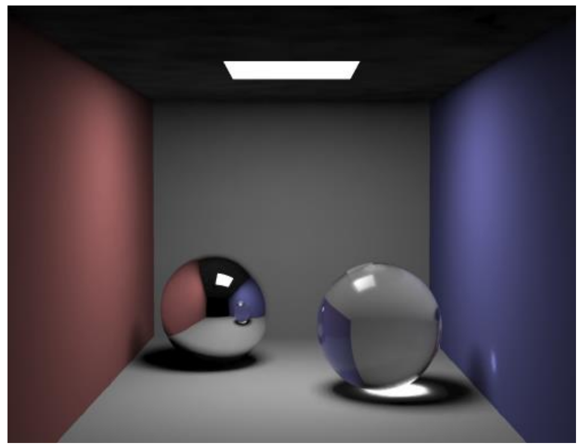
- **Irradiance caching** (next time)



# Results

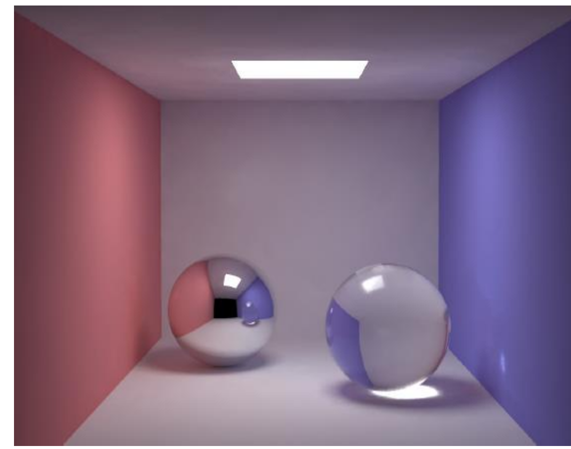


přímé osvětlení (21 s)



kaustiky (45 s)

50 000 photons
in the caustic map

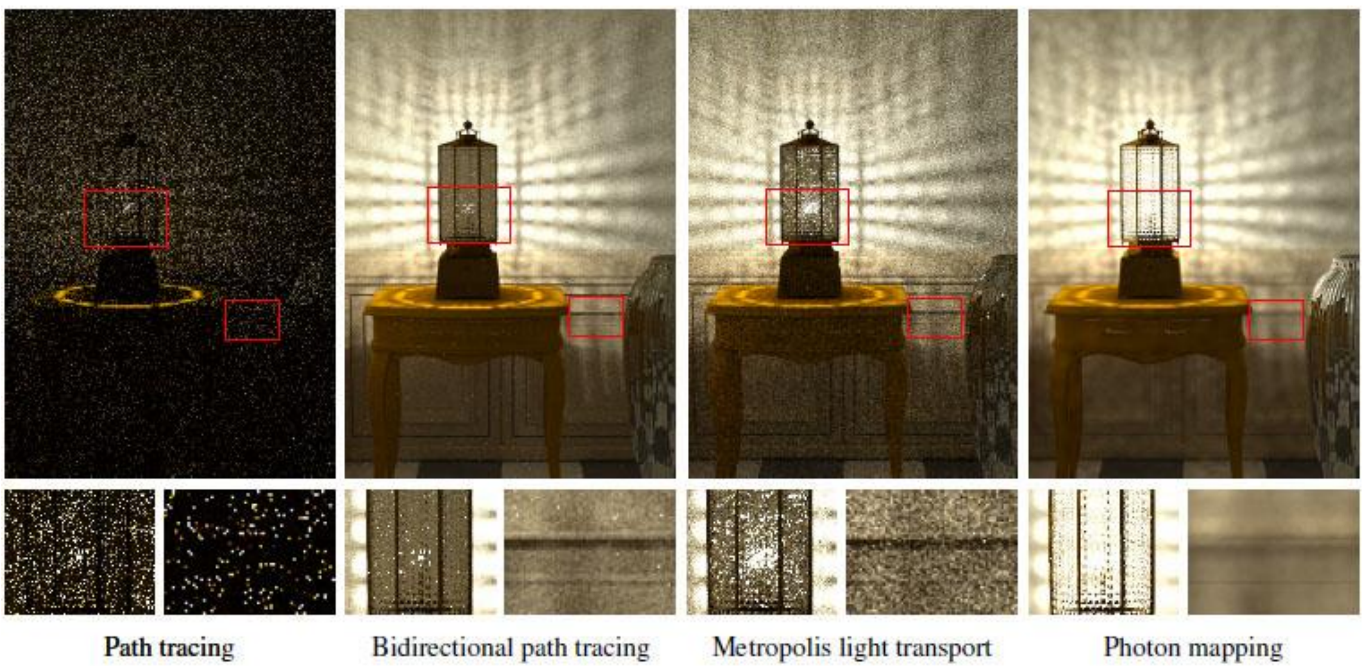


GI (66 s)

200 000 photons
in the global map

# What is photon mapping good at?

- Directly and indirectly visible caustics
- More generally: **SDS paths** (like light on the pool bottom)
  - Classic MC algorithms fail in such cases (path tracing, bidirectional path tracing, metropolis light transport)

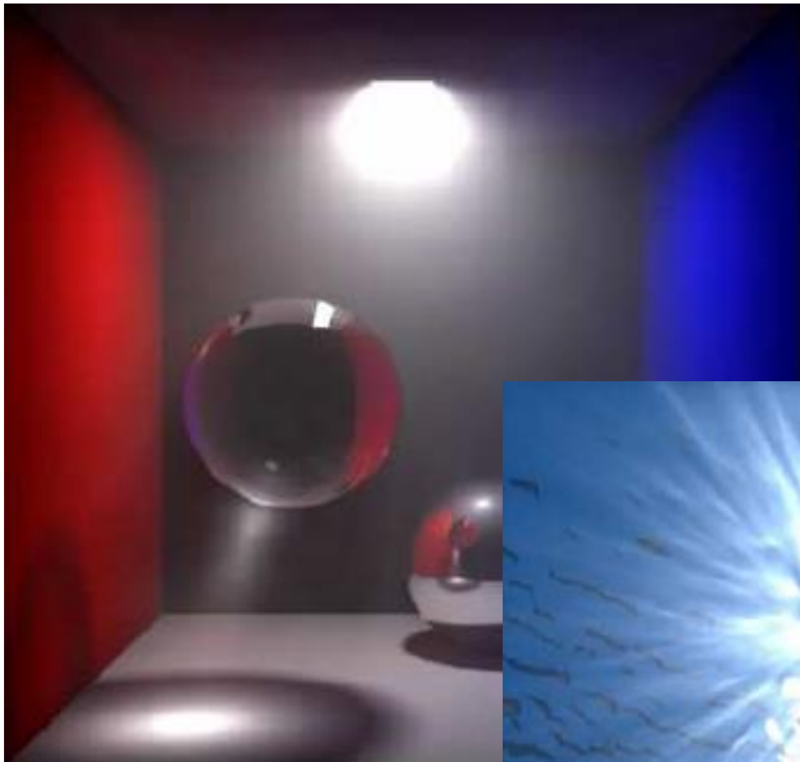


# SDS paths

© H.W.Jensen

© Wojciech Jarosz

# Photon mapping can be easily extended to handle scattering in media



Henrik Wann Jensen

# … and subsurface scattering

# Photon mapping problems

- Does not work well on glossy surfaces


# Photon mapping problems

- Does not work well on glossy surfaces
- So, what's wrong?
  - **Radiance estimate** from the photon map on a glossy surface **suffers from high variance**

# Theoretical problems of photon mapping

- Result is not unbiased
  - Contains systematic error

- Result is **consistent**
  - It theoretically converges as the photon count goes to infinity
  - But this is practically unachievable
  - Solution: **progressive photon mapping**

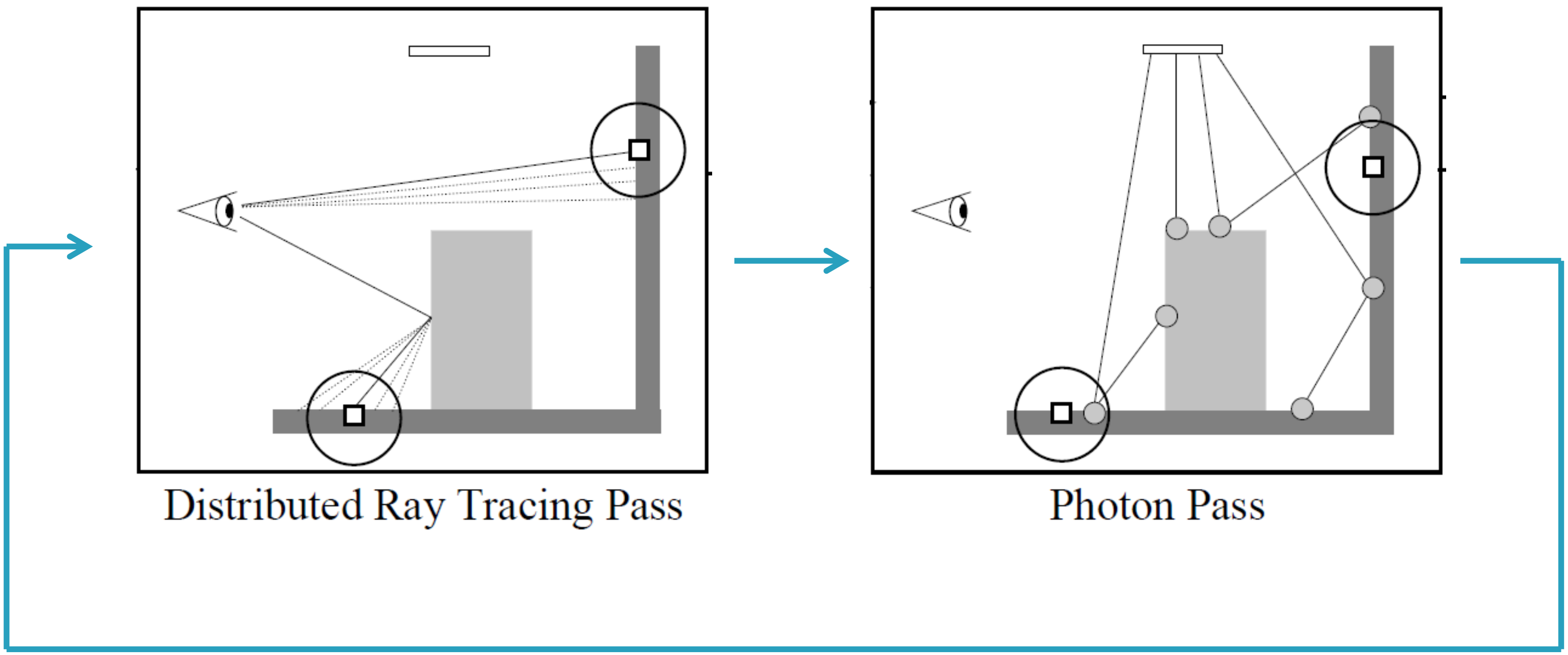
# Progressive photon mapping

# Progressive photon mapping

- Rendering in iterations
- In each iteration, **reduce the photon search radius** such that:
  - Total **bias goes to zero**, and
  - Total **variance goes to zero**
  - (i.e. the resulting estimator is consistent)

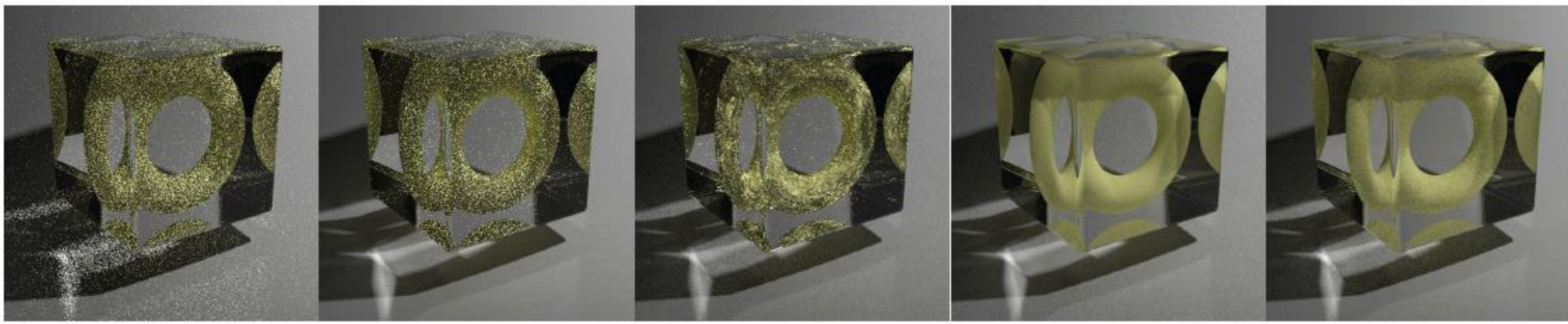
# Progressive photon mapping

- Iterative procedure




© Hachisuka et al.

# Progressive photon mapping



PT BDPT MLT PPM Reference

**Figure 7:** *Torus embedded in a glass cube. The reference image on the far right have been rendered using path tracing with 51500 samples per pixel. The Monte Carlo ray tracing methods fail to capture the lighting within the glass cube, while progressive photon mapping provides a smooth result using the same rendering time.*

# Progressive photon mapping

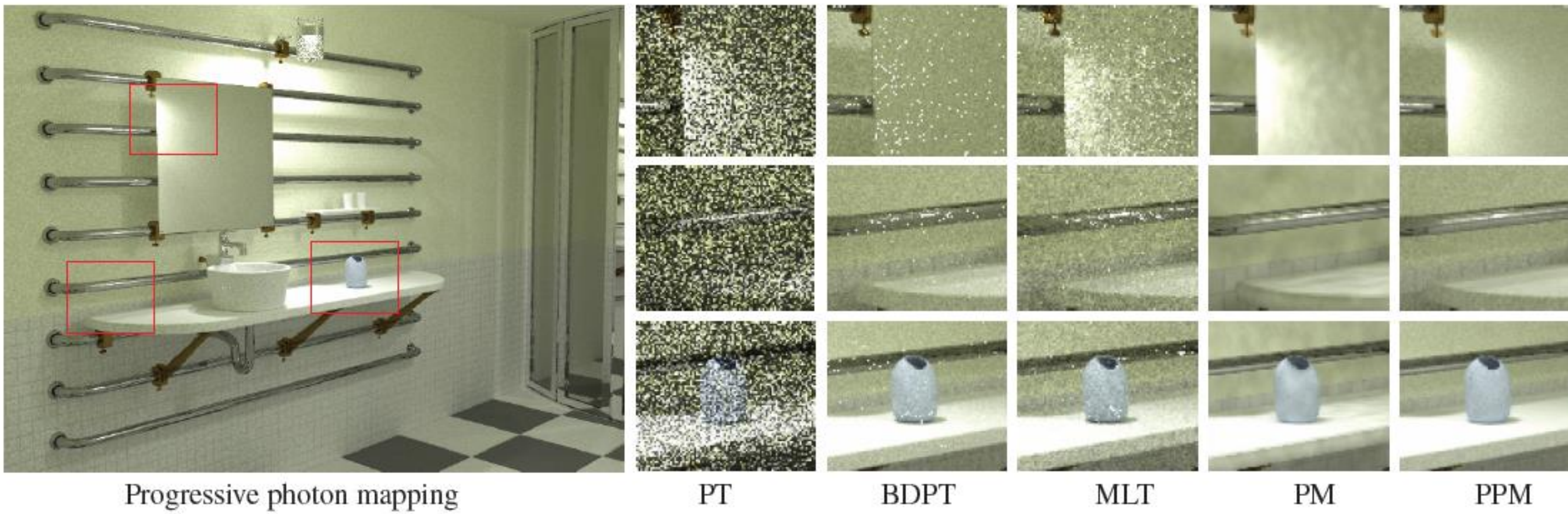
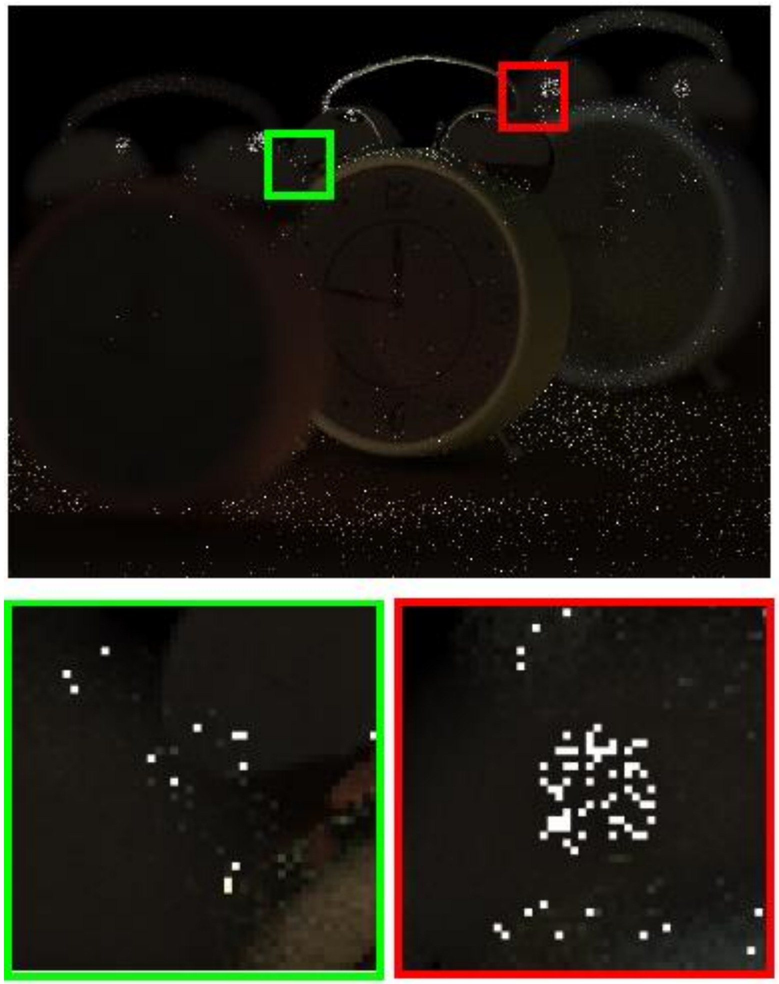
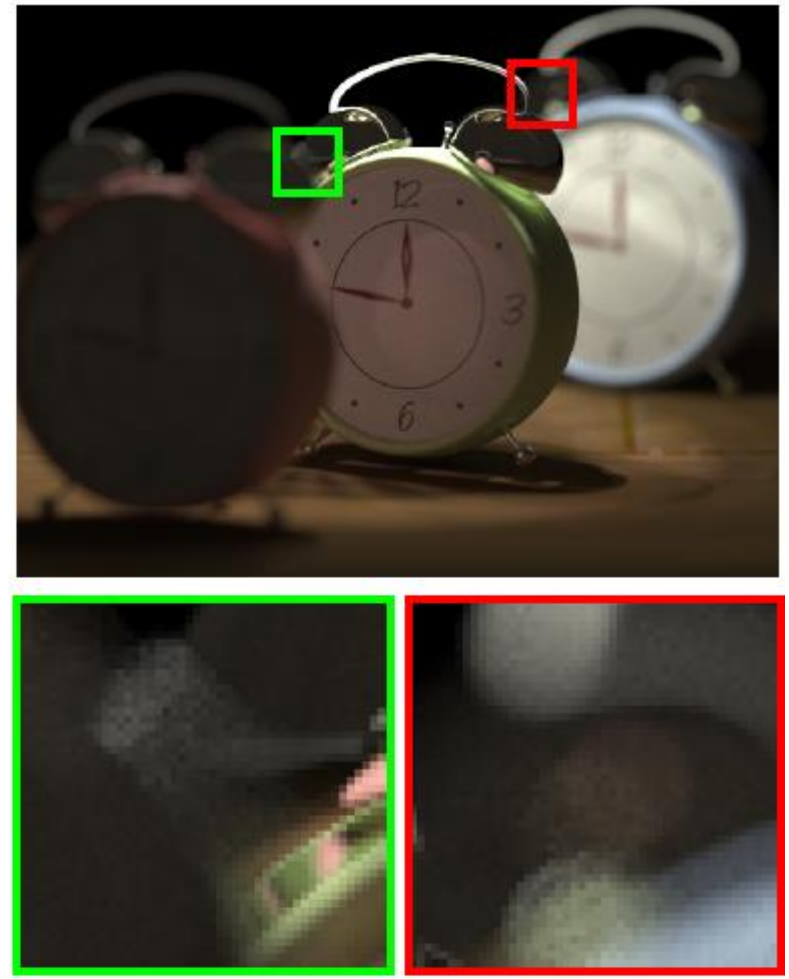



**Figure 8:** *Lighting simulation in a bathroom. The scene is illuminated by a small lighting fixture consisting of a light source embedded in glass. The illumination in the mirror cannot be resolved using Monte Carlo ray tracing. Photon mapping with 20 million photons results in a noisy and blurry image, while progressive photon mapping is able to resolve the details in the mirror and in the illumination without noise.*

# Progressive photon mapping



BDPT



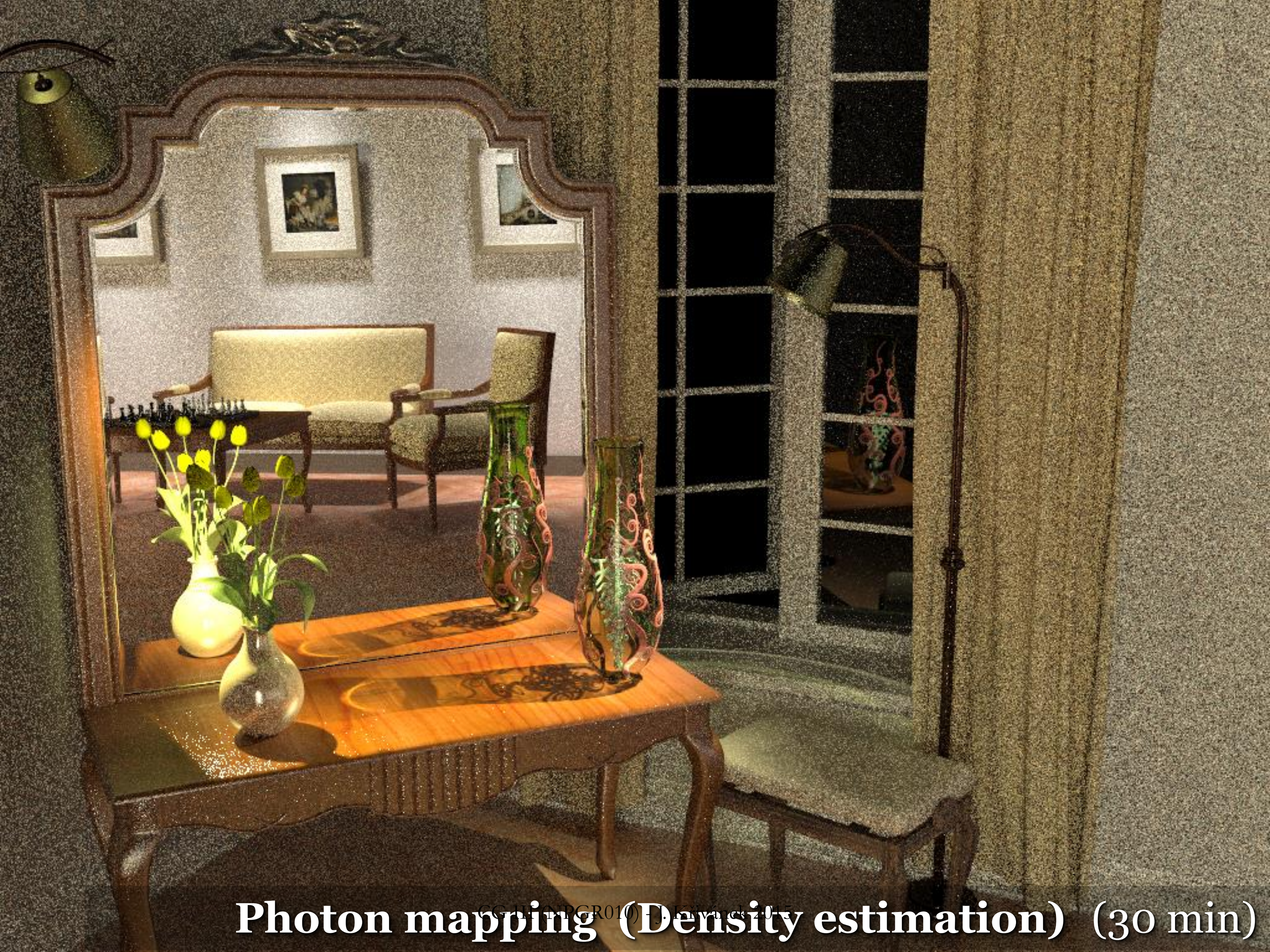

PPM

# Our work:
# Vertex Connection and Merging

# Robust photon mapping

- Where exactly on the camera sub-path should we look-up the photons?
- Commonly solved via a **heuristic**:
  - Diffuse surface … make the look-up right away
  - Specular surface … continue tracing and make the look-up later
- But what exactly should be classified as "diffuse" and "specular"?
  - We need a more **universal** and **robust** solution
  - Solution:
    - **Bidirectional photon mapping** [Vorba 2011]
    - **Vertex Connection and Merging** [Georgiev et al., 2012]

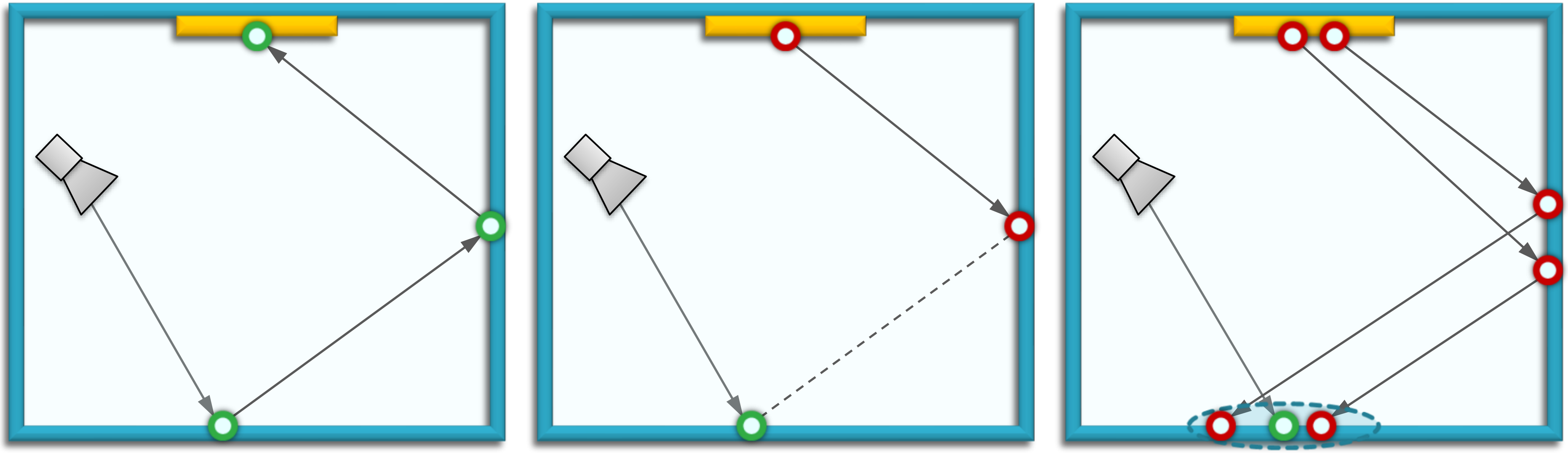
**Bidirectional path tracing** (30 min)



Photon mapping (Density estimation) (30 min)

# Vertex connection and merging (30 min)

# BPT vs PM



Unidirectional sampling | Vertex connection | Density estimation

**Bidirectional path tracing** | **Photon mapping**

# Overview

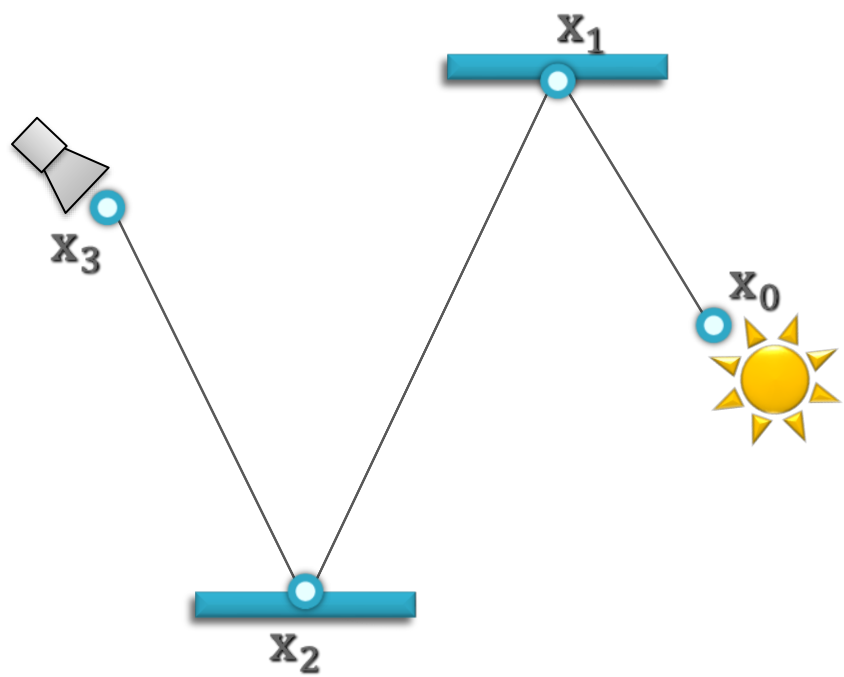
- ☹ **Problem**: different mathematical frameworks
  - **BPT**: Monte Carlo estimator of a path integral
  - **PM**: Density estimation

- ☝ **Key contribution:** Reformulate photon mapping in Veach's path integral framework

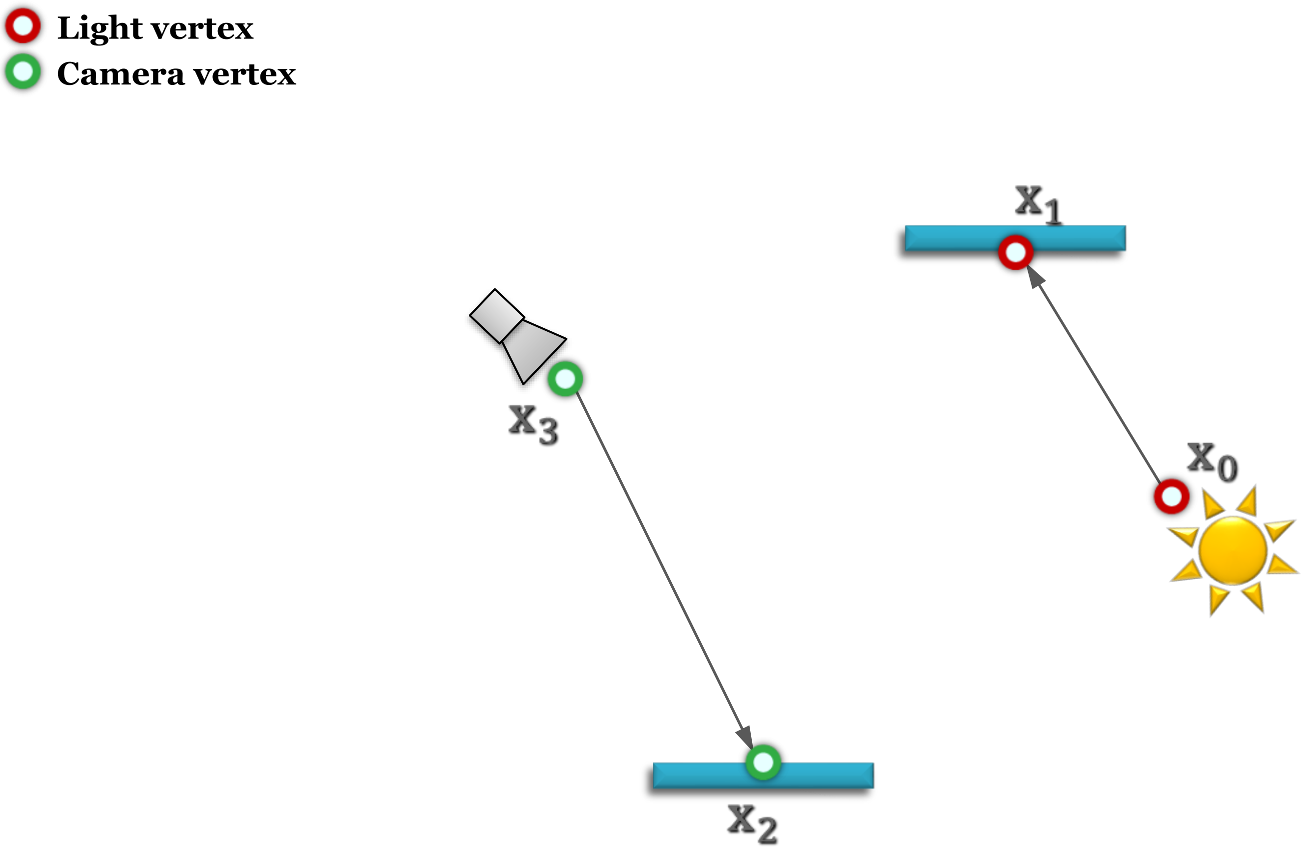
  1) Formalize as path sampling technique
  2) Derive path probability density

- ✓ Combination of BPT and PM into a **robust** algorithm

# Bidirectional MC path sampling

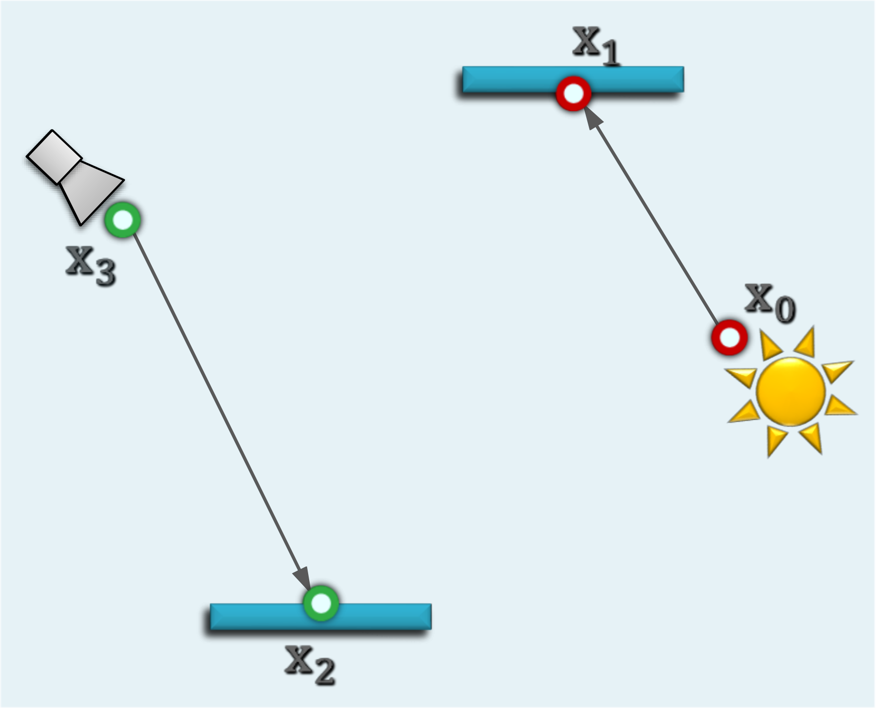


# Bidirectional MC path sampling



# Bidirectional MC path sampling

- Light vertex
- Camera vertex

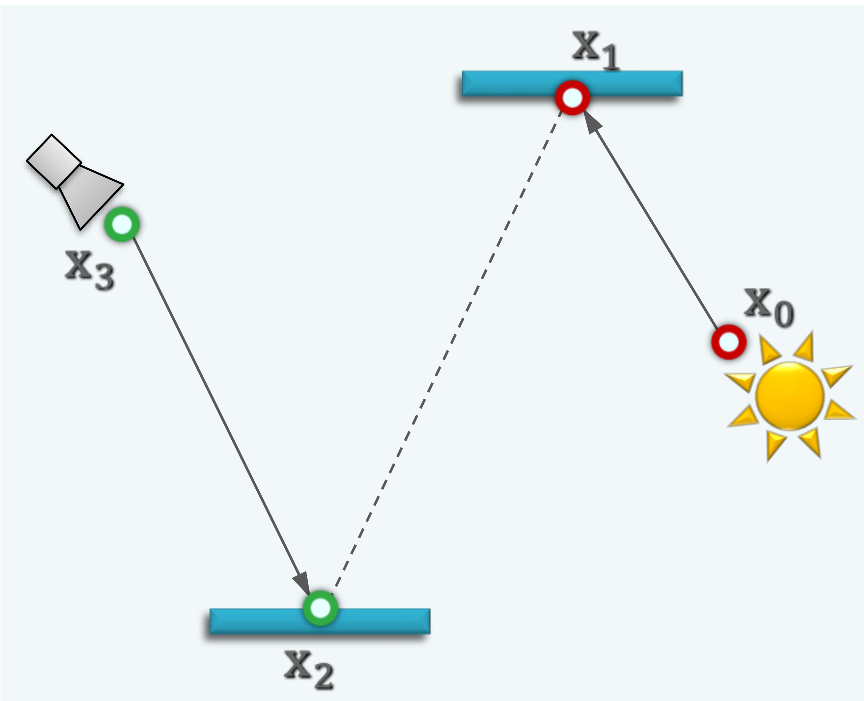


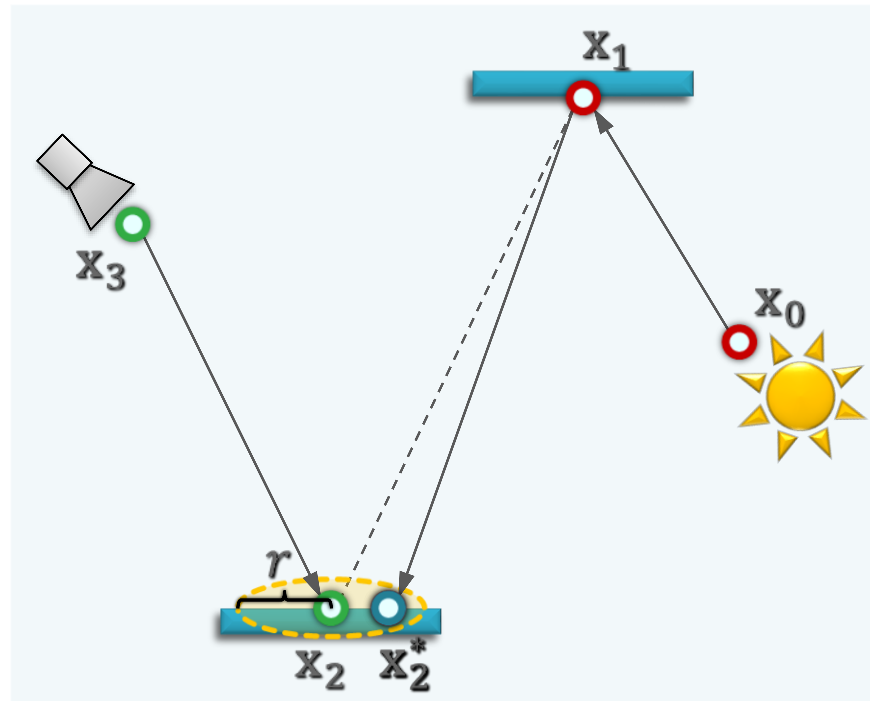

Bidirectional path tracing

Photon mapping

# Bidirectional MC path sampling

- **Light vertex**
- **Camera vertex**




**Bidirectional path tracing** / **Vertex connection**

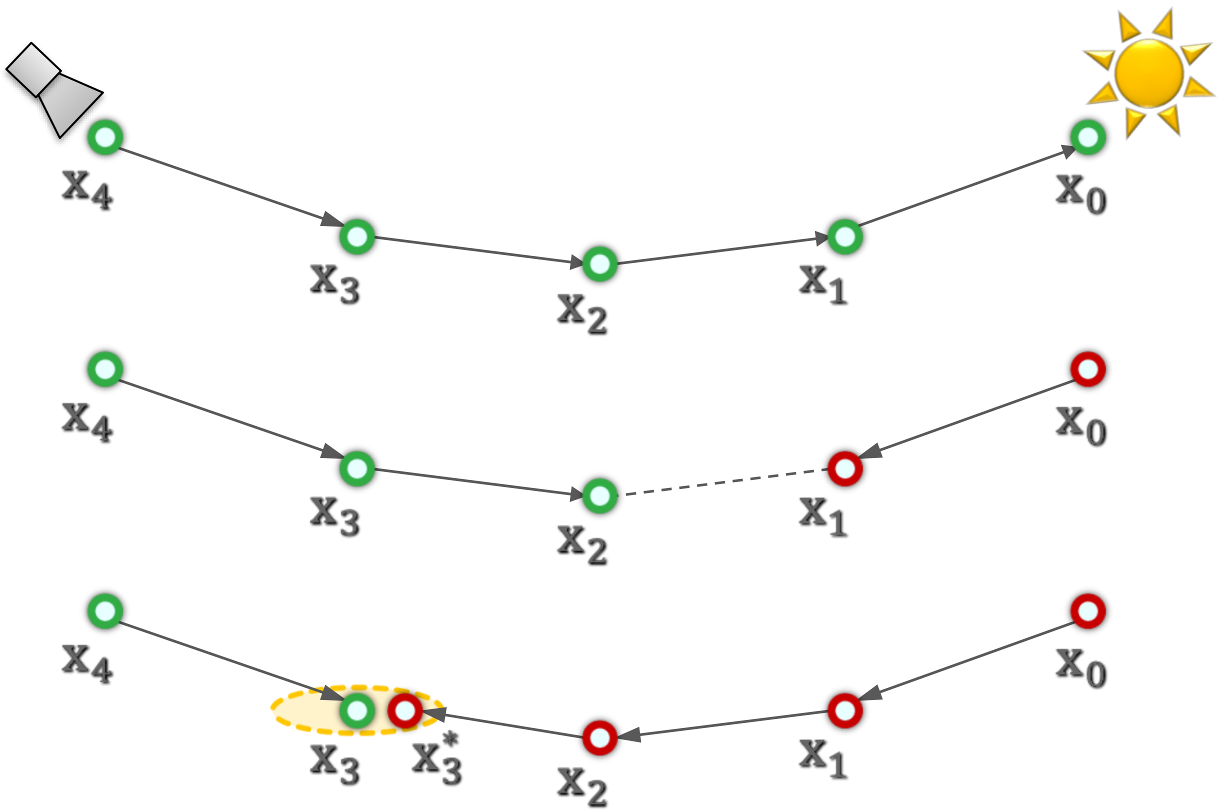
$$p_{VC}(\bar{\mathrm{x}}) = p(\mathrm{x}_0)p(\mathrm{x}_0 \to \mathrm{x}_1)\, p(\mathrm{x}_3)p(\mathrm{x}_3 \to \mathrm{x}_2)$$

**Photon mapping** / **Vertex merging**

$$p_{VM}(\bar{\mathrm{x}}) \approx p(\mathrm{x}_0)p(\mathrm{x}_0 \to \mathrm{x}_1)\, P(\|\mathrm{x}_2 - \mathrm{x}_2^*\| < r)\, p(\mathrm{x}_3)p(\mathrm{x}_3 \to \mathrm{x}_2)$$

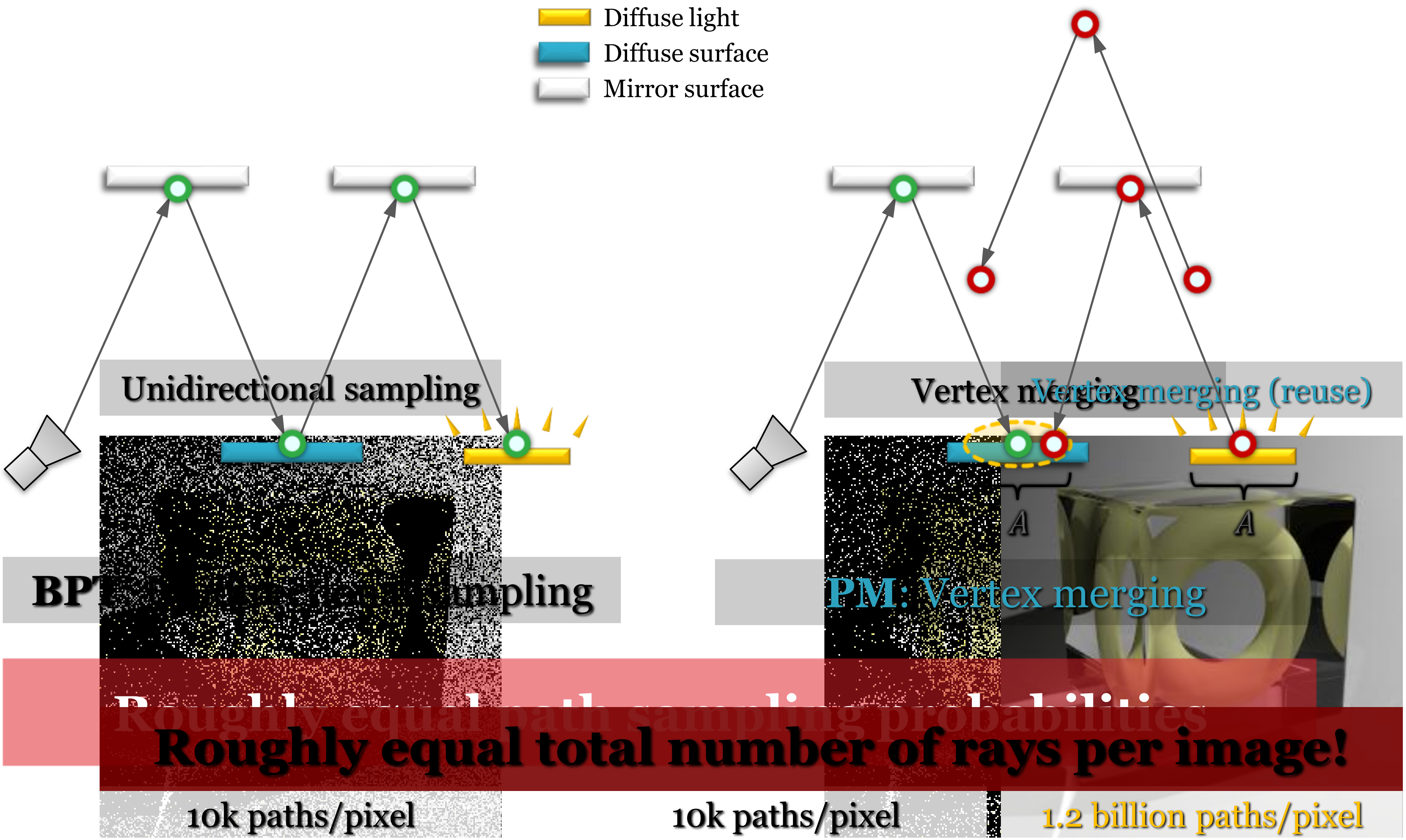
# Sampling techniques

**Light vertex**
**Camera vertex**



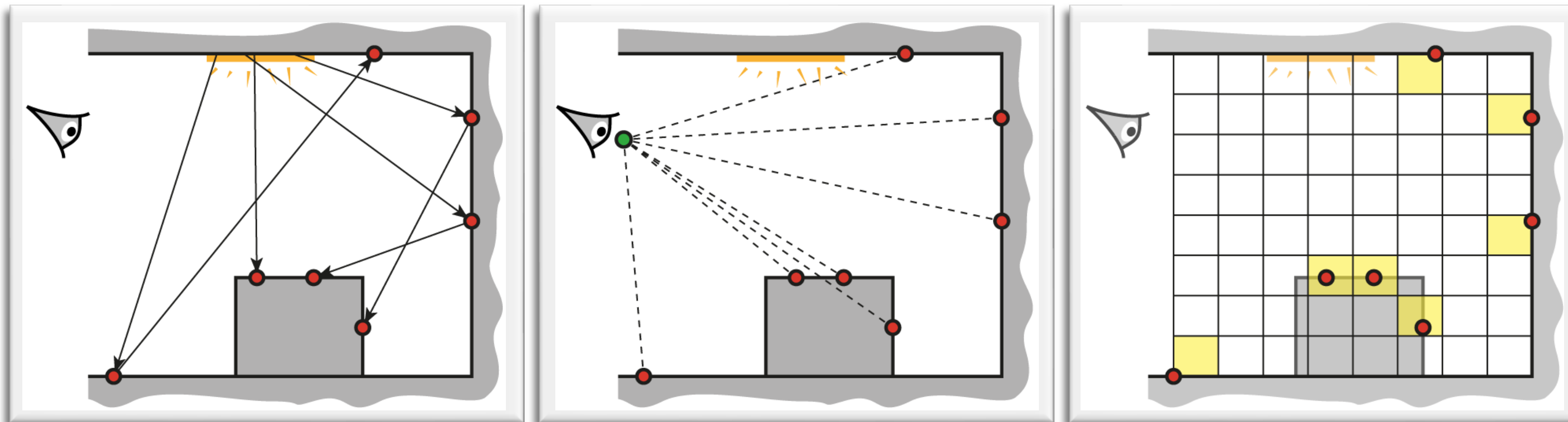

| | |
|---|---|
| Unidirectional | 2 ways |
| Vertex connection | 4 ways |
| **Vertex merging** | **5 ways** |
| **Total** | **11 ways** |

# Technique comparison – SDS Paths



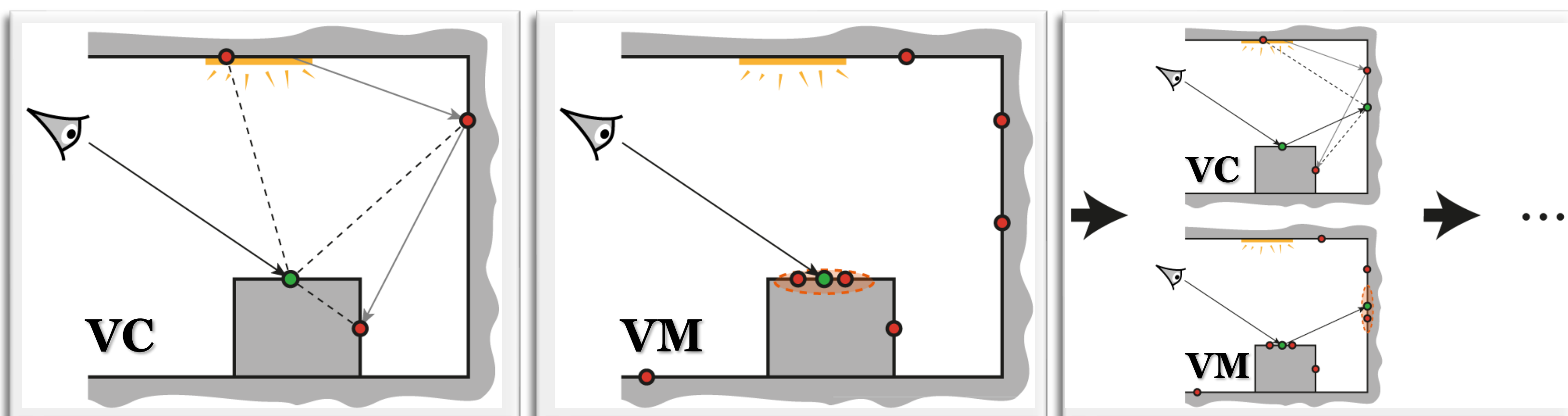
# VCM – Algorithm overview

## Stage 1: Light sub-path sampling



a) Trace sub-paths | b) Connect to eye | c) Build search struct.

## Stage 2: Eye sub-path sampling




a) Vertex connection | b) Vertex merging | c) Continue sub-path

**Bidirectional path tracing** (30 min)

**Stochastic progressive photon mapping** (30 min)

# Vertex connection and merging (30 min)

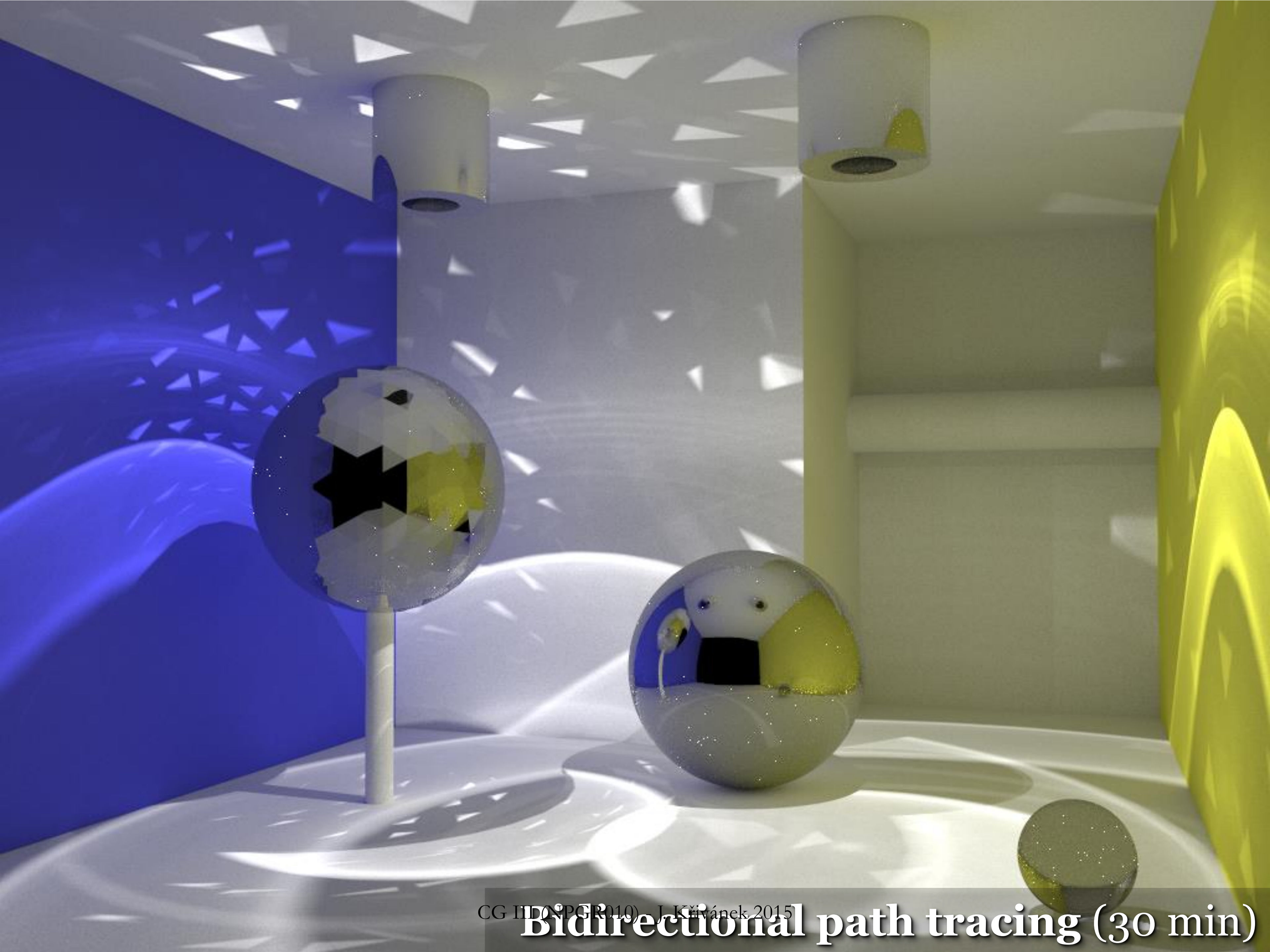

# Relative technique contributions


# Bidirectional path tracing (30 min)

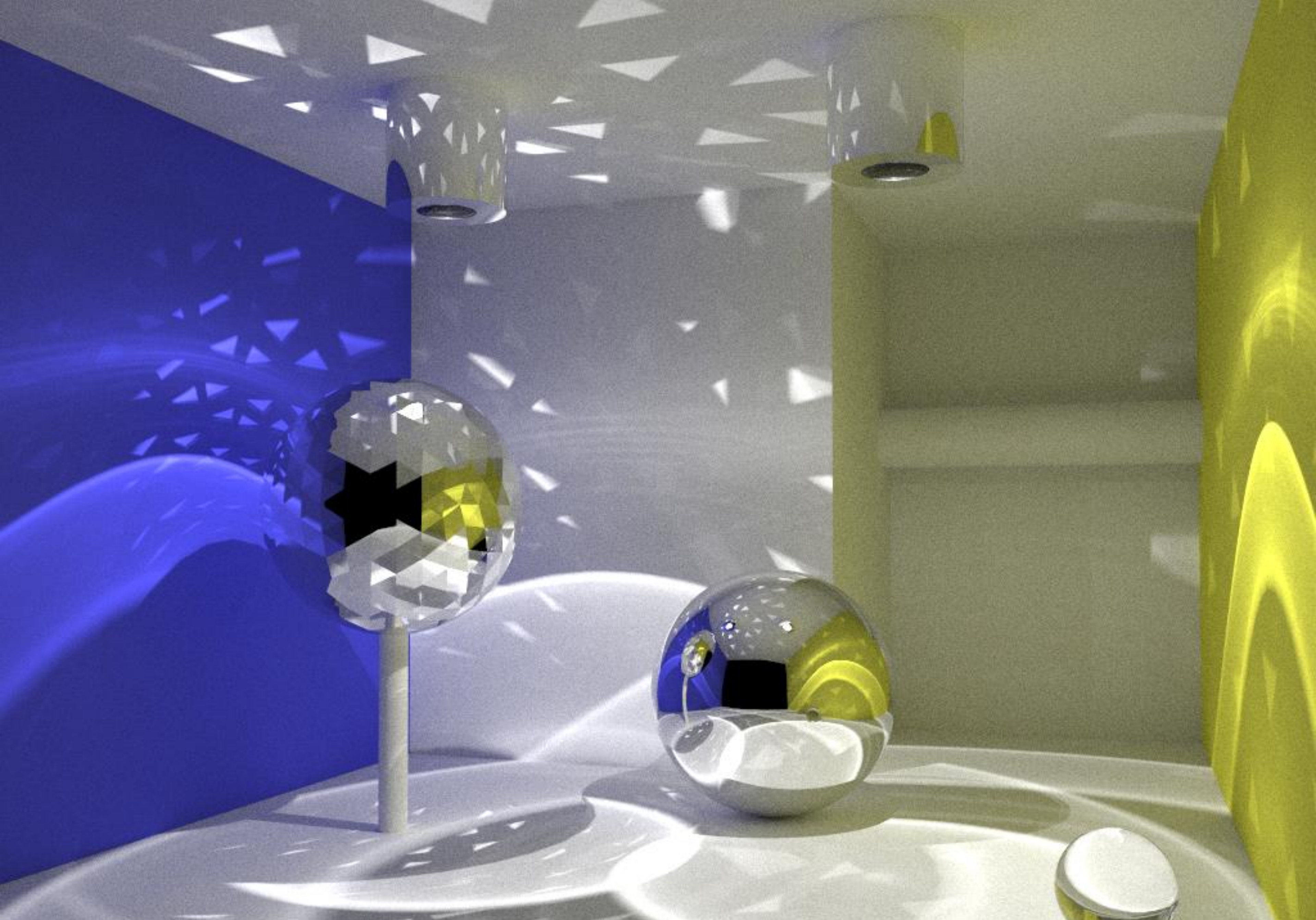

Stochastic progressive photon mapping (30 min)


# Vertex connection and merging (30 min)

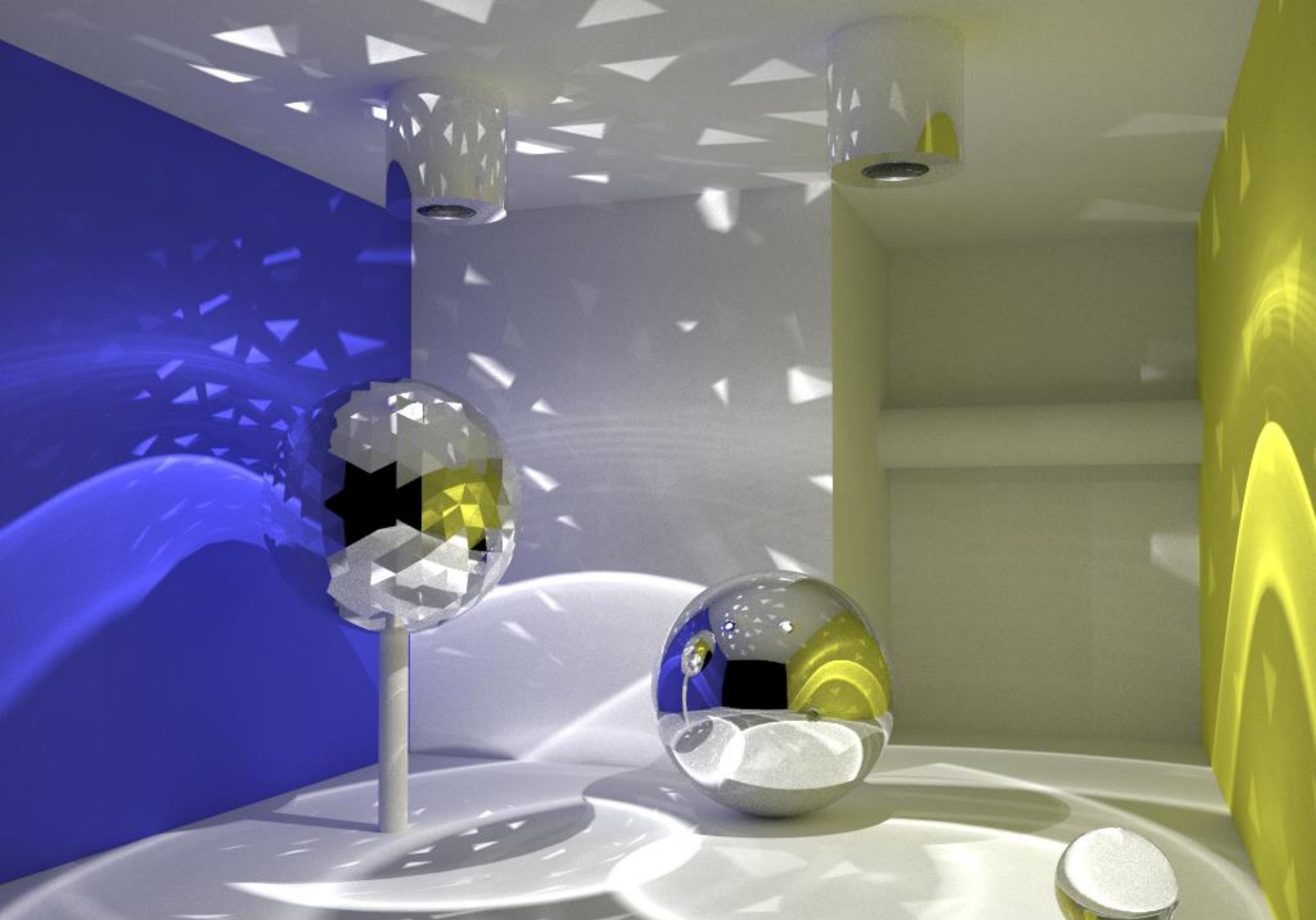

# Relative technique contributions



# Bidirectional path tracing (30 min)

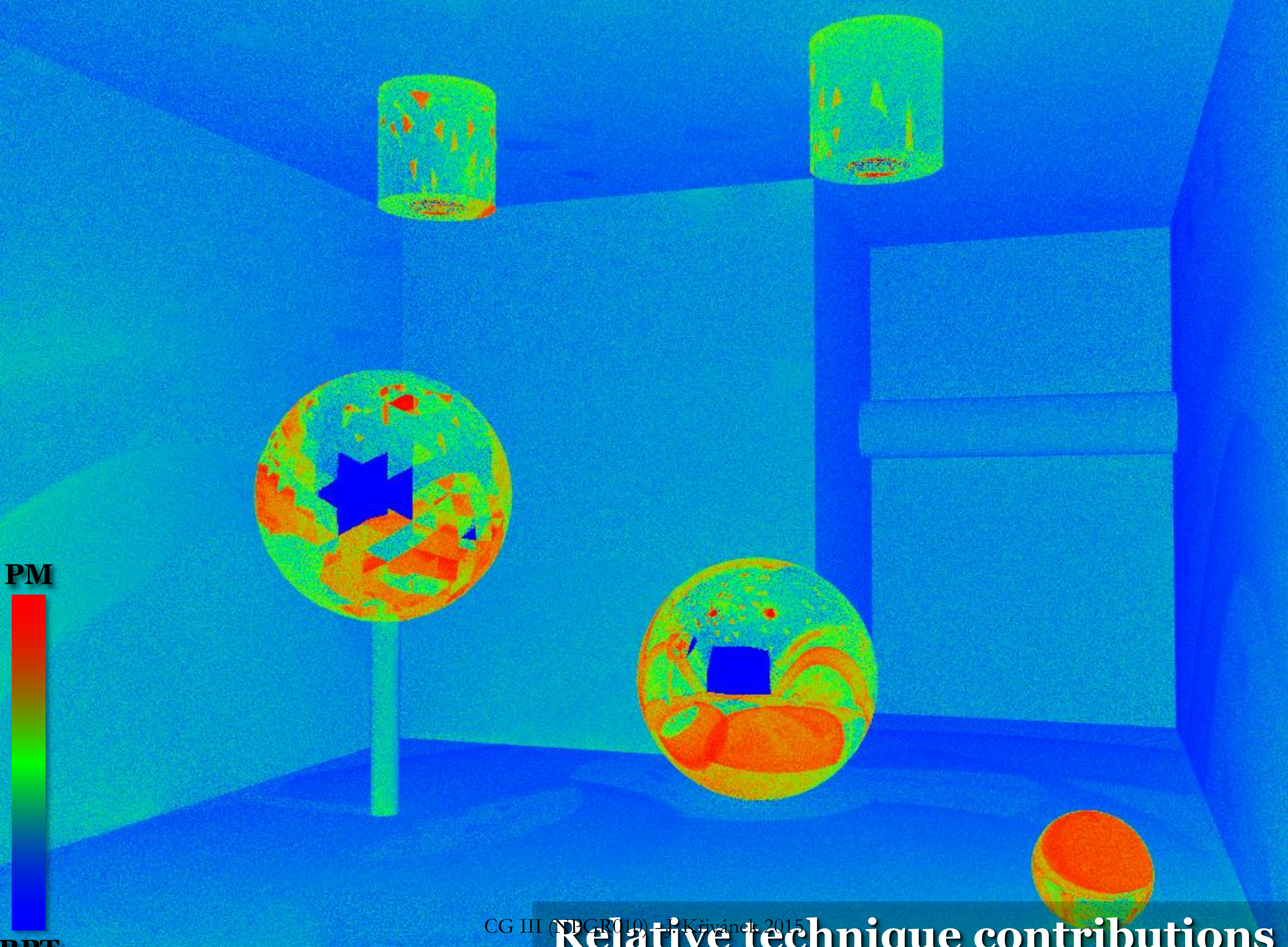


**Stochastic progressive photon mapping (30 min)**

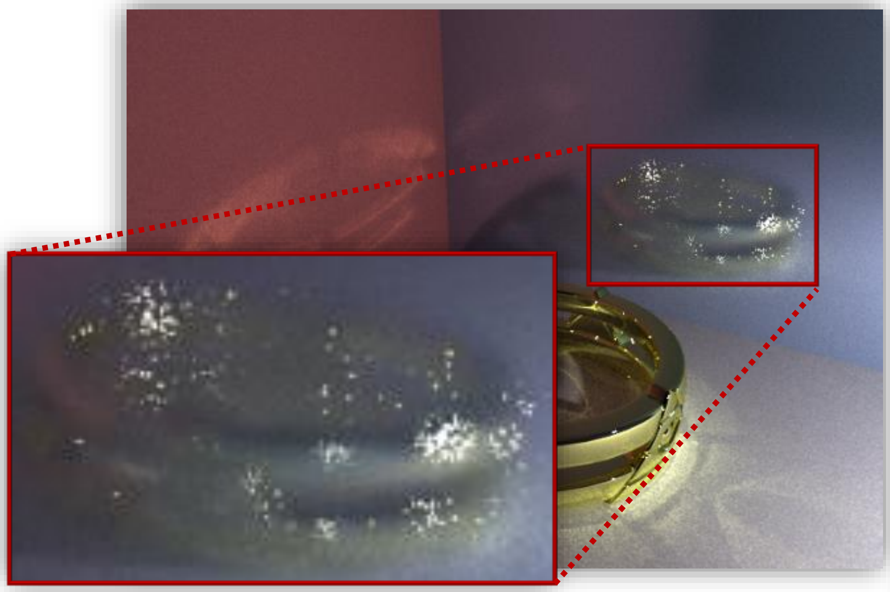


# Vertex connection and merging (30 min)

# Relative technique contributions



# Remaining challenges



# References

- *Georgiev et al.*, "Light Transport Simulation with Vertex Connection and Merging"
- *Hachisuka et al.* "A Path Space Extension for Robust Light Transport Simulation"
  - Same algorithm, different theoretical derivations

# VCM in production

# References


- Jensen H.W.: **Realistic Image Synthesis using Photon Mapping**. A.K. Peters, 2001
- Hachisuka & Jensen. **Stochastic Progressive Photon Mapping**, ACM Trans. Graph. (SIGGRAPH Asia 2009). link
- Knaus & Zwicker. **Progressive photon mapping: A probabilistic approach**. ACM Trans. Graph., 2011. link
- Georgiev, Křivánek, Davidovič, Slusallek. **Light Transport Simulation with Vertex Connection and Merging.** ACM Trans. Graph. (SIGGRAPH Asia 2012). link
- Hachisuka, Pantaleoni, Jensen. **A Path Space Extension for Robust Light Transport Simulation.** ACM Trans. Graph. (SIGGRAPH Asia 2012). link