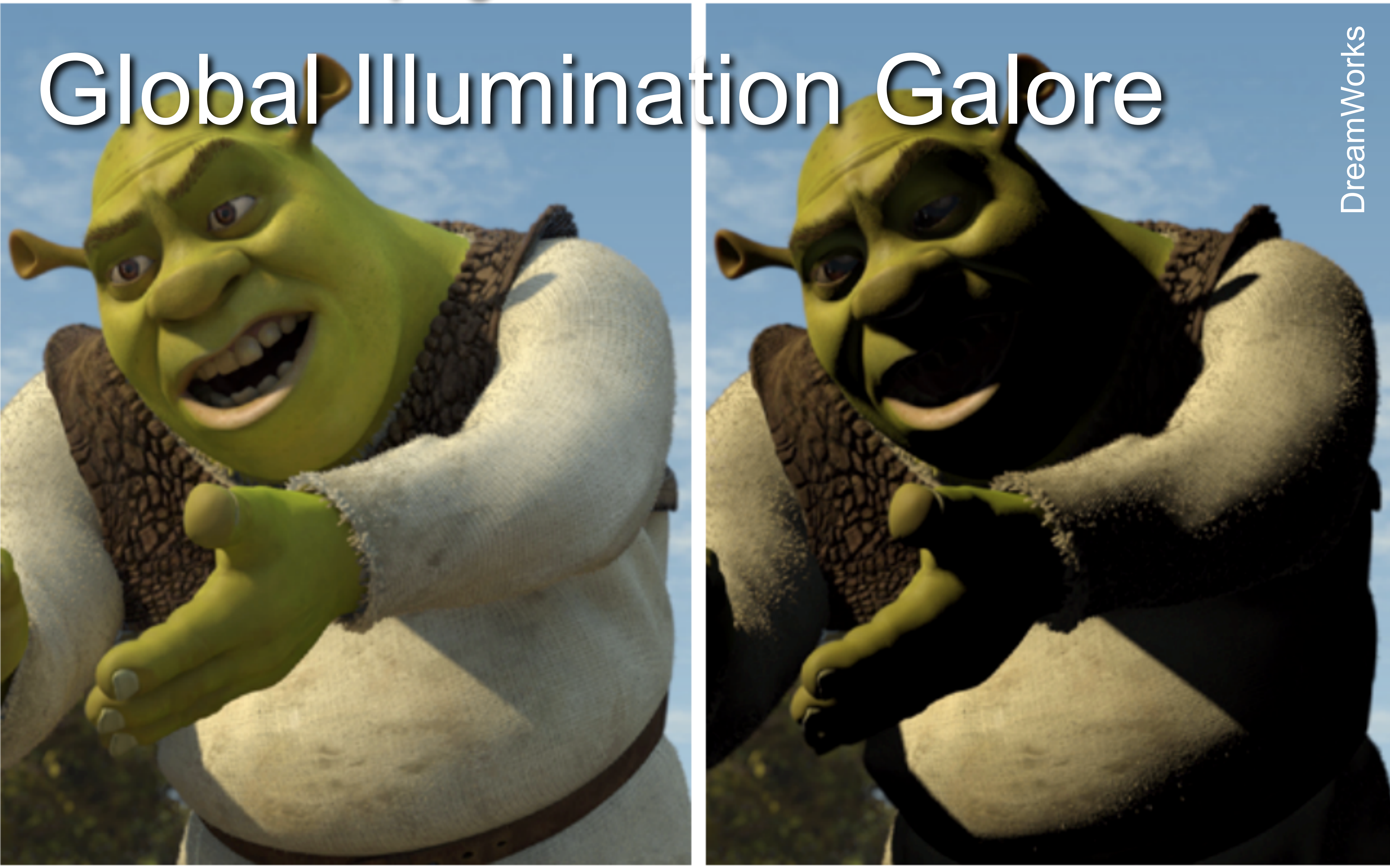
# Global Illumination Galore

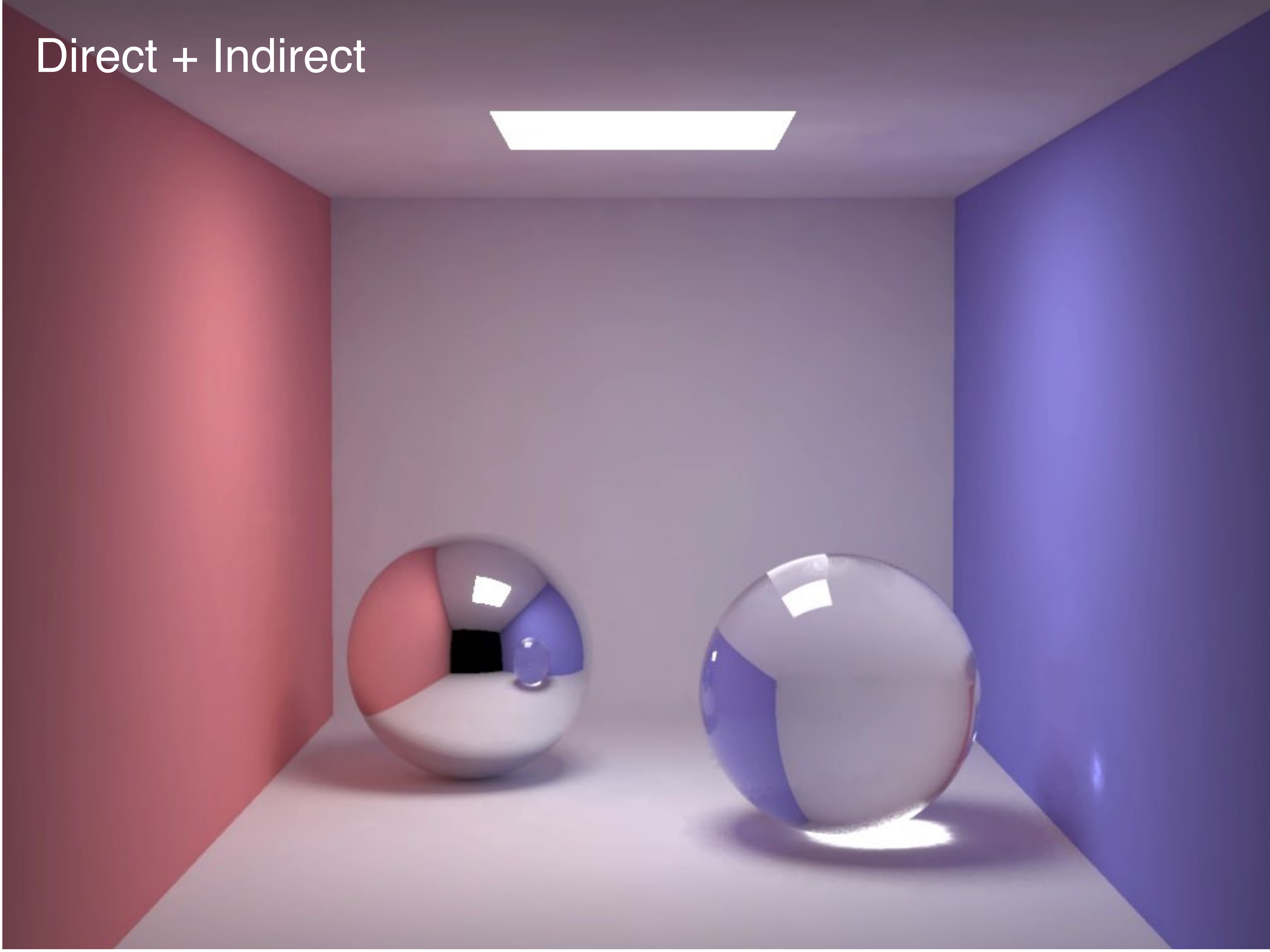


(a) direct and indirect lighting (b) direct lighting only

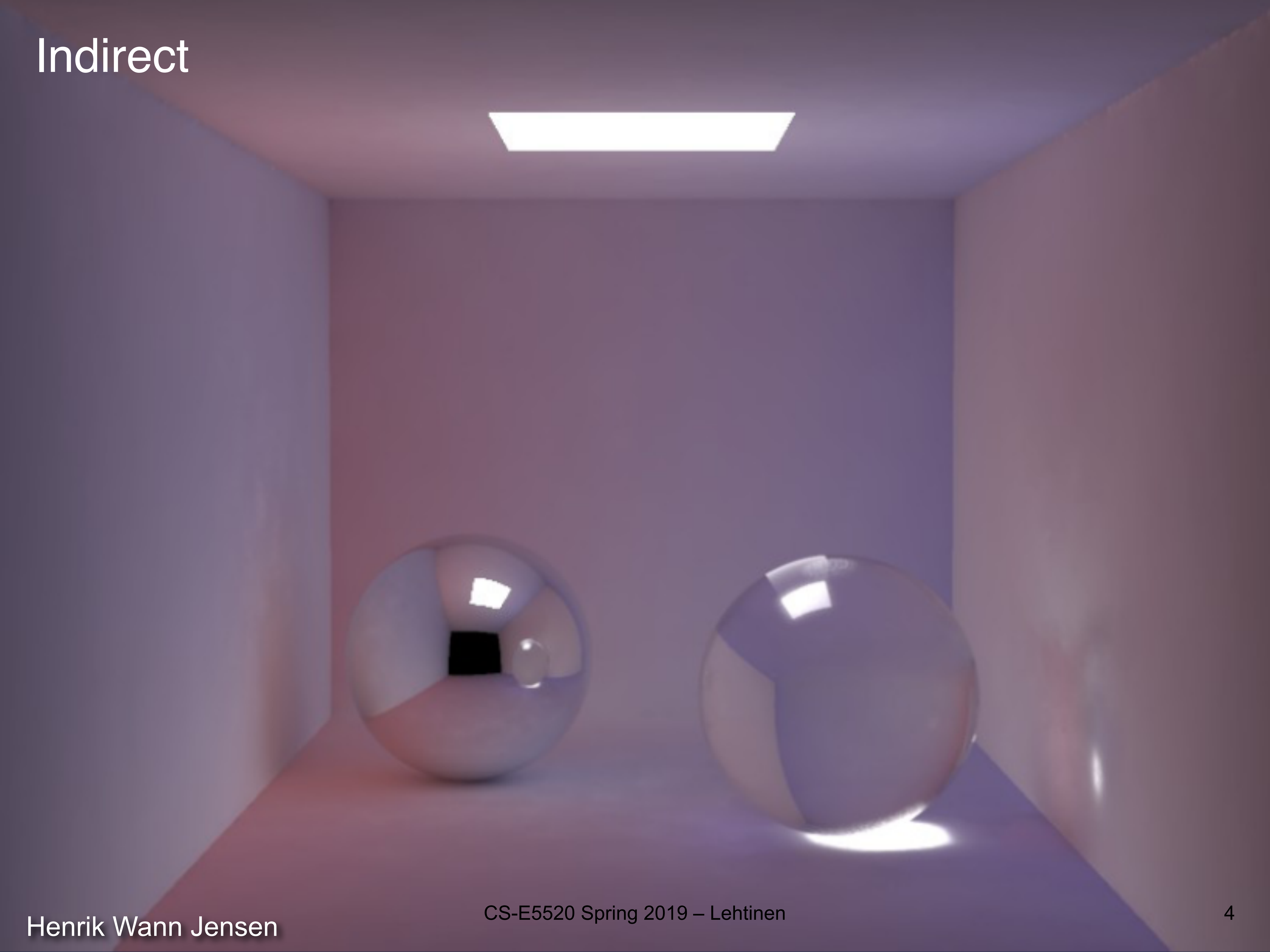
## Today

- Global Illumination Galore!
  - Irradiance Caching and filtering
    - Indirect light varies slowly
    - So, let's interpolate sparse samples, or remove noise by filtering
  - Photon Mapping
    - Global Illumination through density estimation

- The stuff we didn't talk about while focusing on "pure" Monte Carlo techniques

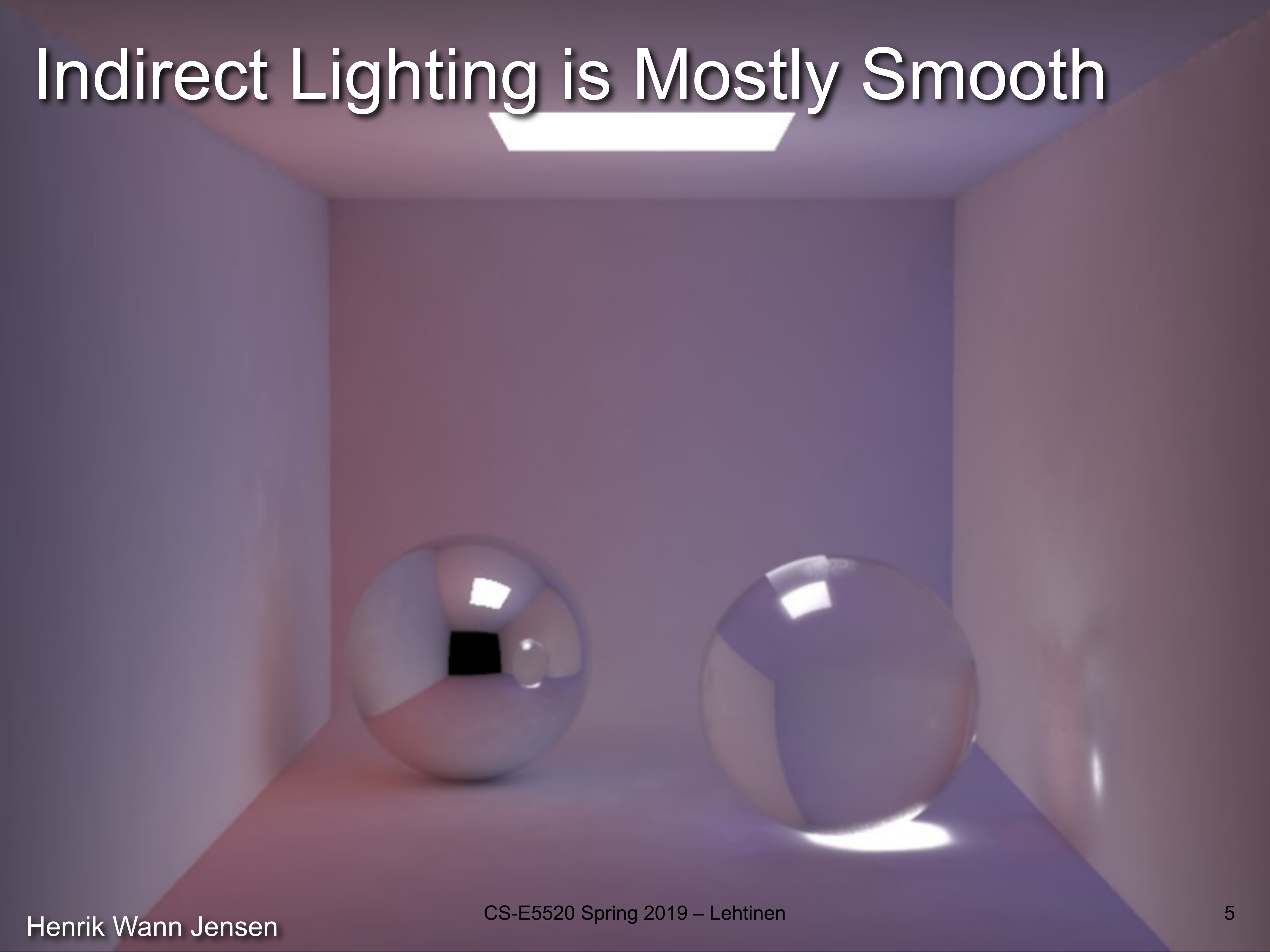

Direct + Indirect

## Indirect



Henrik Wann Jensen

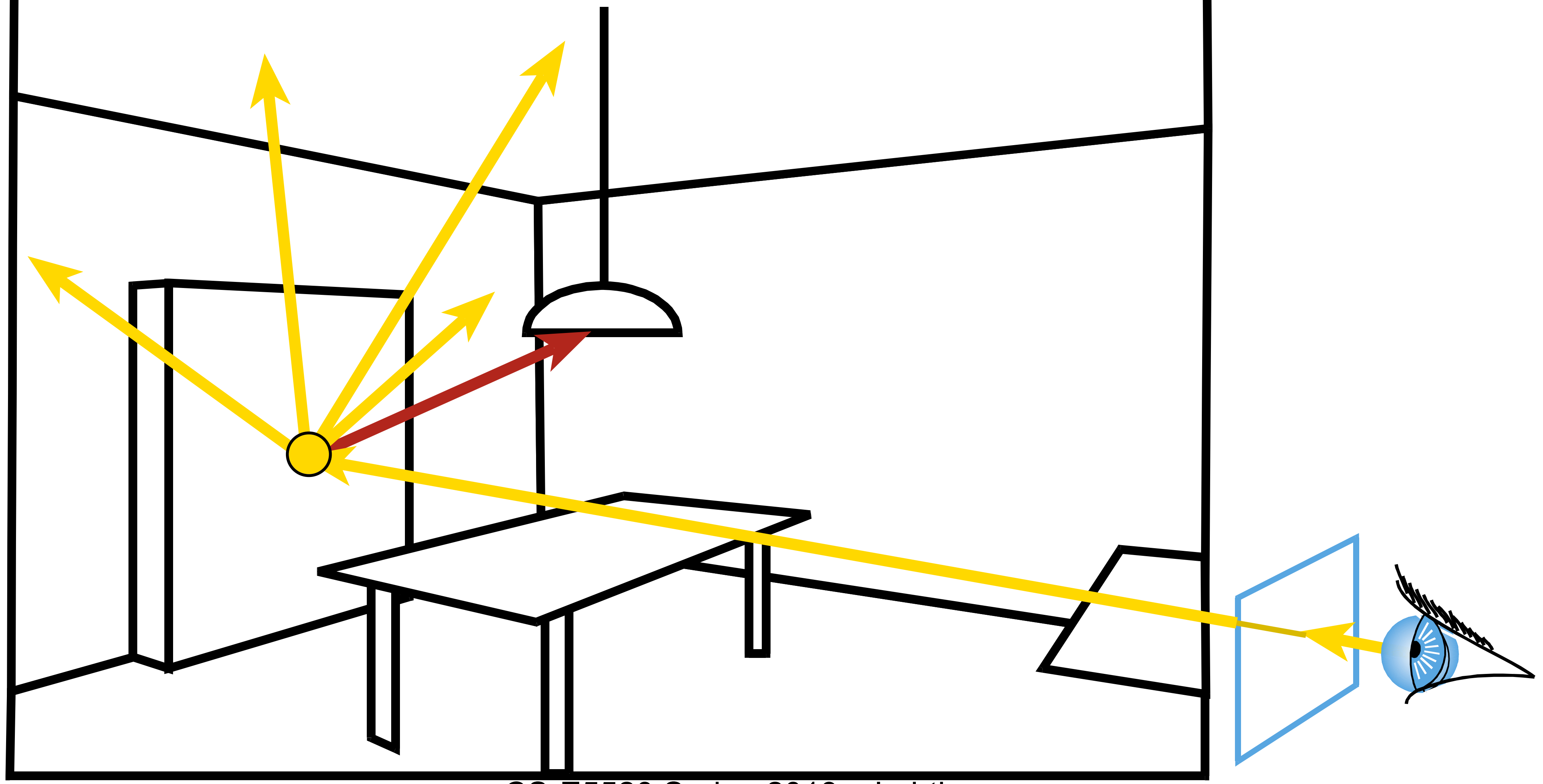
## Indirect Lighting is Mostly Smooth



Henrik Wann Jensen

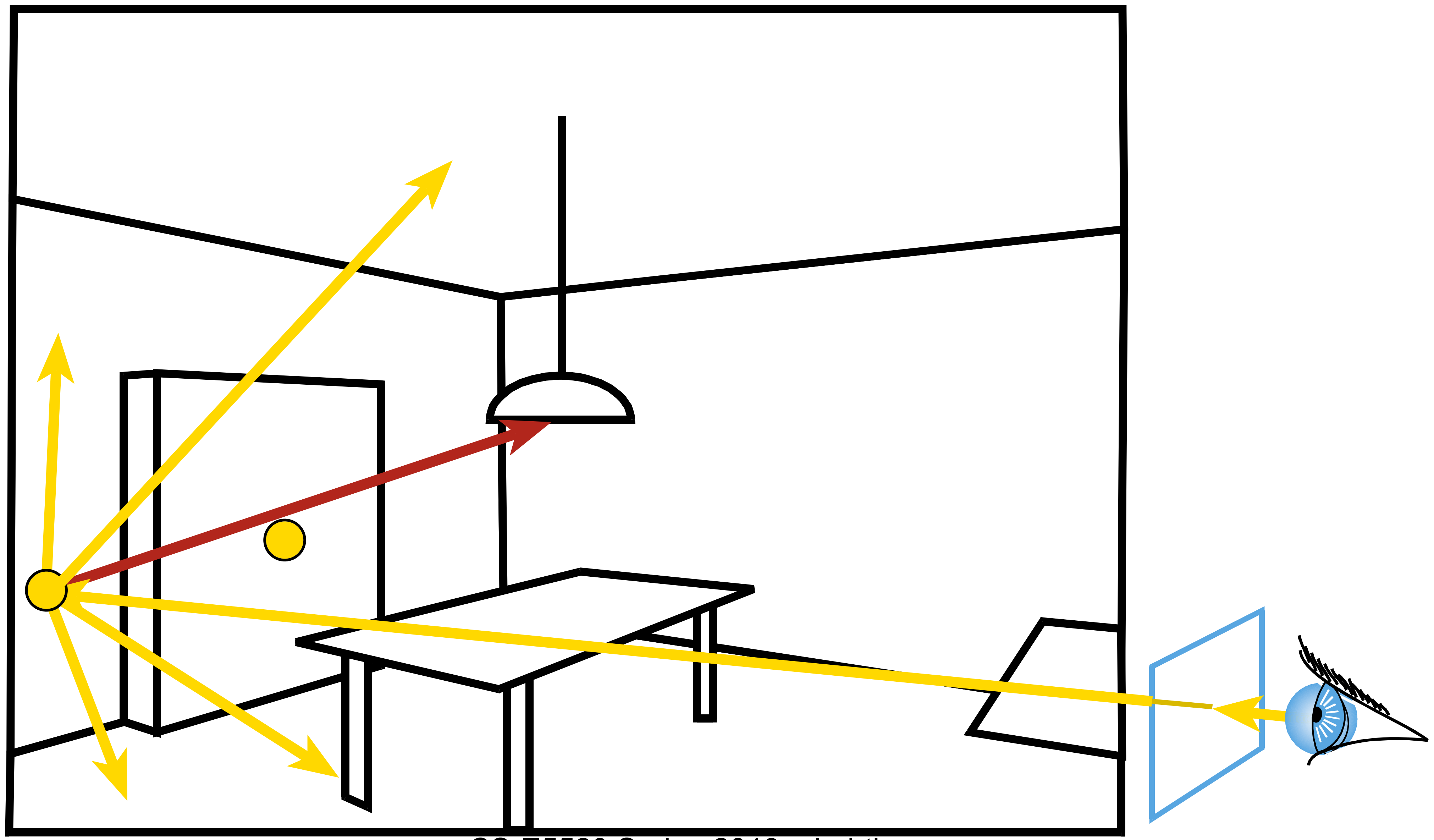
## Irradiance Caching [Ward 1988]

- Indirect illumination is most often smooth



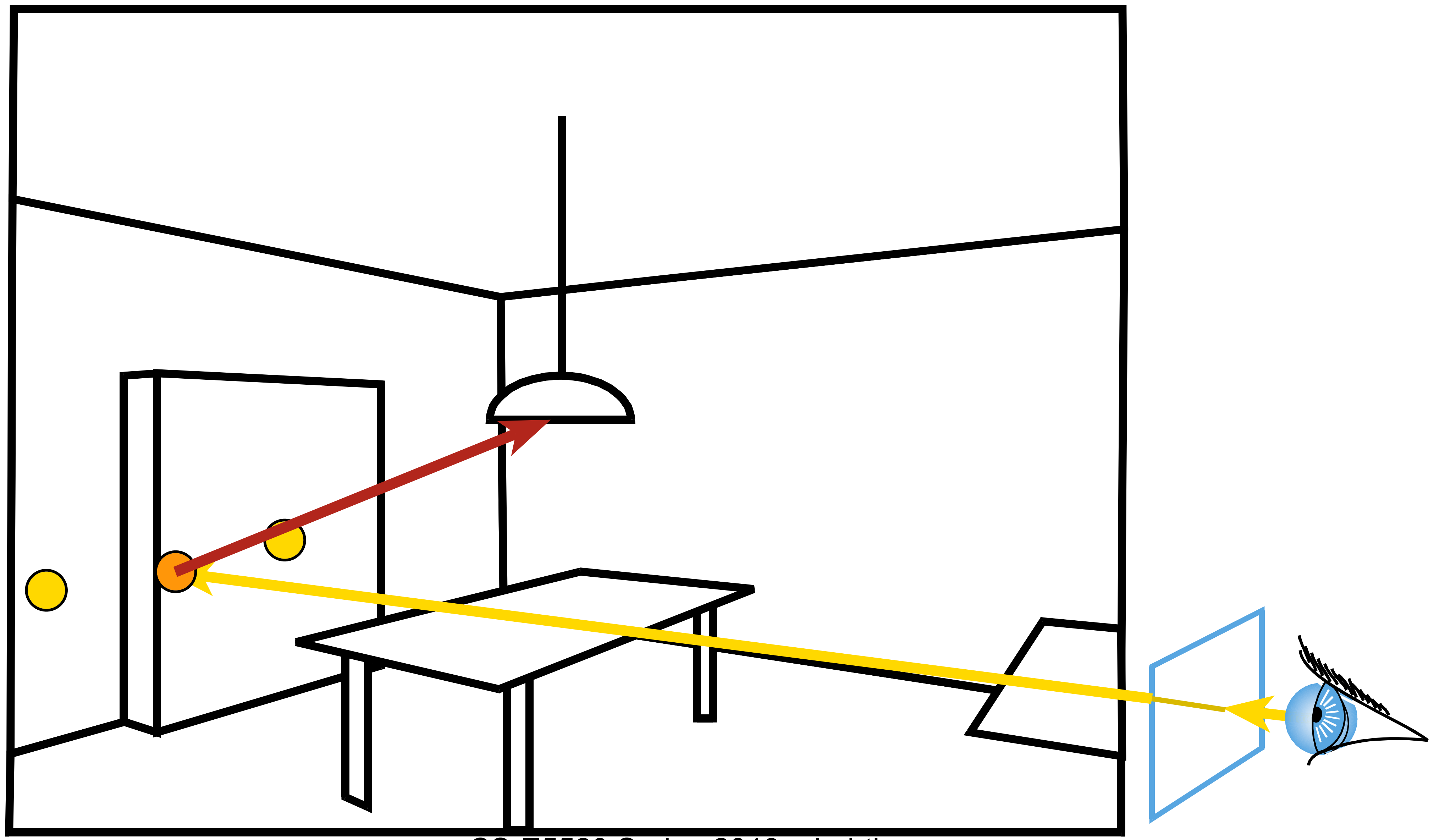
# Irradiance Caching [Ward 1988]

- Indirect illumination is most often smooth



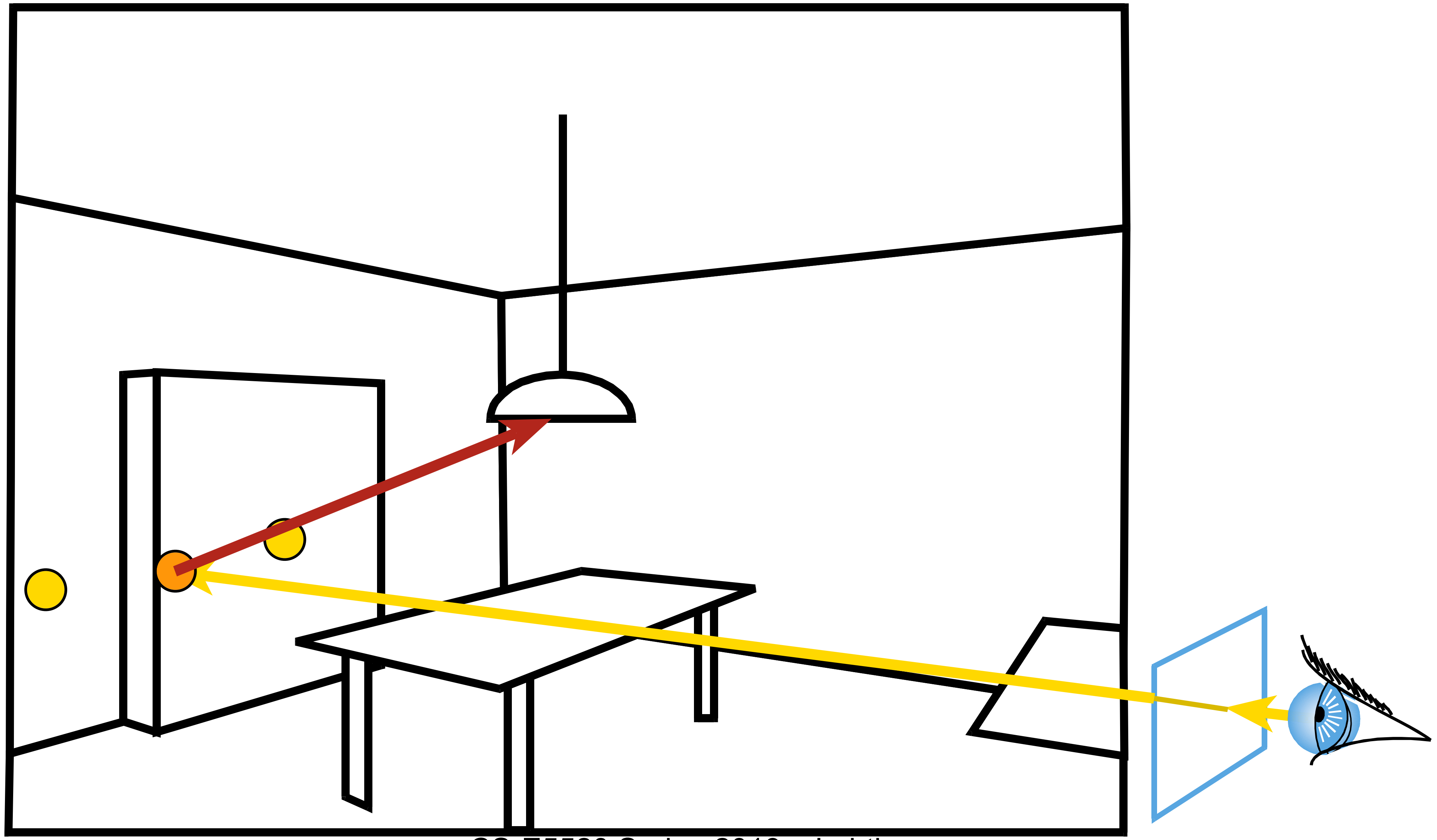
## Irradiance Caching [Ward 1988]

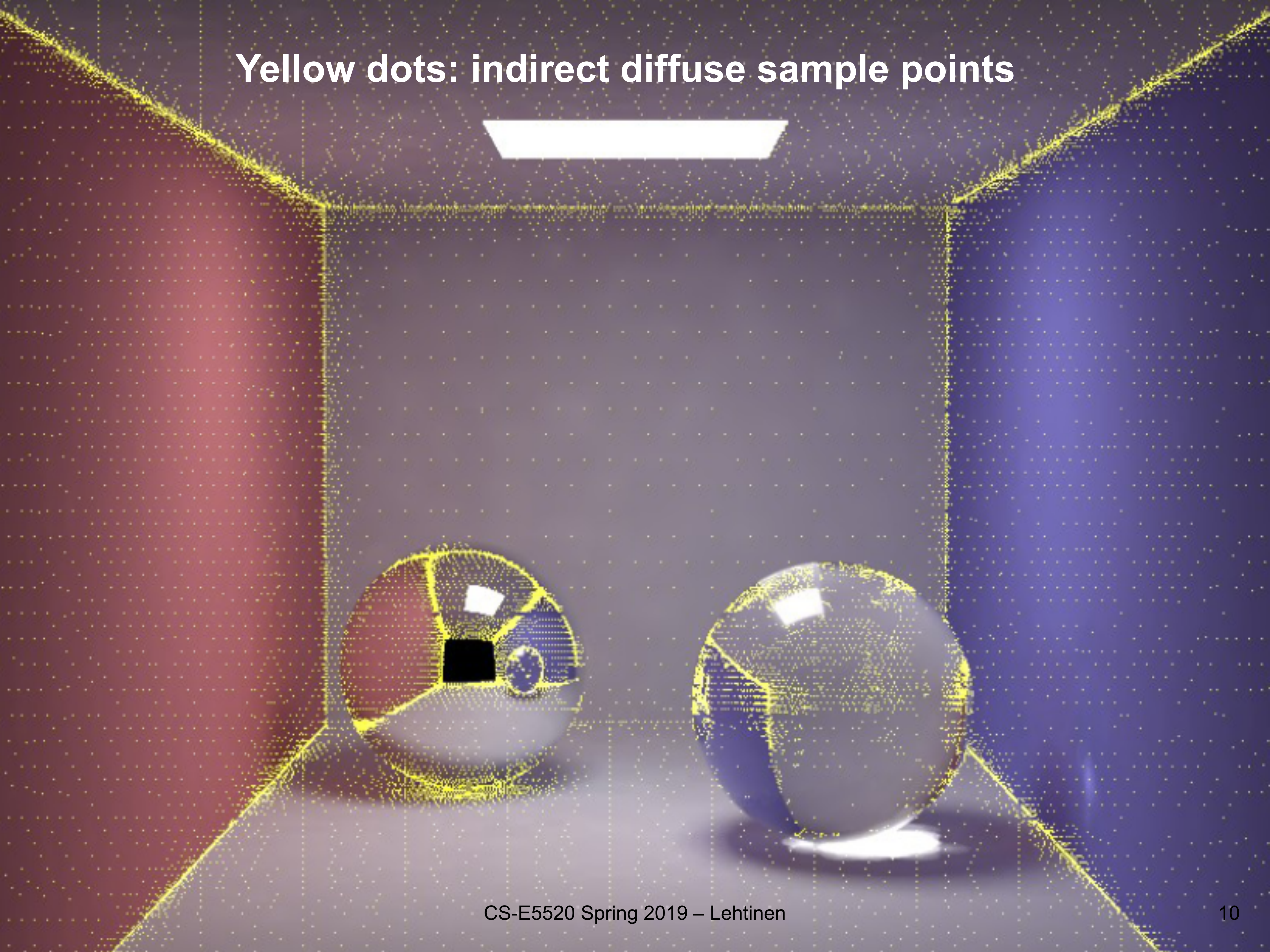
- Indirect illumination is most often smooth
  ==> Sample sparsely, interpolate nearby values



## Irradiance Caching [Ward 1988]

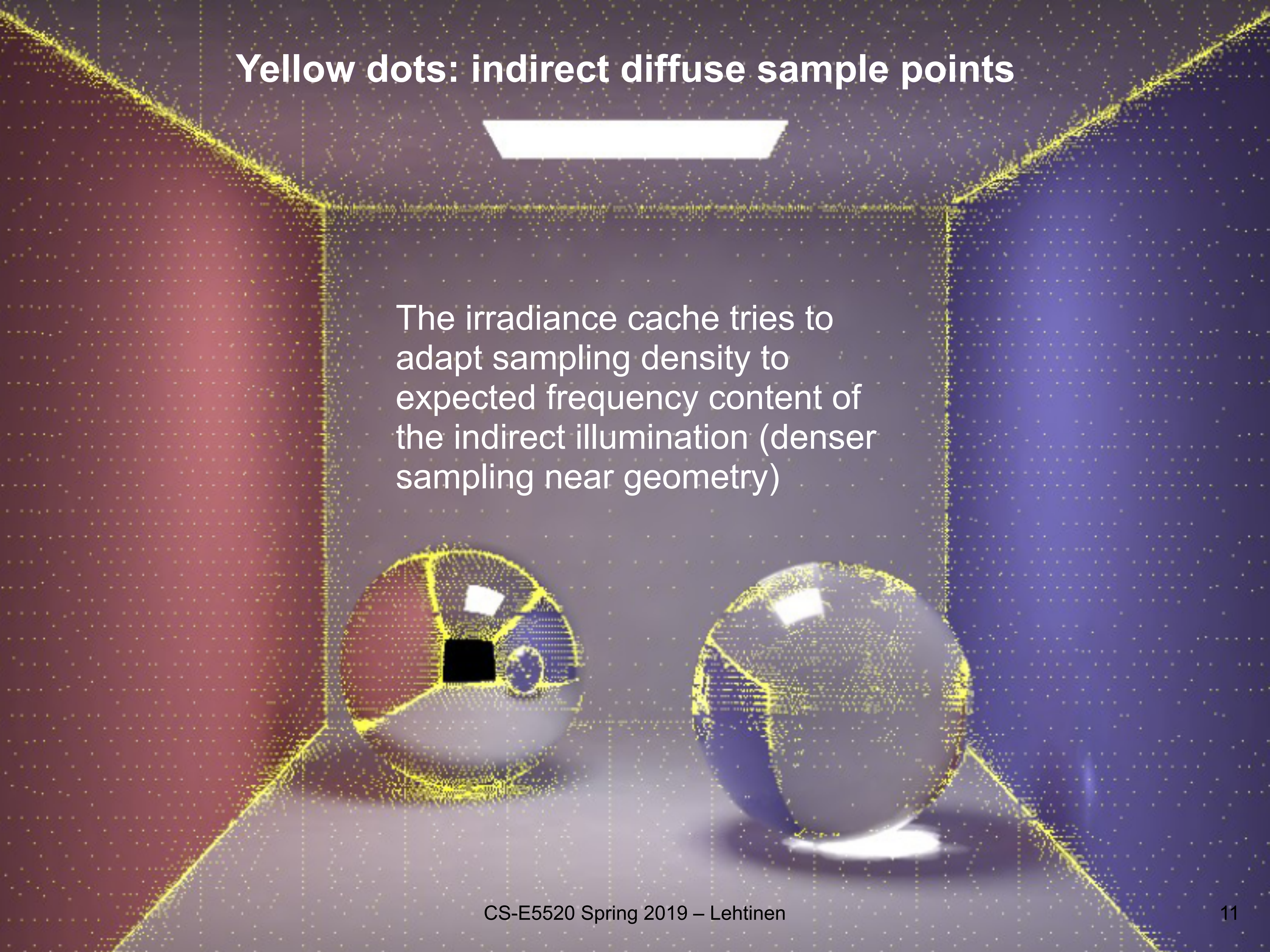
- Store the indirect illumination sparsely on surfaces
- Interpolate existing cached values
- But do full calculation for direct lighting





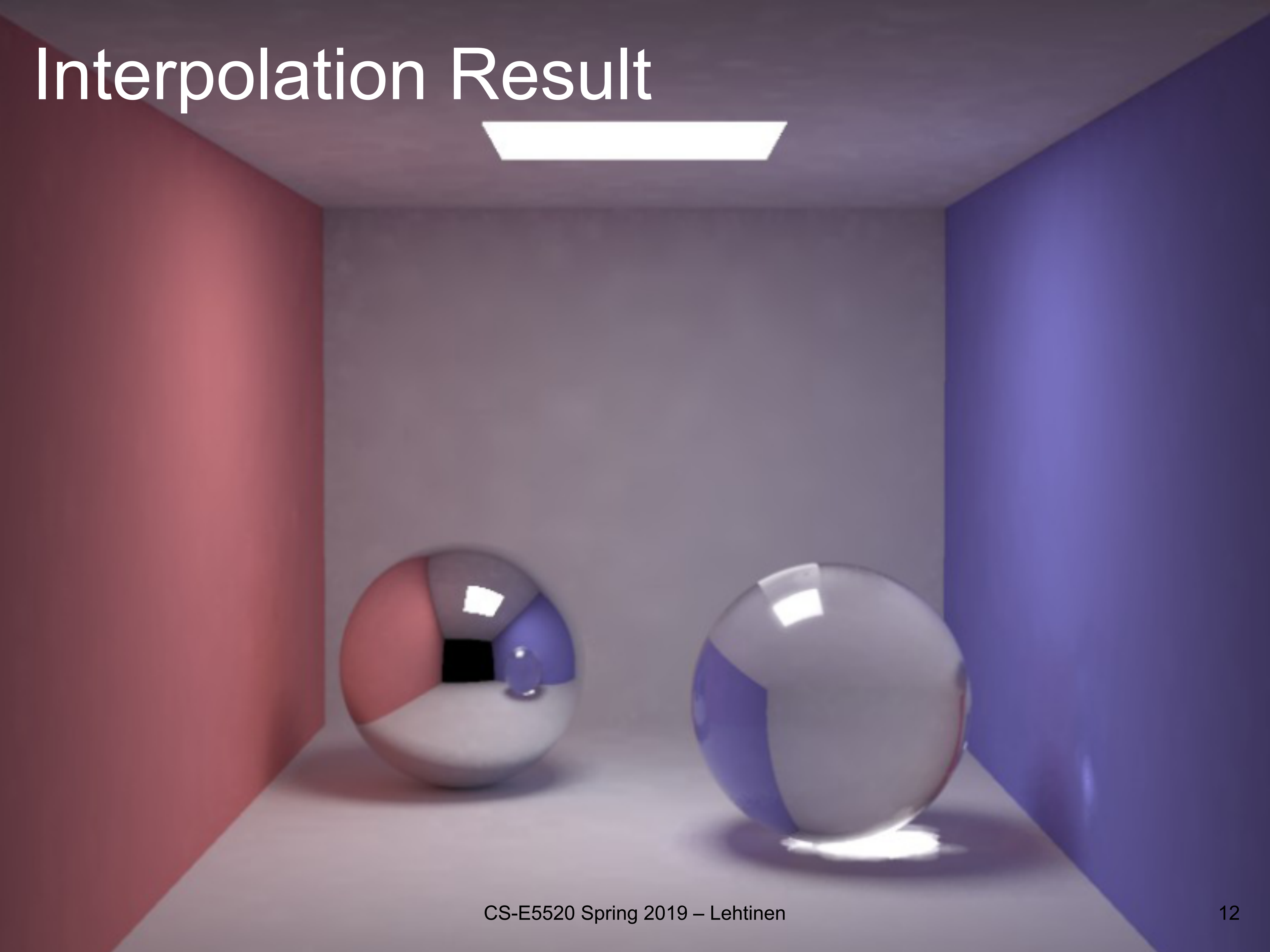
## Yellow dots: indirect diffuse sample points

## Yellow dots: indirect diffuse sample points



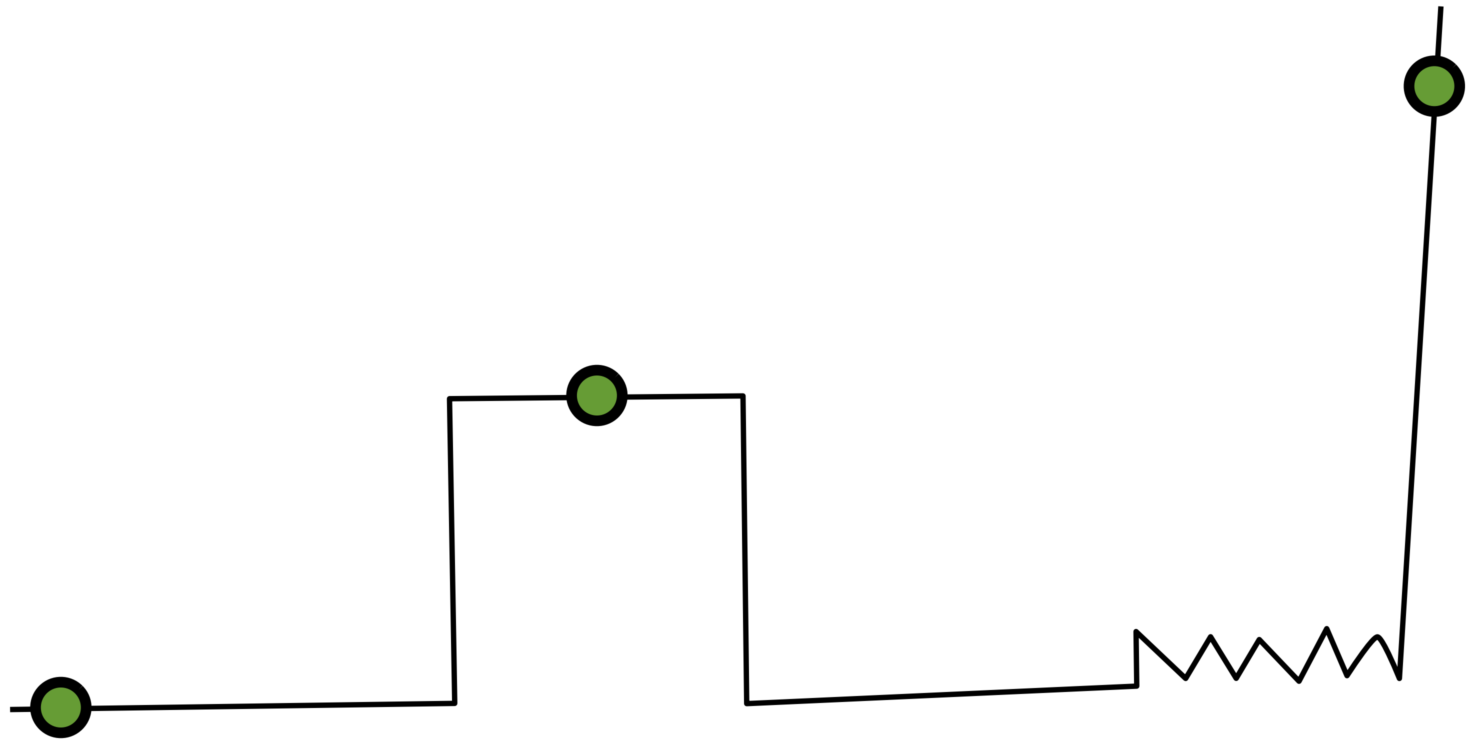
The irradiance cache tries to adapt sampling density to expected frequency content of the indirect illumination (denser sampling near geometry)

# Interpolation Result



## IC “Error Metric”

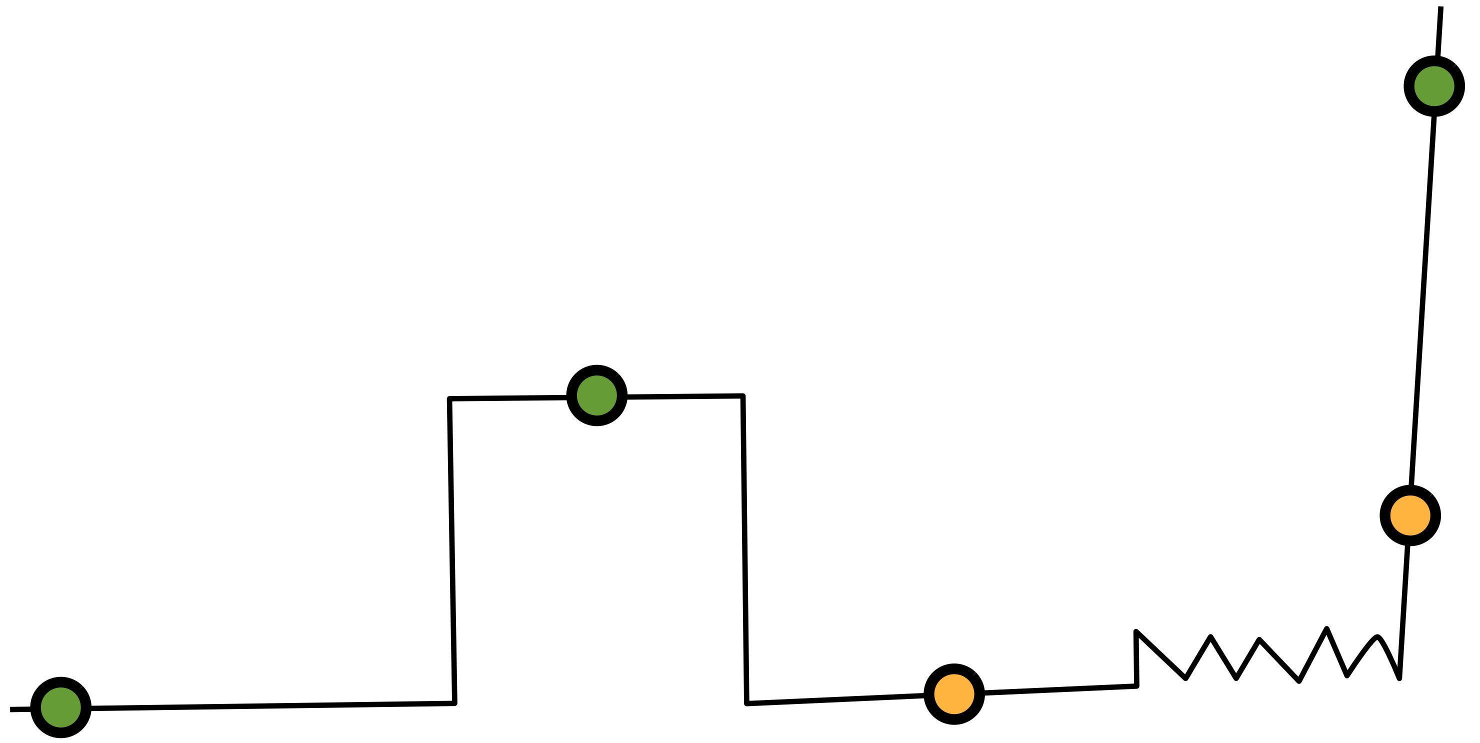
- The Irradiance Cache tries to predict the expected variation in indirect irradiance by looking at how far the surface point is from other, nearby geometry



**probably pretty smooth** ●

# IC "Error Metric"

- The Irradiance Cache tries to predict the expected variation in indirect irradiance by looking at how far the surface point is from other, nearby geometry

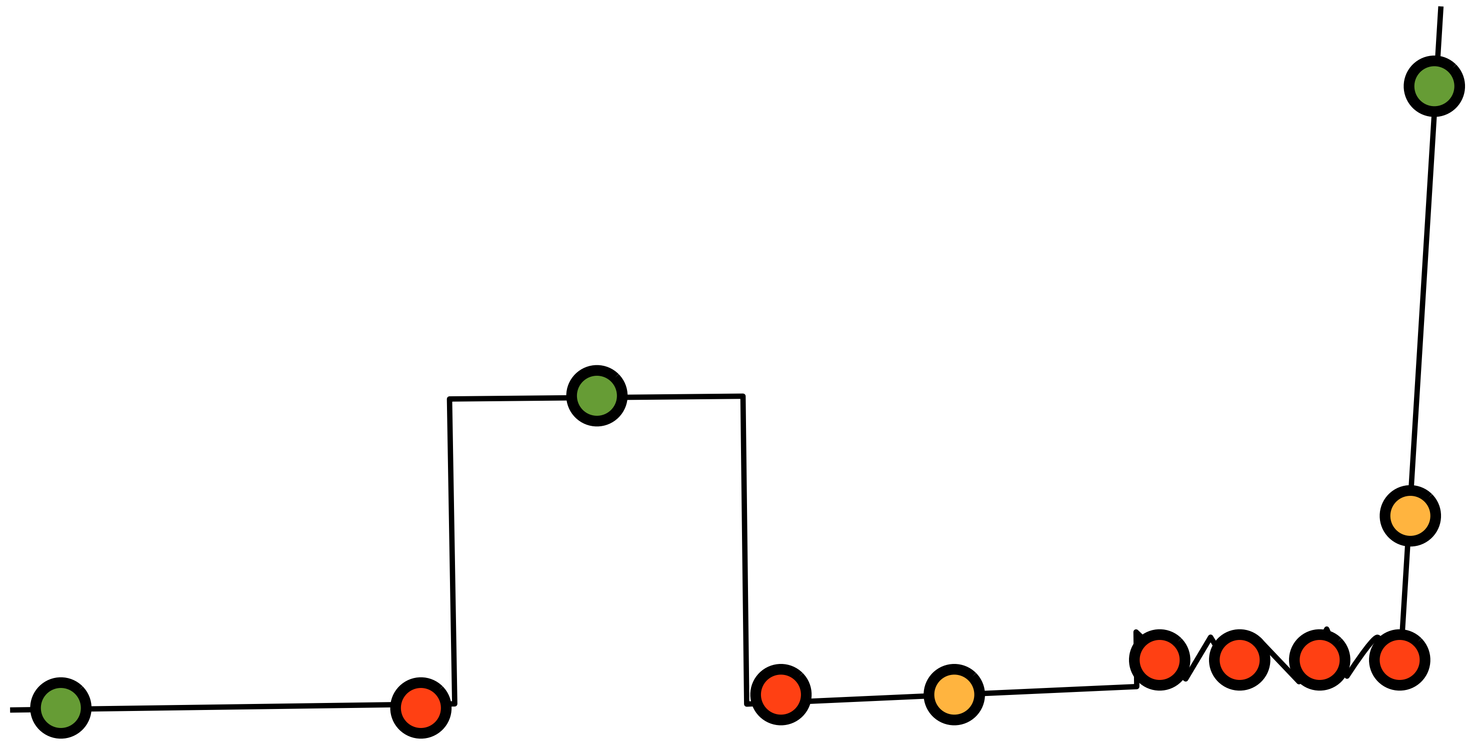


probably pretty smooth 🟢

maybe not so smooth 🟠

# IC "Error Metric"

- The Irradiance Cache tries to predict the expected variation in indirect irradiance by looking at how far the surface point is from other, nearby geometry



🟢 probably pretty smooth

🟠 maybe not so smooth

🔴 could be hairy

## The "Error Metric"

- Measures the *harmonic mean distance $D_h$* to other geometry in the above hemisphere, modulated by difference of normals (see Greg's paper for details)
  - Harmonic mean defined as

$$\frac{1}{\int \frac{1}{x} f(x)\, \mathrm{d}x}$$

  - Some implementations use minimum distance as well

- Why quotes? It's not really a metric

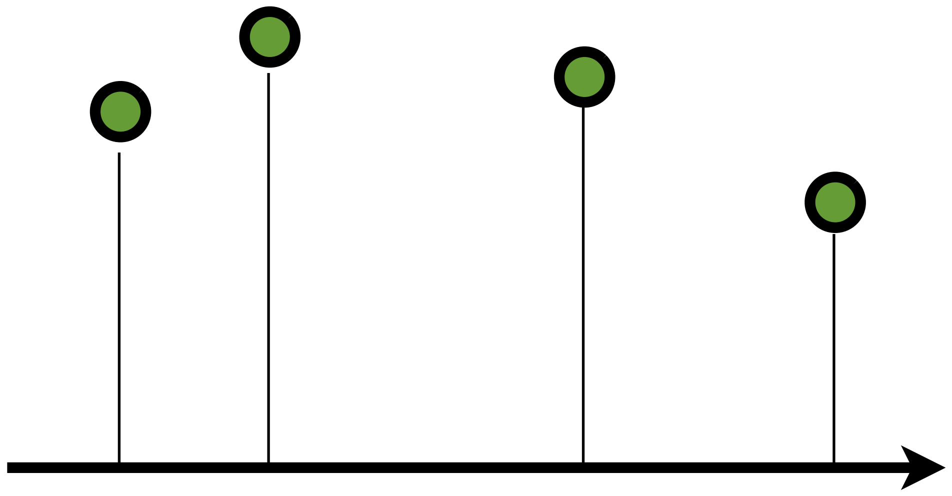
## Irradiance Caching Algorithm

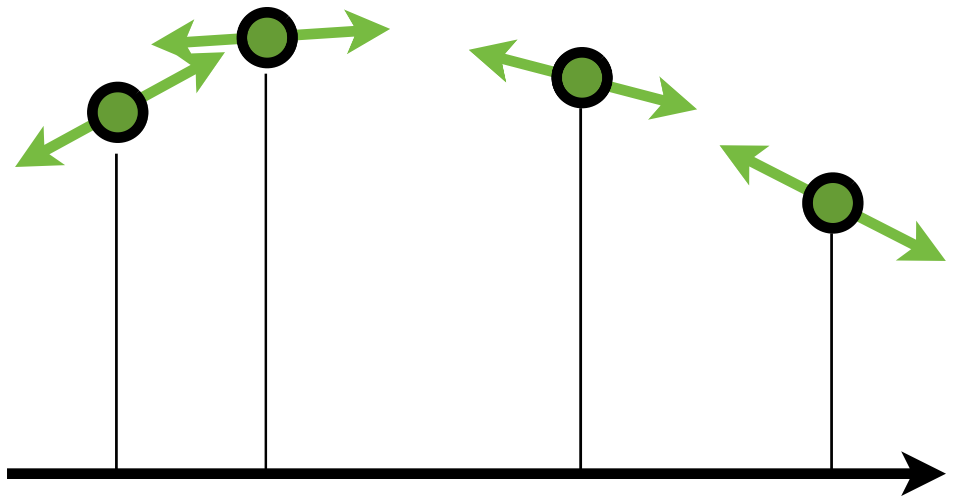
- Two passes
  - First pass: go over all pixels, measure $D_h$
    - If no previous cache entry is closer than threshold, add new cache entry
    - Compute its irradiance accurately using tons of samples
    - Store radiance with a 3D point in a BVH
    - BVH for points..? Easy
  - Second pass: render all pixels, interpolate irradiance from cache entries
    - Use weights such that cache entries that are nearby and whose normals are similar contribute more

- Original paper is not two-pass, but it makes *no sense*

# Extensions

- In a follow-up paper, Ward introduces irradiance gradients to further aid reconstruction
  - Idea: what if don't just know the function values at the samples points, but also how fast it's changing?
  - "Higher-order interpolation", particularly the Hermite kind


plain vanilla irradiance cache


with gradients

DreamWorks

## IC WAS used in production, too

Read this paper that describes the use of
vector irradiance and simplified geometry for indirect light in Shrek 2
*These days people mostly do path tracing + denoising*

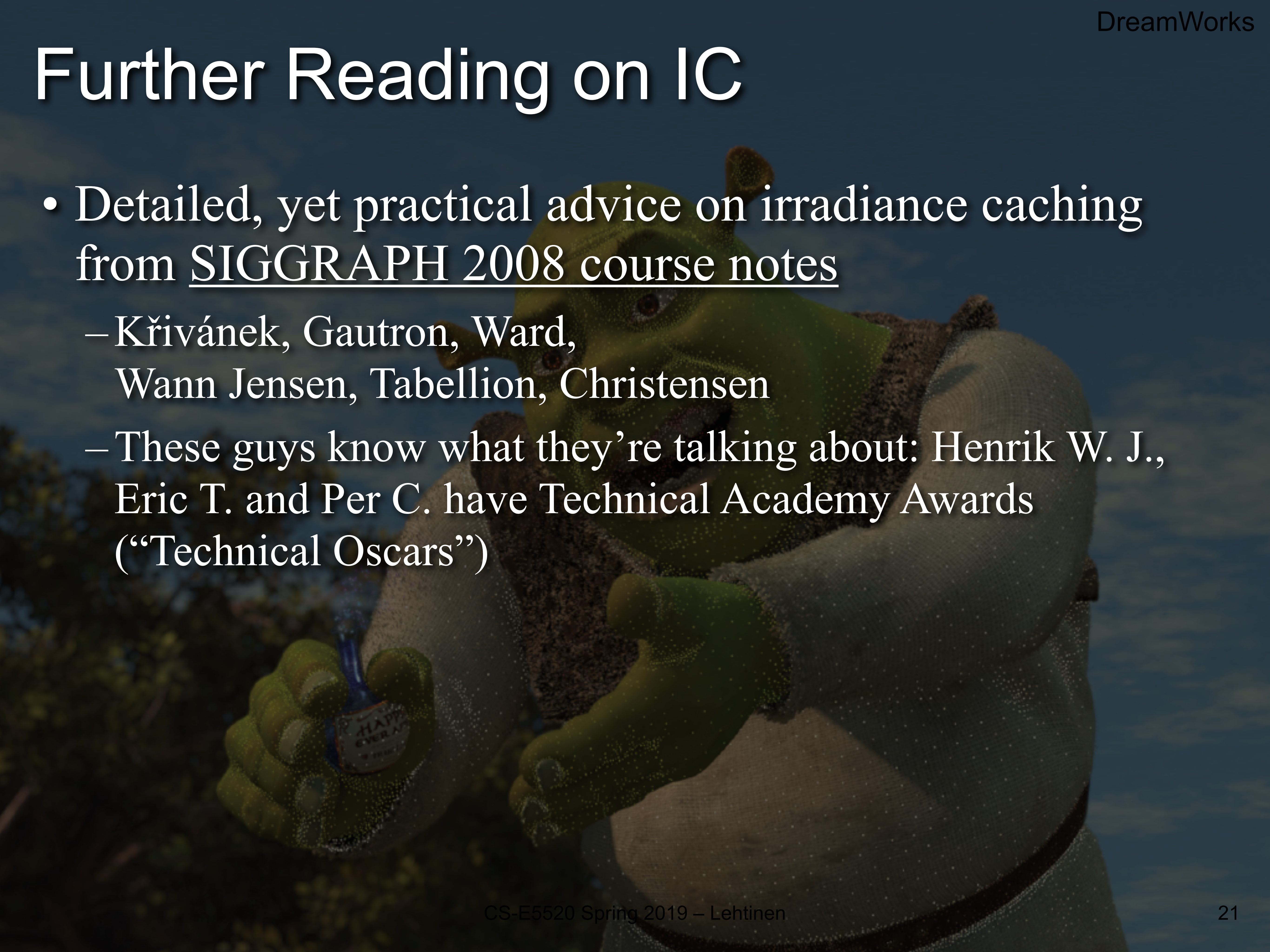
HAPPY
EVER A

DreamWorks

## IC used in production, too

Read this paper that describes the use of vector irradiance and simplified geometry for indirect light in Shrek 2

Eric Tabellion,
our rendering compo juror in 2013!



## Further Reading on IC

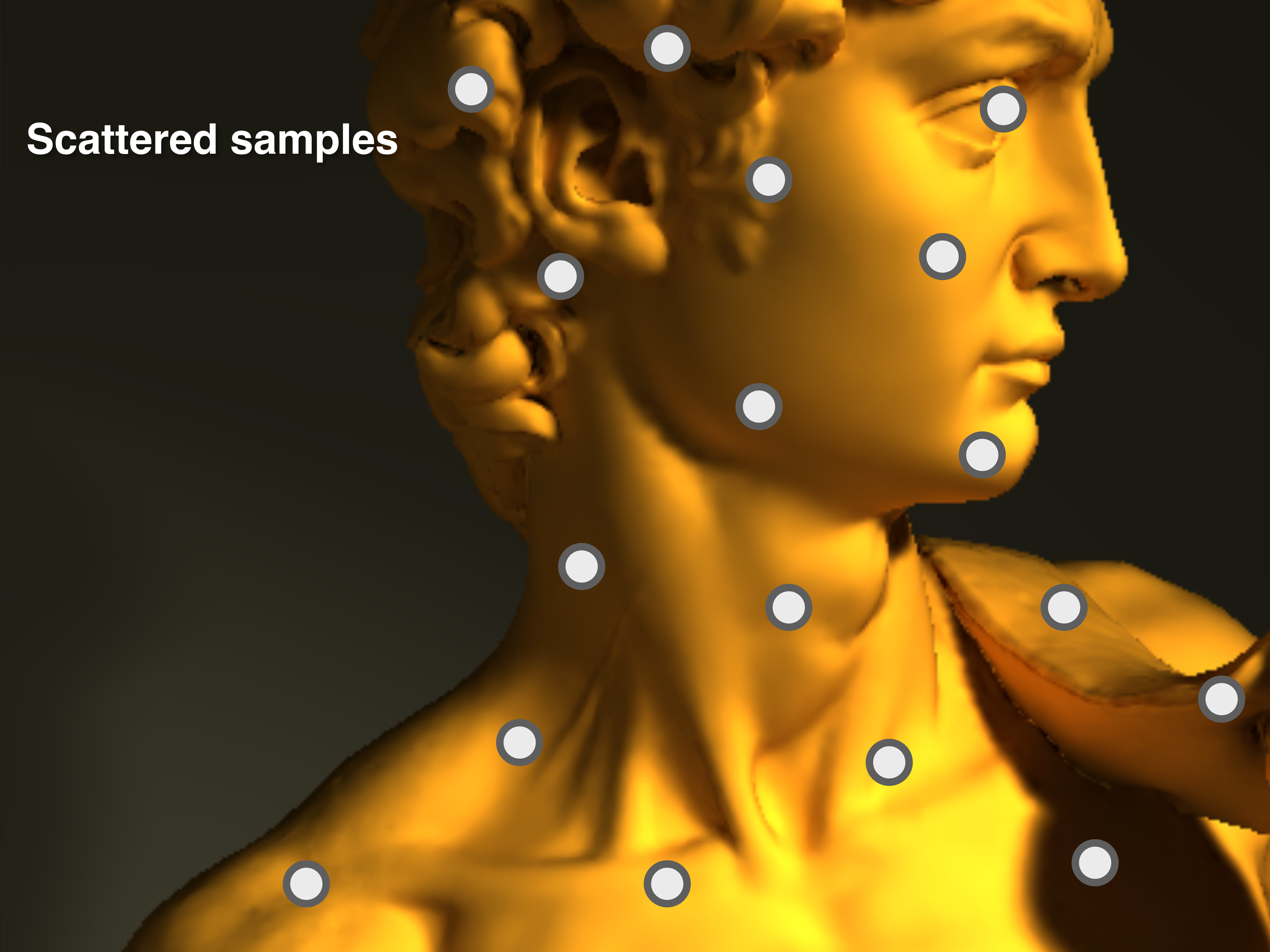
- Detailed, yet practical advice on irradiance caching from SIGGRAPH 2008 course notes
  - Křivánek, Gautron, Ward, Wann Jensen, Tabellion, Christensen
  - These guys know what they're talking about: Henrik W. J., Eric T. and Per C. have Technical Academy Awards ("Technical Oscars")

## Related Algorithm

- Can also construct “hierarchical basis functions” from points alone
  - Lehtinen, Zwicker, Turquin, Kontkanen, Durand, Aila 2008
- Hierarchy allows computation to proceed coarse to fine
  - Connection to IC: Also interpolation from points
- This is the last extra credit task in the radiosity assignment

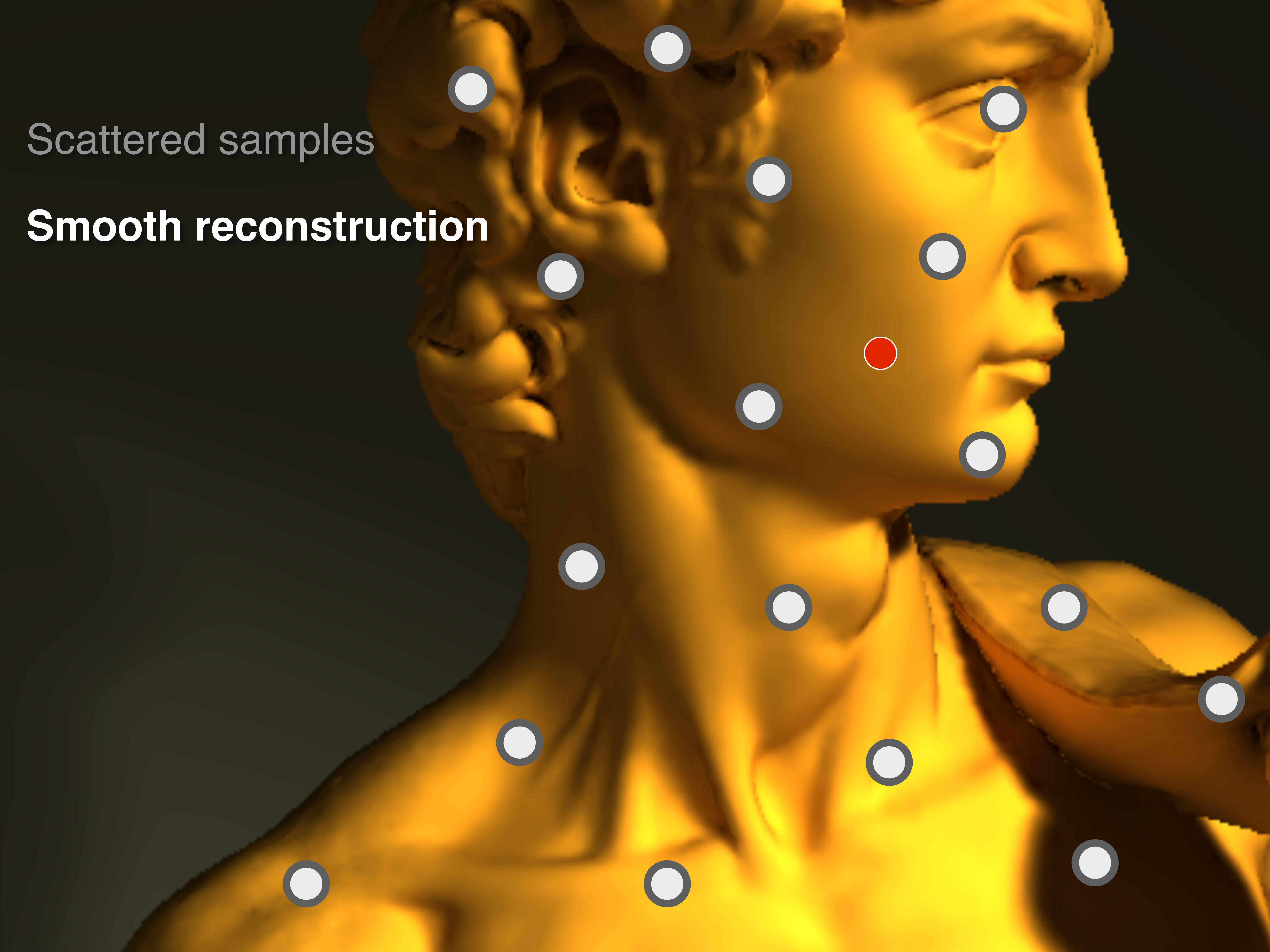
## Overview of our approach

Scattered samples



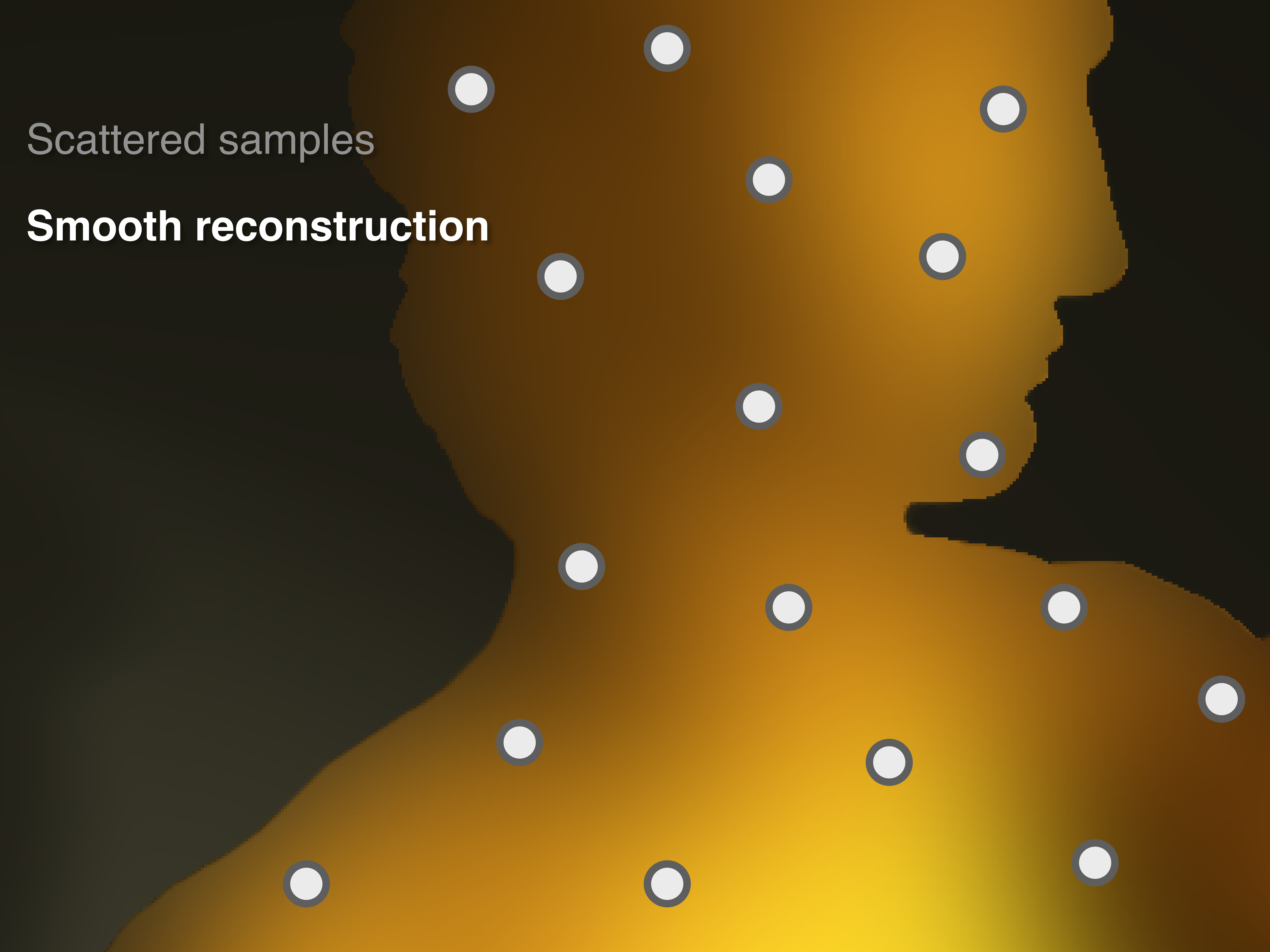
Scattered samples

**Smooth reconstruction**



Scattered samples

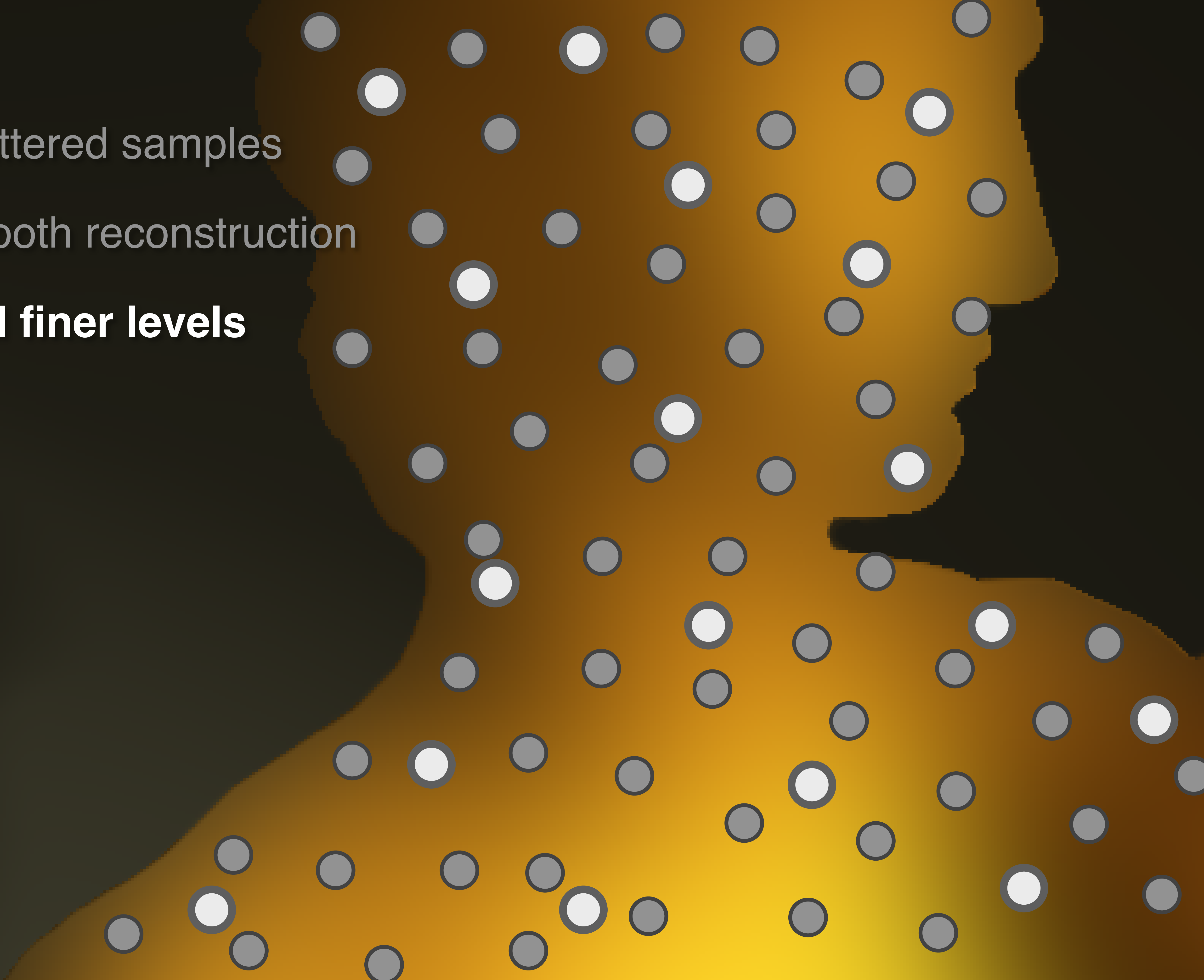
**Smooth reconstruction**

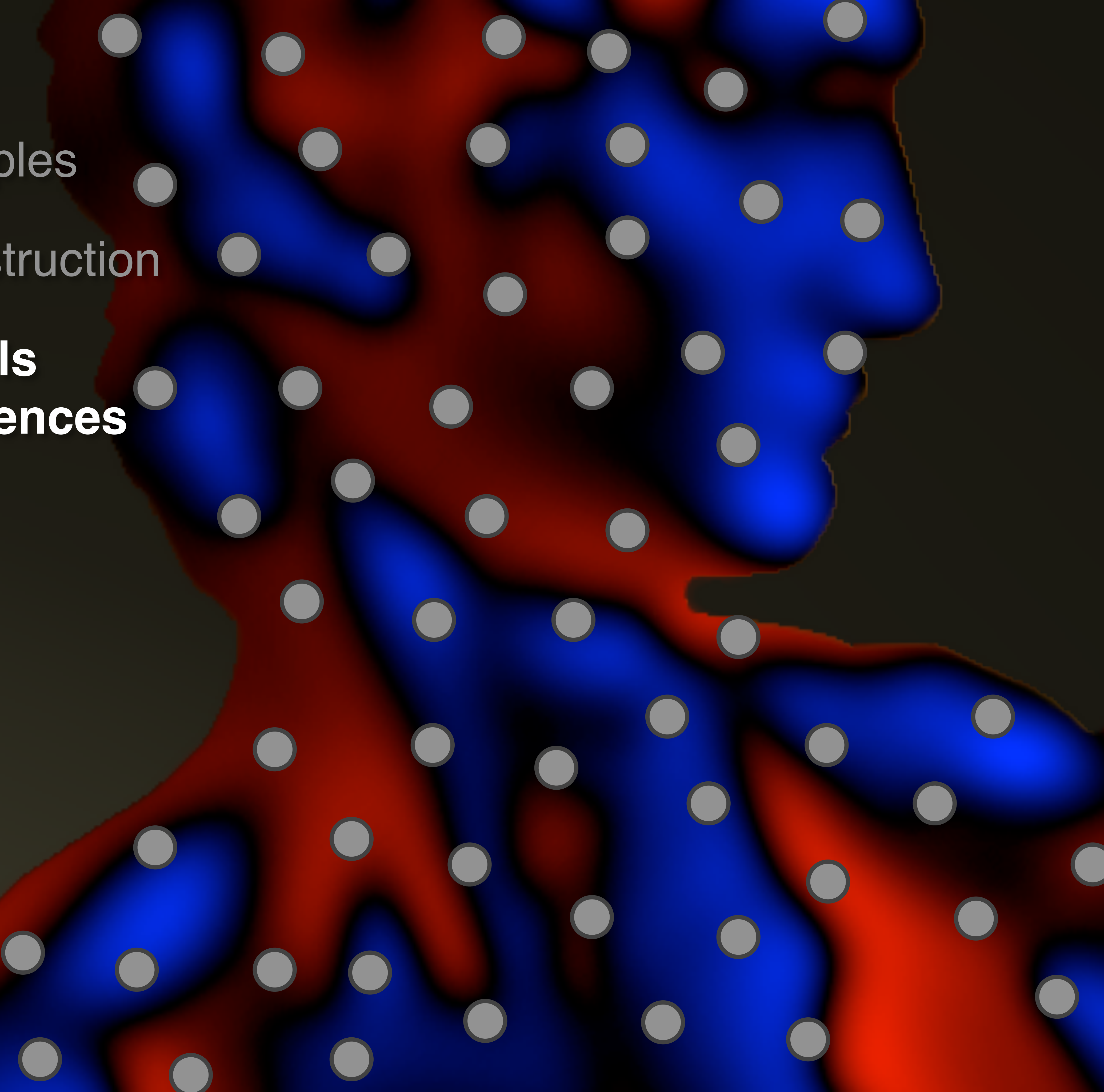


Scattered samples

Smooth reconstruction

**Add finer levels**



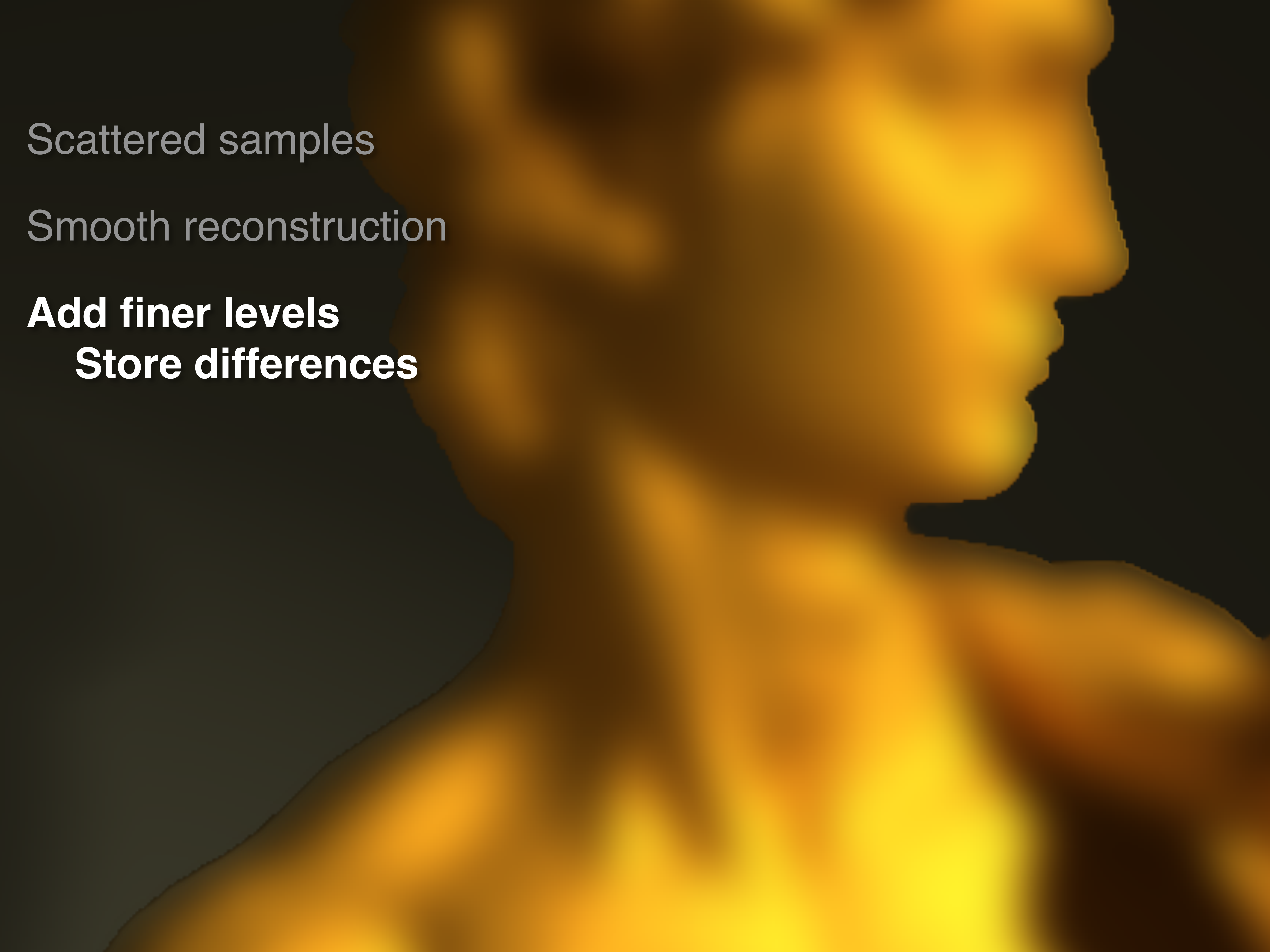


Scattered samples

Smooth reconstruction

**Add finer levels**

**Store differences**



Scattered samples

Smooth reconstruction

**Add finer levels**

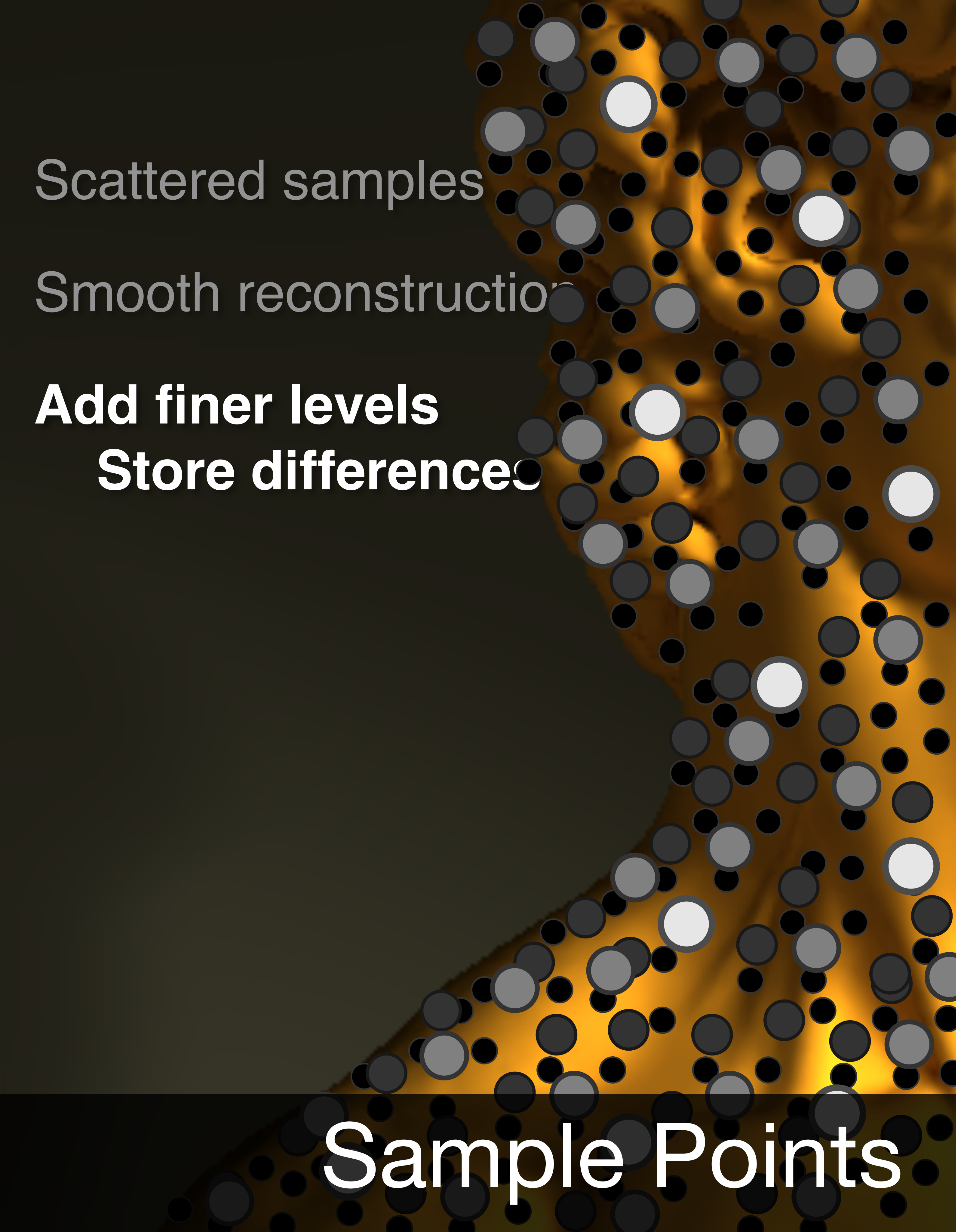
- **Store differences**

Scattered samples

Smooth reconstruction

**Add finer levels**

**Store differences**



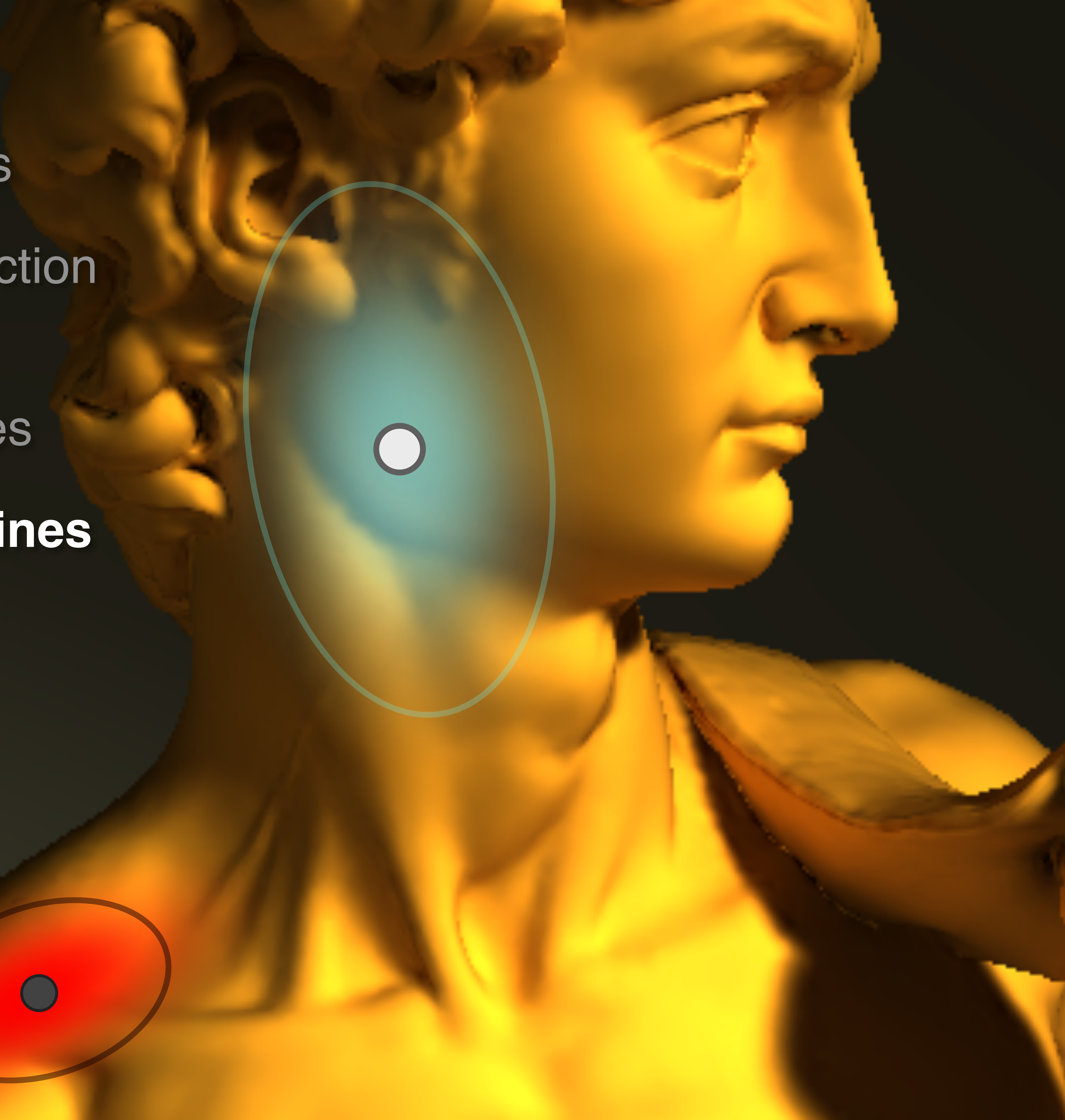
Scattered samples

Smooth reconstruction

Add finer levels

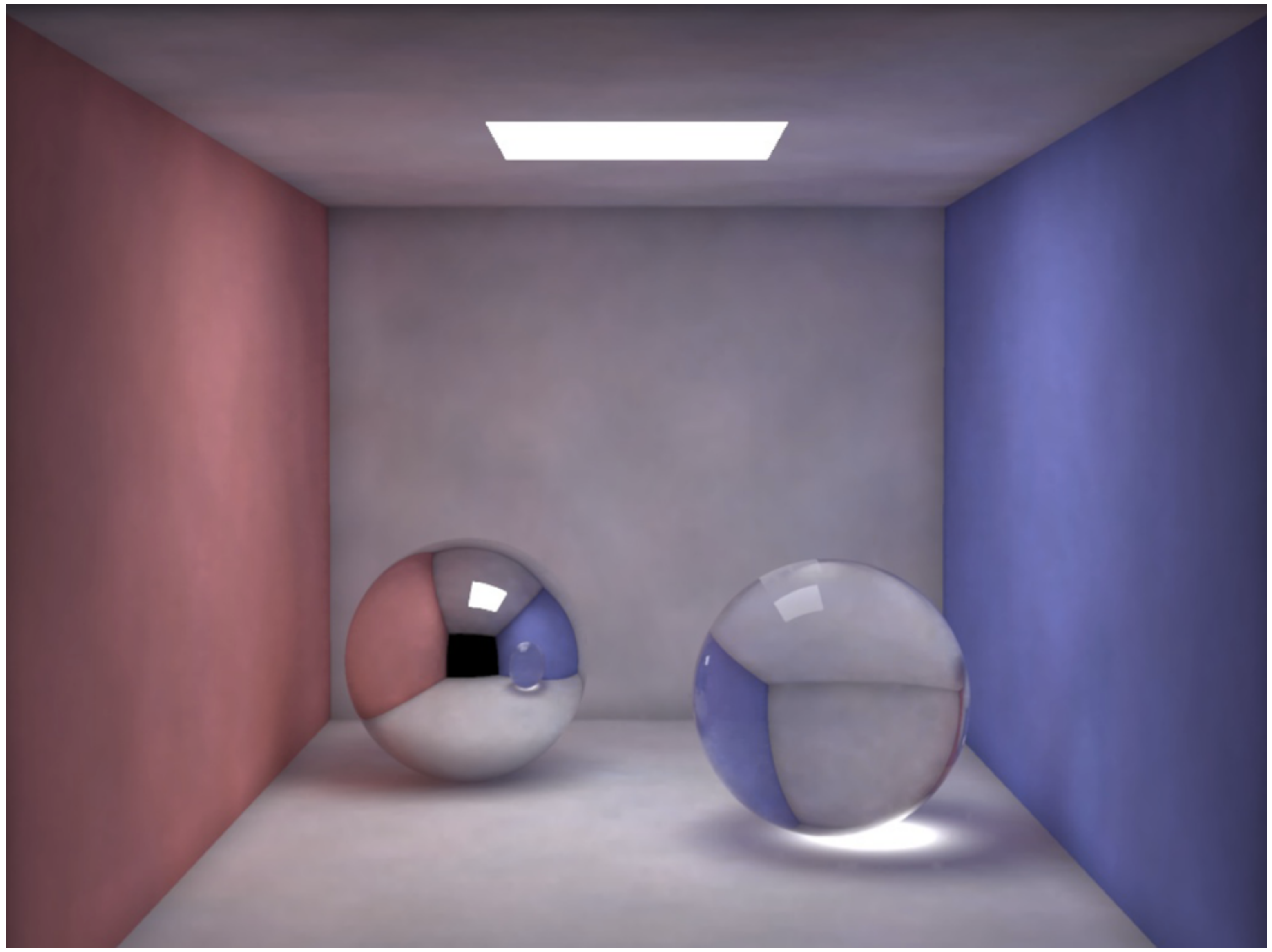
Store differences

**Each sample defines a basis function**



# Irradiance Cache Problems

- When interpolating radiance, any noise in the cached values will get transformed into low-frequency splotchy artifacts
- You need to compute the IC samples with really high quality
  - Tons of hemisphere rays per sample



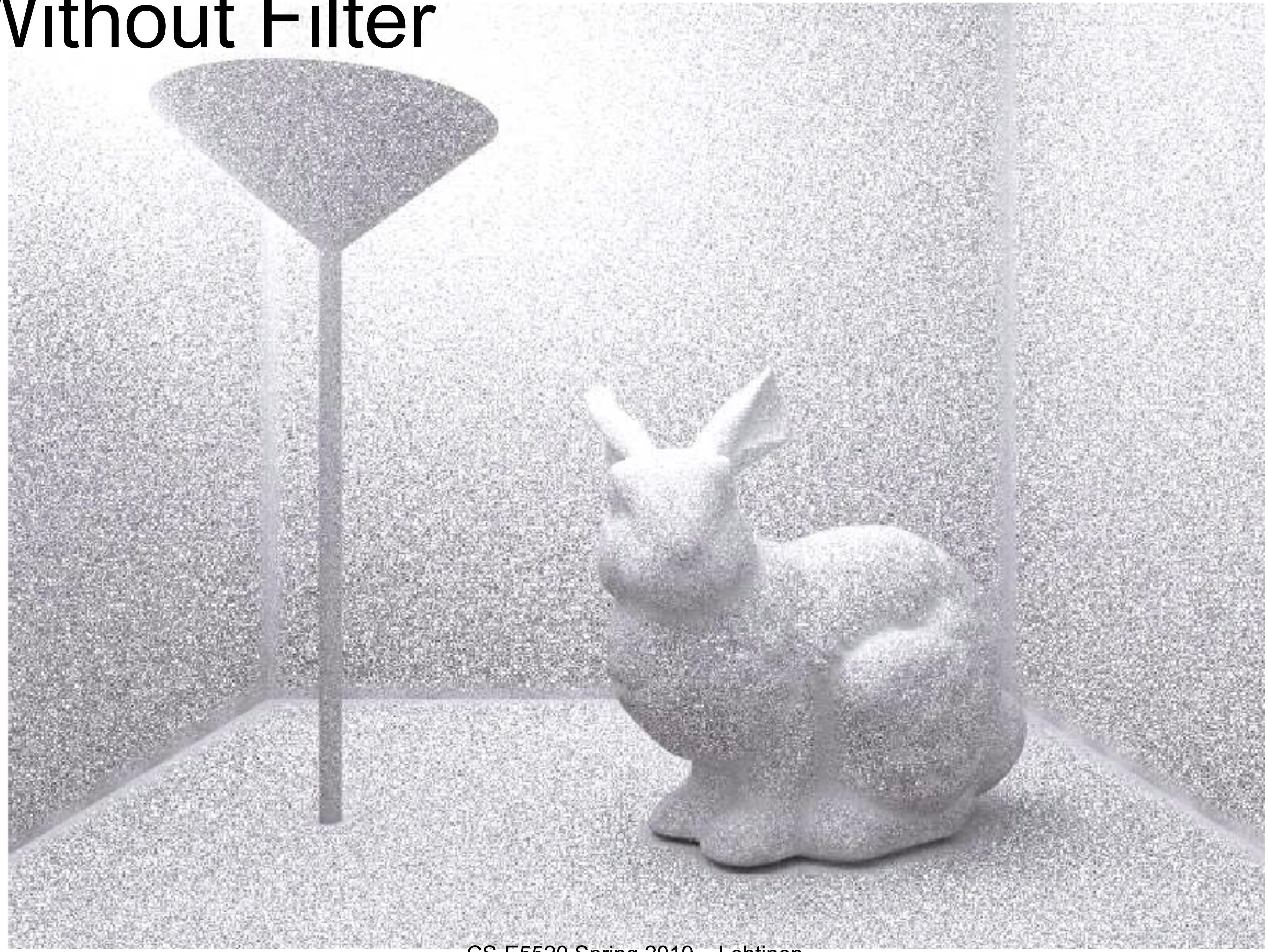
(Image from a photon mapper, but it looks pretty much the same)

# Irradiance Filtering

- We can instead compute a very noisy estimate *at each pixel*, and then try to remove noise by filtering
  - In contrast: IC interpolates *sparse*, good samples
  - Kontkanen, Räsänen, Keller 2004
  - And lots of subsequent work, like
    - A trous wavelets (youtube link)
    - "Random Parameter Filtering"
    - "Adaptive Manifolds"

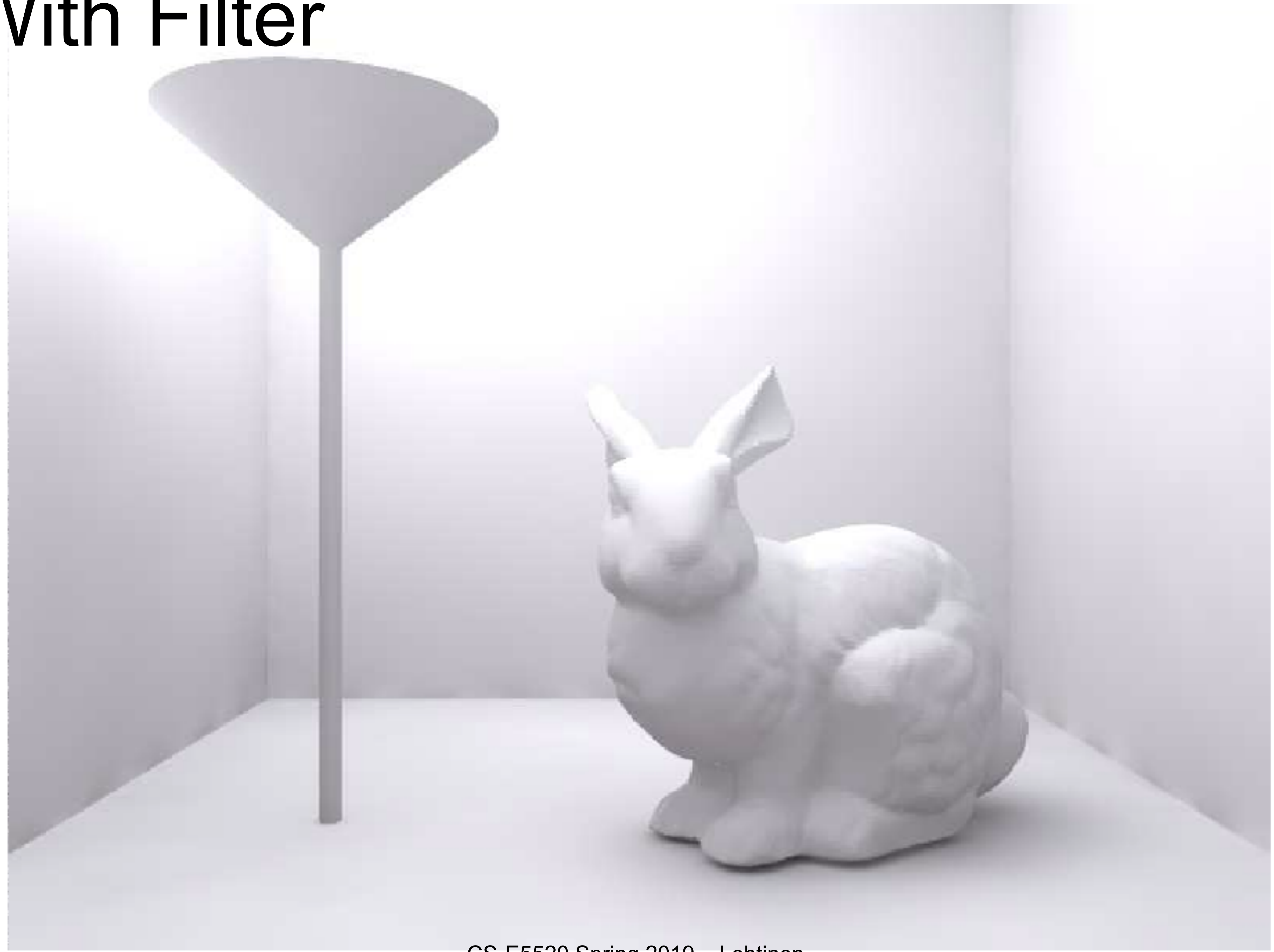
- And if you allow yourself a little more domain knowledge, you get still further
  - Our stuff from SIGGRAPH 2012

# Without Filter



Kontkanen et al.

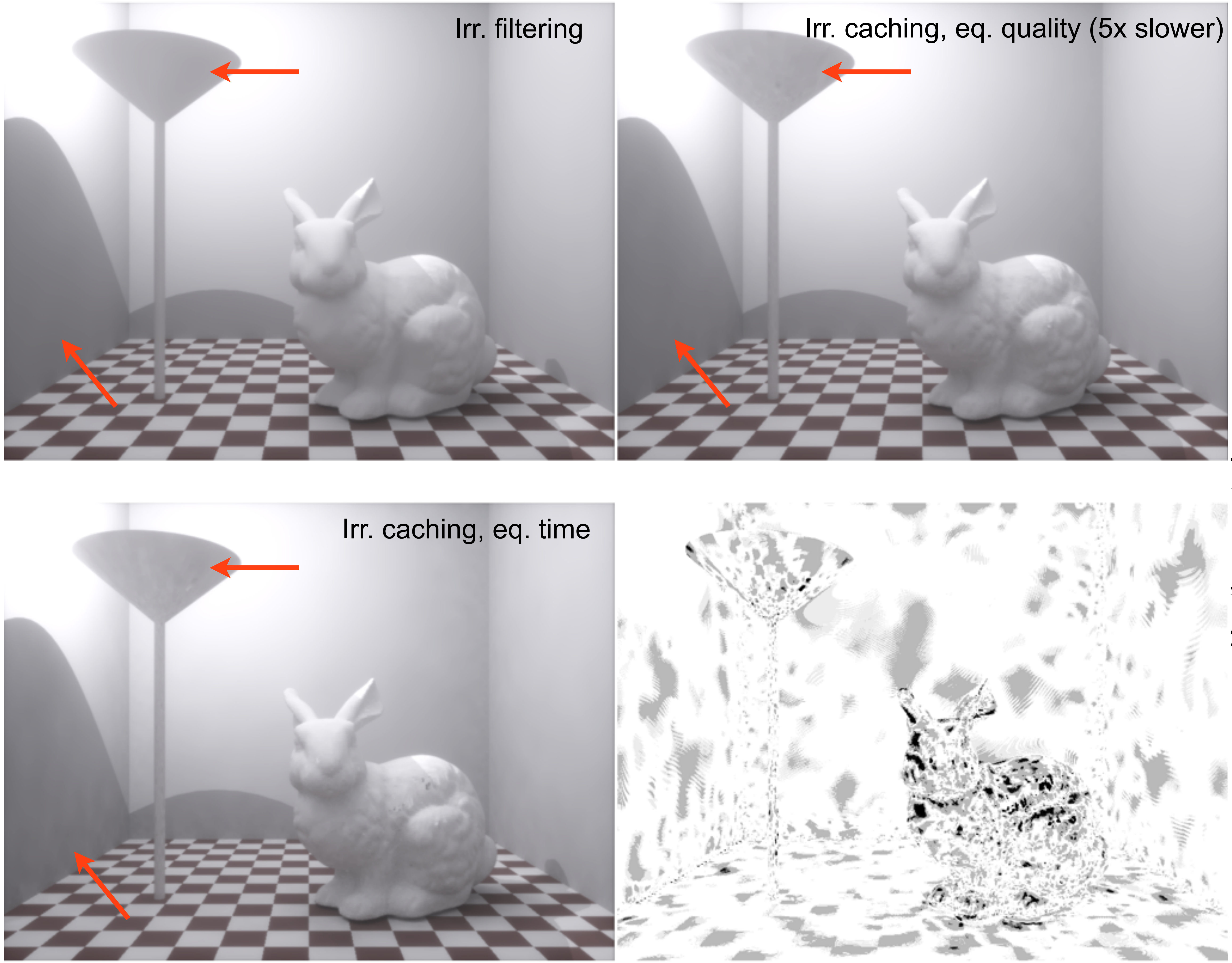
# With Filter



Kontkanen et al.

# A Trous Video

- None of these methods completely eliminate low-frequency noise
  - But they make the situation a whole lot better!



Kontkanen et al.

# Current: Denoise using CNNs

(SIGGRAPH 2017)

## Kernel-Predicting Convolutional Networks for Denoising Monte Carlo Renderings

STEVE BAKO*, University of California, Santa Barbara
THIJS VOGELS*, ETH Zürich & Disney Research
BRIAN MCWILLIAMS, Disney Research
MARK MEYER, Pixar Animation Studios
JAN NOVÁK, Disney Research
ALEX HARVILL, Pixar Animation Studios
PRADEEP SEN, University of California, Santa Barbara
TONY DEROSE, Pixar Animation Studios
FABRICE ROUSSELLE, Disney Research

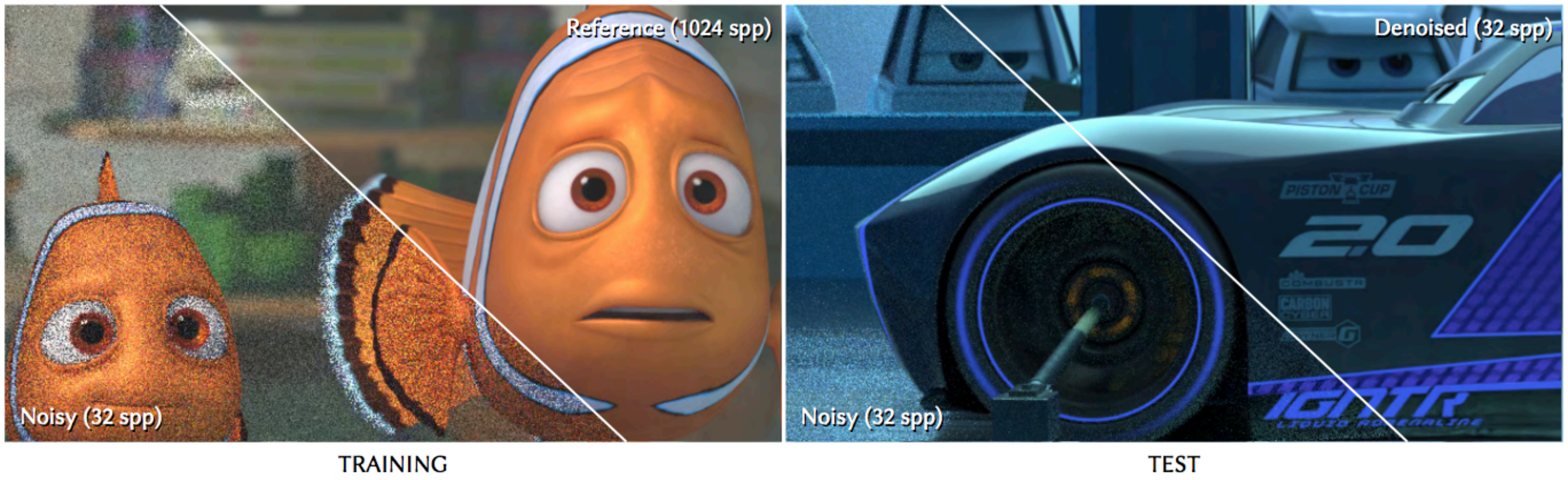


Fig. 1. We introduce a deep learning approach for denoising Monte Carlo-rendered images that produces high-quality results suitable for production. We train a convolutional neural network to learn the complex relationship between noisy and reference data across a large set of frames with varying distributed effects from the film *Finding Dory* (left). The trained network can then be applied to denoise new images from other films with significantly different style and content, such as *Cars 3* (right), with production-quality results.

# Another Take: NVIDIA's Recurrent RNN denoiser (also SIGGRAPH '17)

## Interactive Reconstruction of Monte Carlo Image Sequences using a Recurrent Denoising Autoencoder

CHAKRAVARTY R. ALLA CHAITANYA, NVIDIA, University of Montreal and McGill University
ANTON S. KAPLANYAN, NVIDIA
CHRISTOPH SCHIED, NVIDIA and Karlsruhe Institute of Technology
MARCO SALVI, NVIDIA
AARON LEFOHN, NVIDIA
DEREK NOWROUZEZAHRAI, McGill University
TIMO AILA, NVIDIA

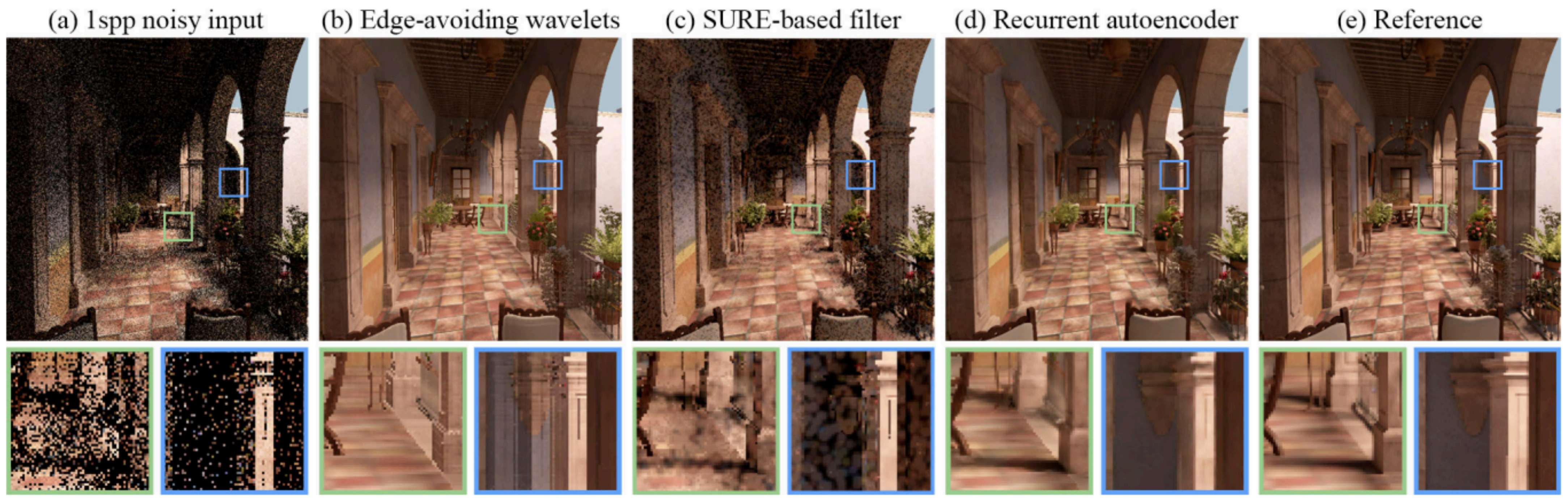


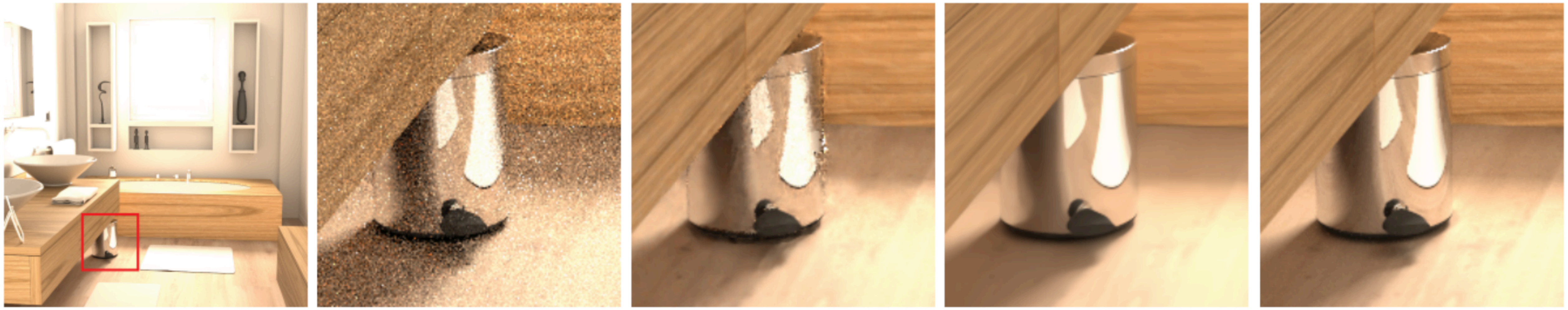
Fig. 1. Left to right: (a) noisy image generated using path-traced global illumination with one indirect inter-reflection and 1 sample/pixel; (b) edge-avoiding wavelet filter [Dammertz et al. 2010] (10.3ms at 720p, SSIM: 0.7737); (c) SURE-based filter [Li et al. 2012] (74.2ms, SSIM: 0.5960); (d) our recurrent denoising autoencoder (54.9ms, SSIM: 0.8438); (e) reference path-traced image with 4096 samples/pixel.

# New SotA (SIGGRAPH 2019)

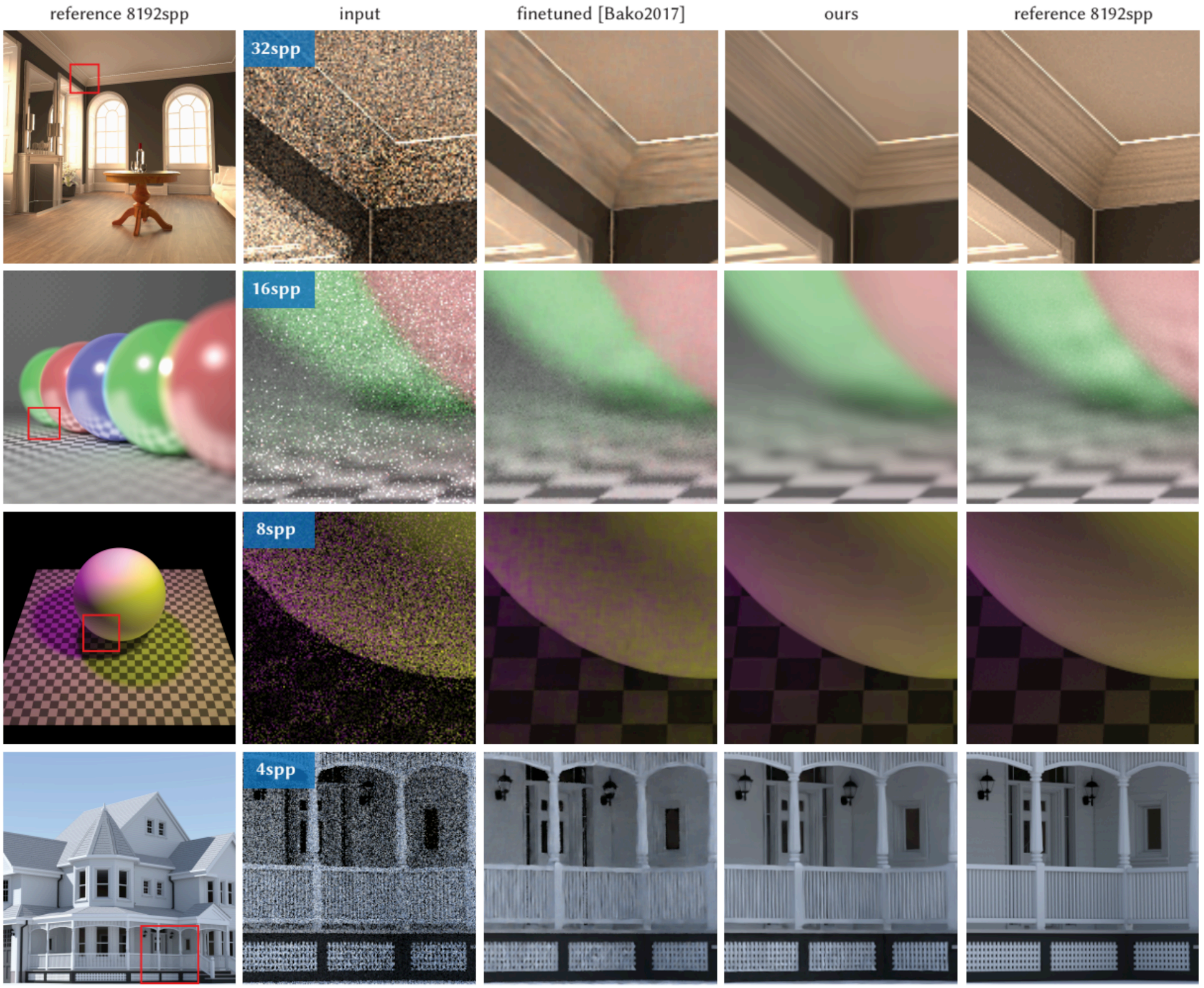
- (Ask me personally for a preprint)

## Sample-based Monte Carlo Denoising using a Kernel-Splatting Network

MICHAËL GHARBI, Adobe and MIT CSAIL
TZU-MAO LI, MIT CSAIL
MIIKA AITTALA, MIT CSAIL
JAAKKO LEHTINEN, Aalto University and NVIDIA
FRÉDO DURAND, MIT CSAIL



Gharbi et al. SIGGRAPH 2019



# Key ideas

- (1) CNN operates on both invidual samples and pixel-level aggregates, (2) predict scatter kernels, not gather

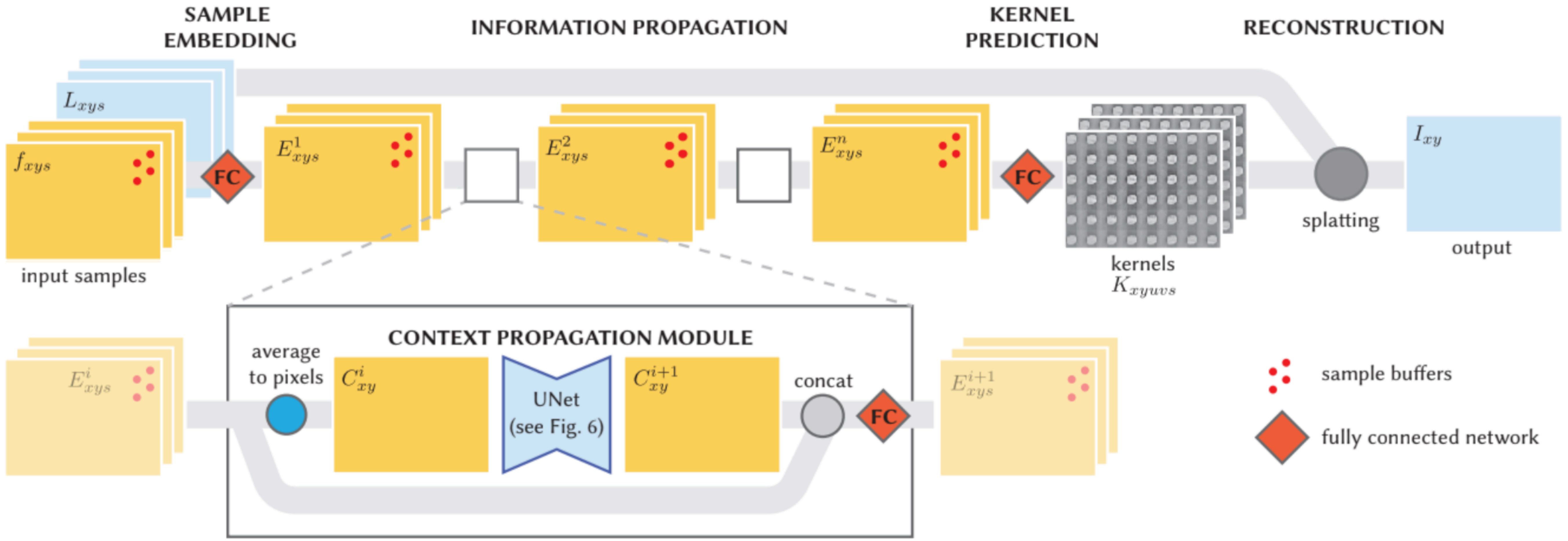
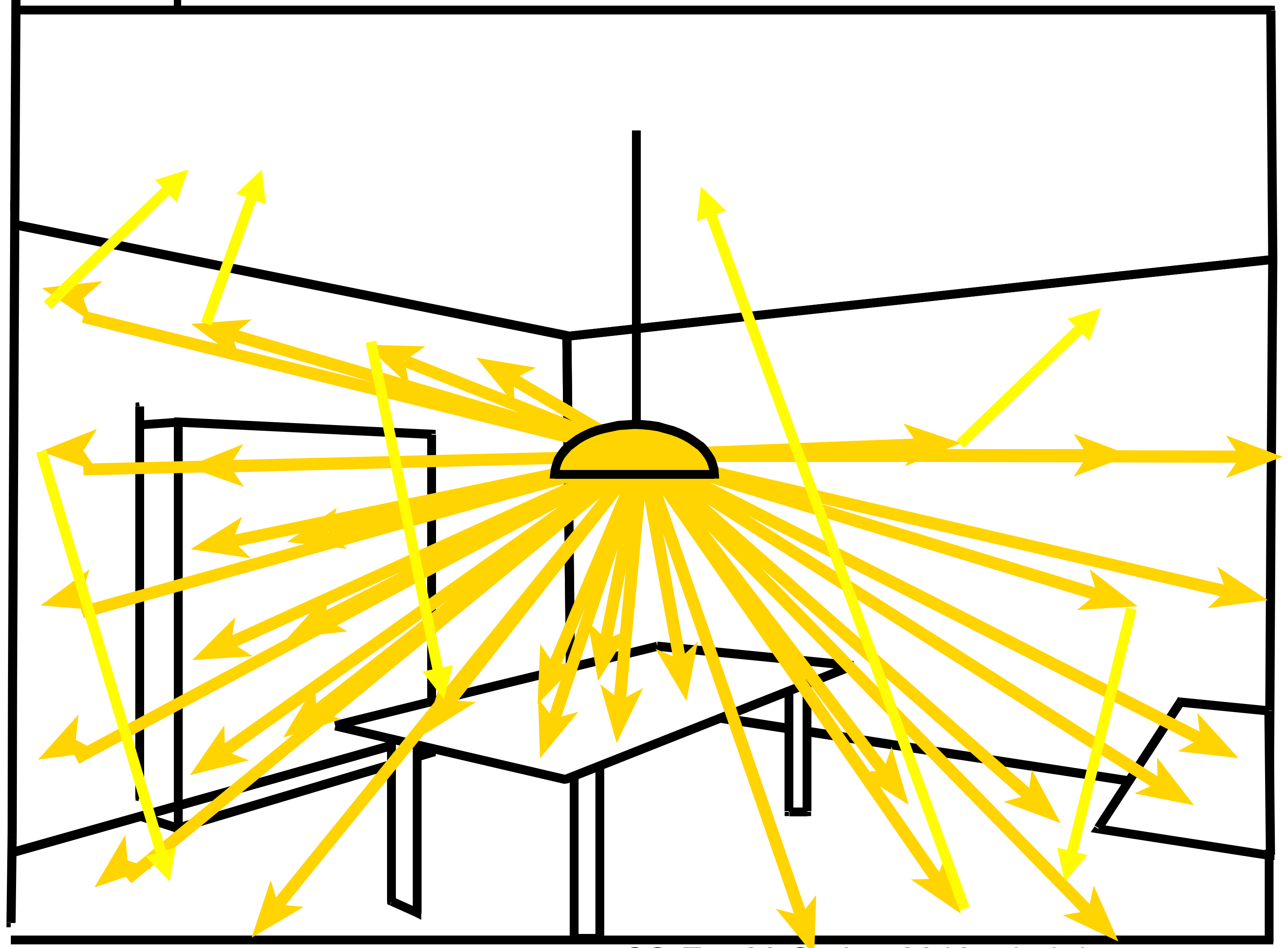


Fig. 3. We develop a novel kernel splatting architecture that maps unordered samples to an image. To support permutation invariance [Aittala and Durand 2018; Zaheer et al. 2017] and process arbitrary number of samples per pixel, we need to process each sample individually while making them aware of the neighborhood patterns. To achieve this, we generate *sample embeddings E* for individual samples and average them into *context features C* for propagating information. We repeat this process for the sample embeddings and context features to better exchange information. Finally, we produce a splatting kernel for each sample, similar to previous kernel-predicting methods [Bako et al. 2017]. This results in an architecture that does not change its outcome based on the order of samples, and is able to process arbitrary number of samples per pixel.

Gharbi et al. SIGGRAPH 2019

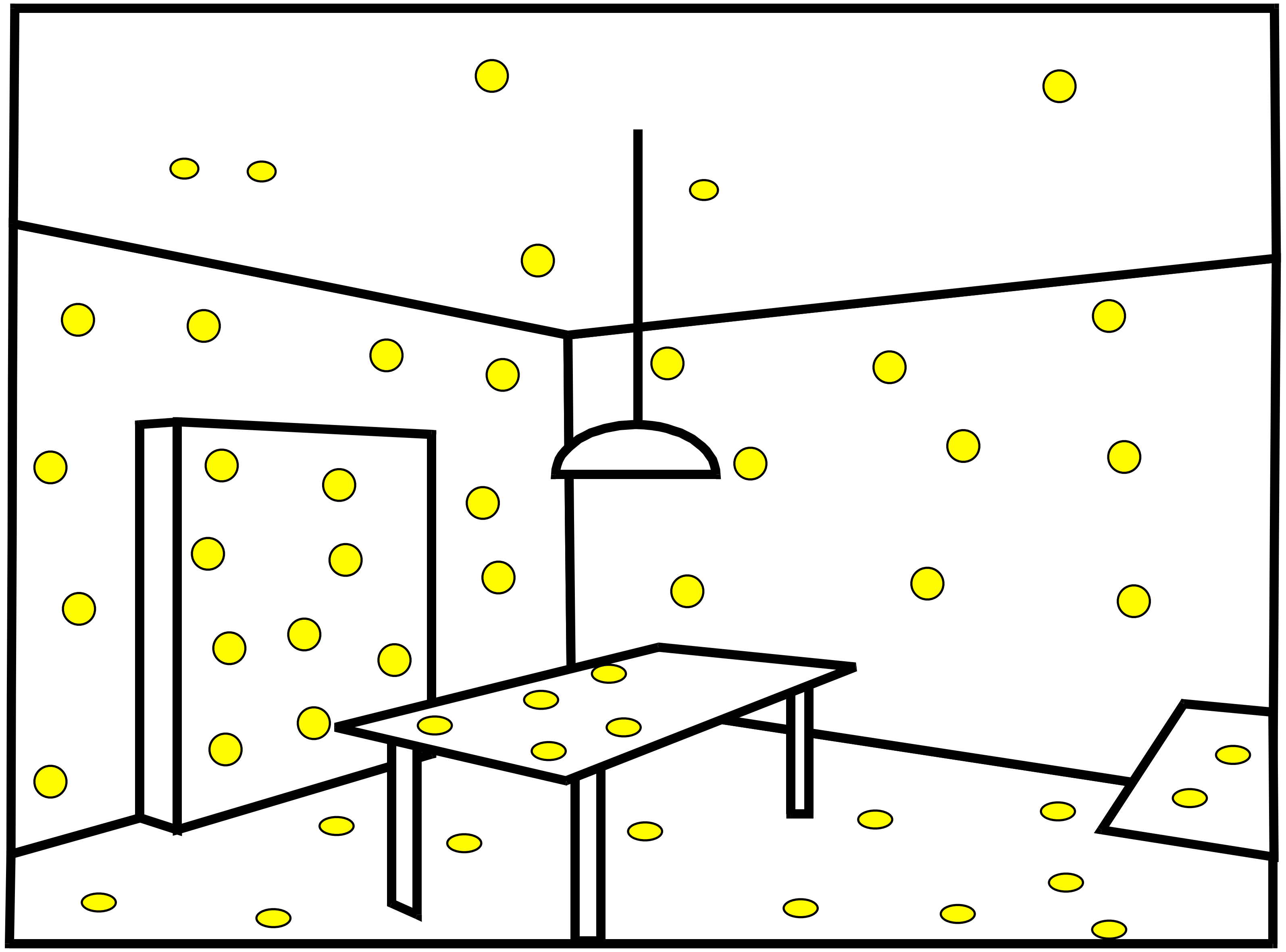
# Photon Mapping

- Preprocess: cast rays from light sources, let them bounce around randomly in the scene
- Store "photons"



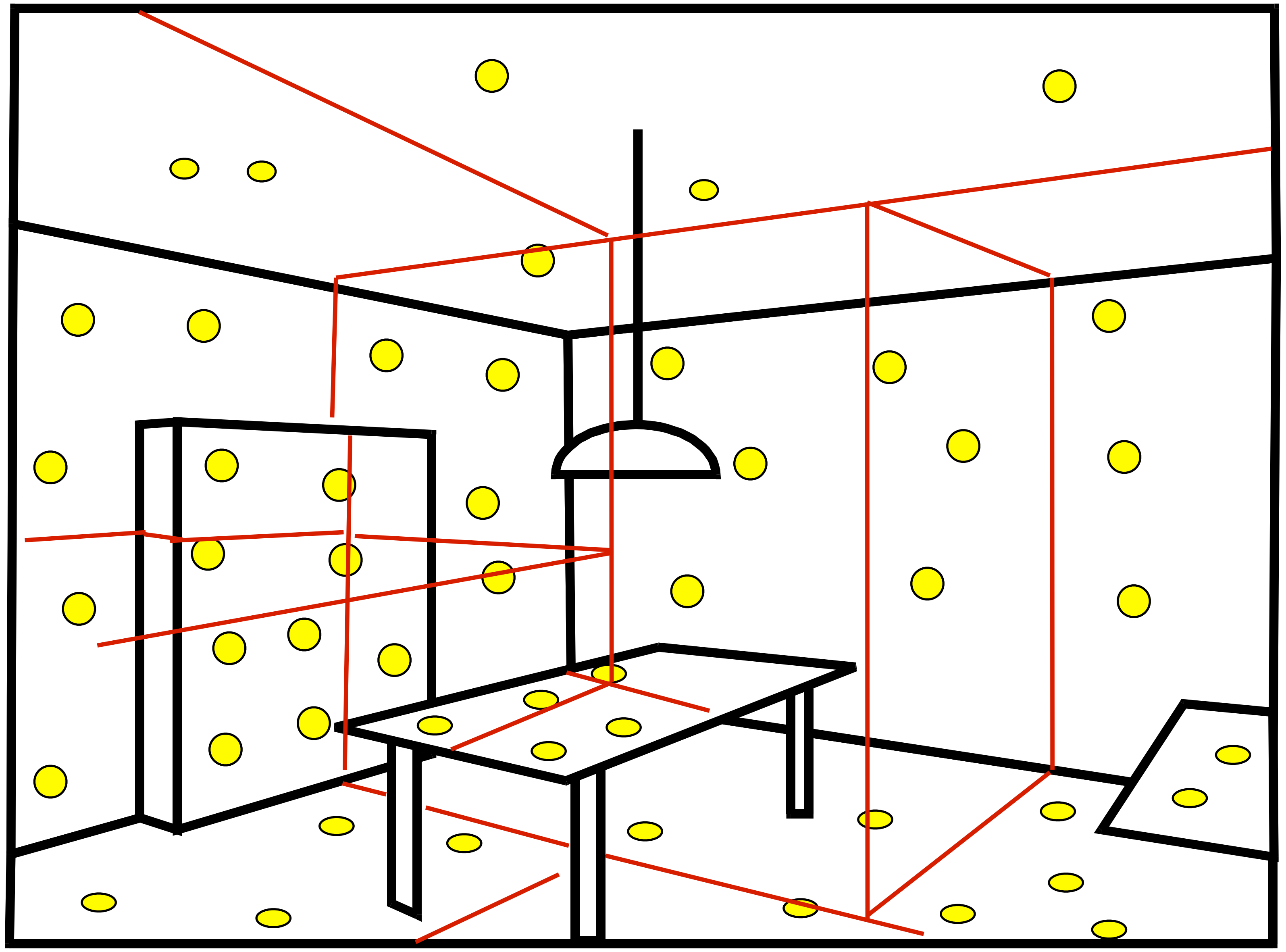
## Photon Mapping

- Preprocess: cast rays from light sources
- Store photons (position + light power + incoming direction)



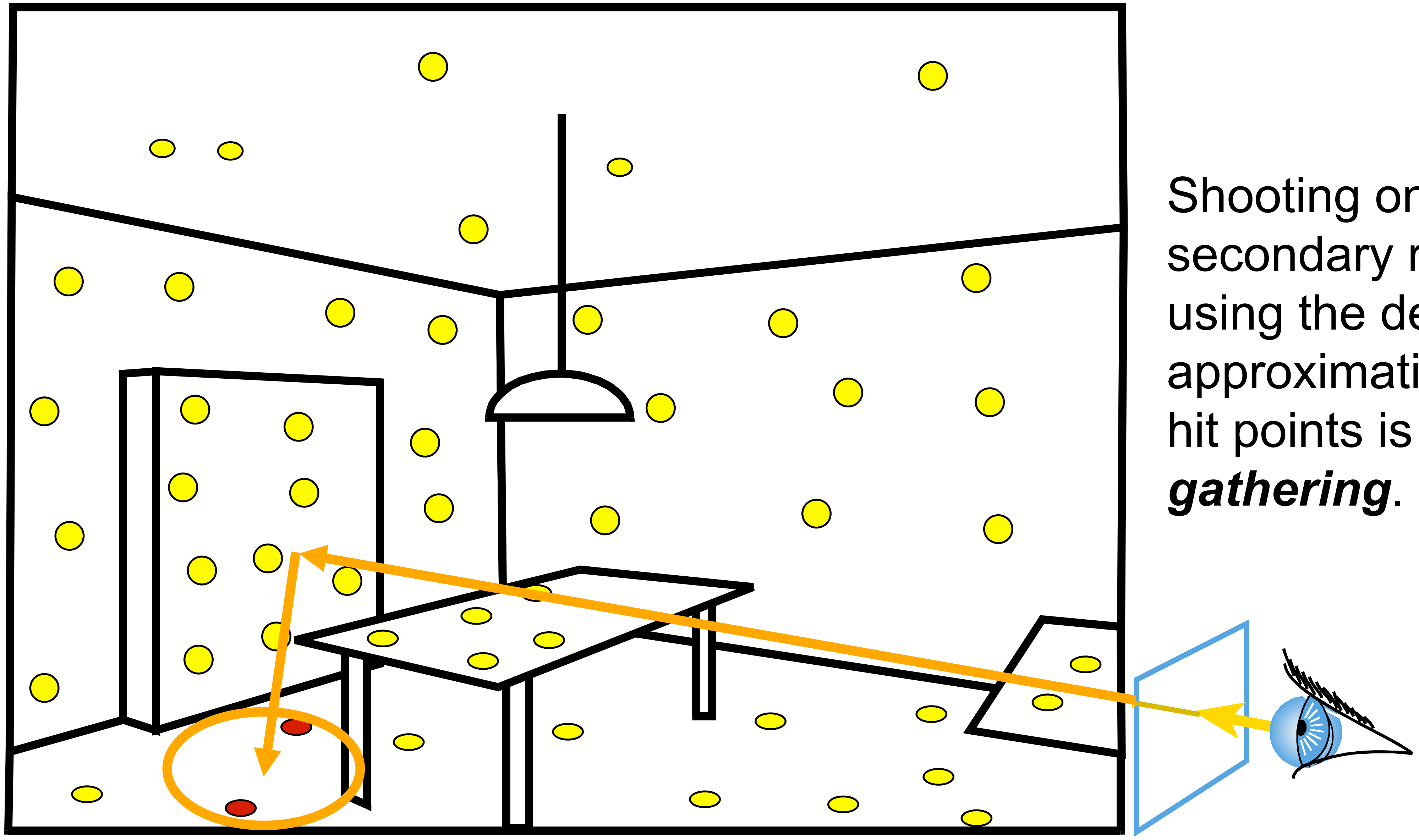
# The Photon Map

- Efficiently store photons for fast access
- Use hierarchical spatial structure (kd-tree)



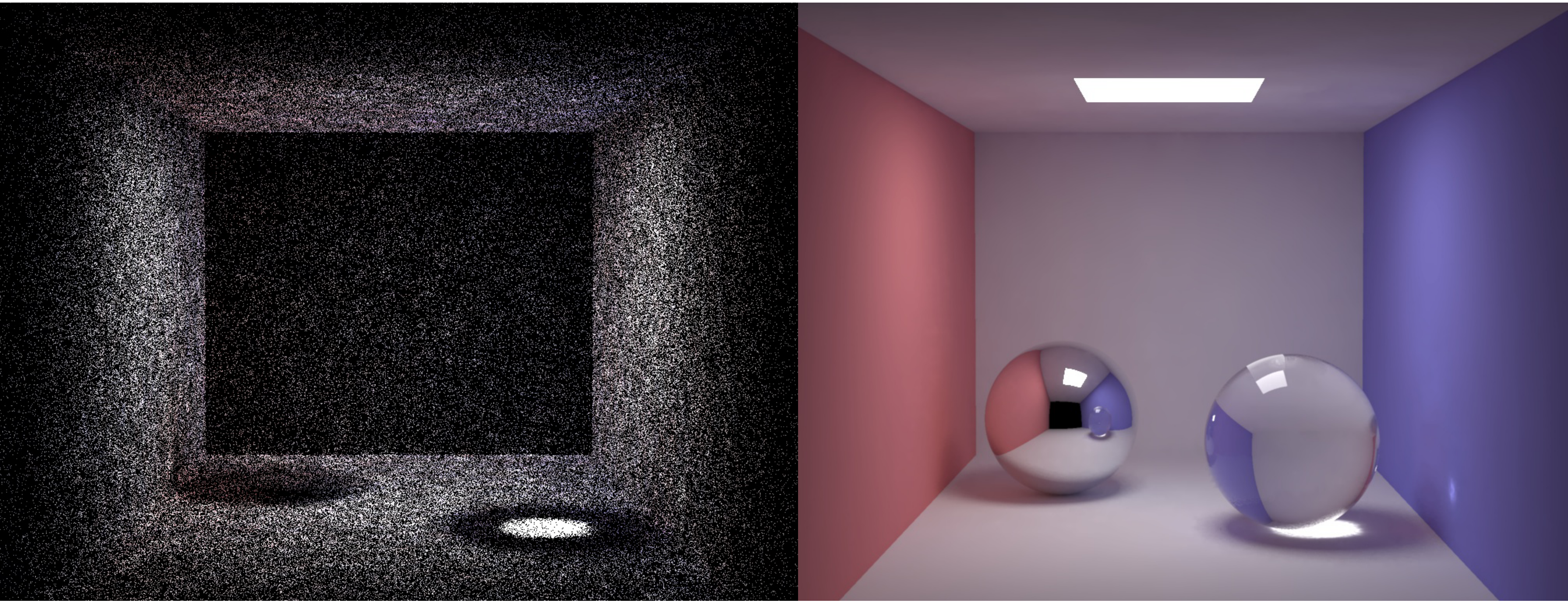
# Photon Mapping - Rendering

- Cast primary rays
- For secondary rays
  - reconstruct irradiance using adjacent stored photon
  - Take the k closest photons
- Combine with irradiance caching and a number of other techniques



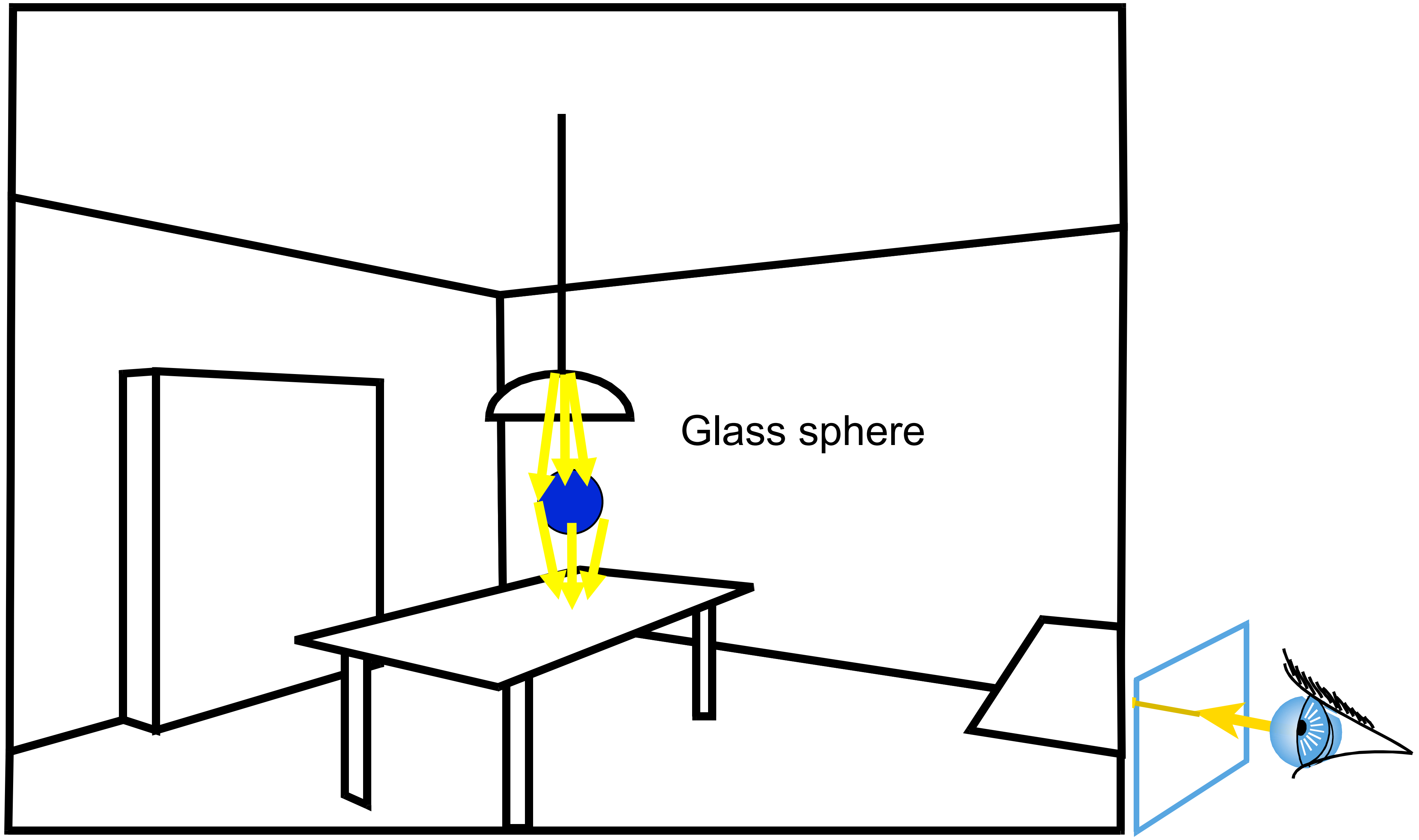
Shooting one bounce of secondary rays and using the density approximation at those hit points is called ***final gathering***.

## Photon Map Results



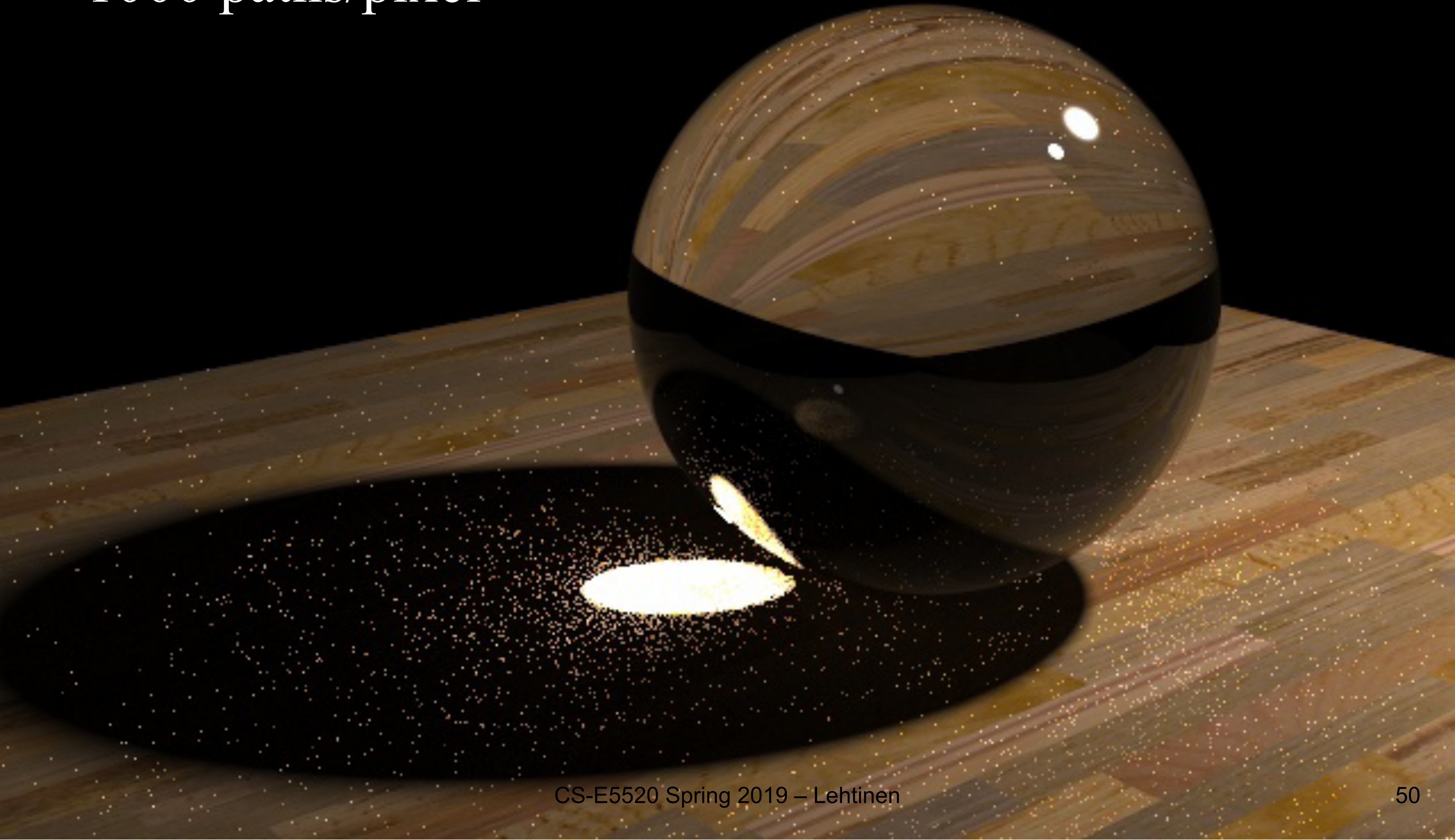
# Photon Mapping - Caustics

- Special photon map for specular reflection and refraction



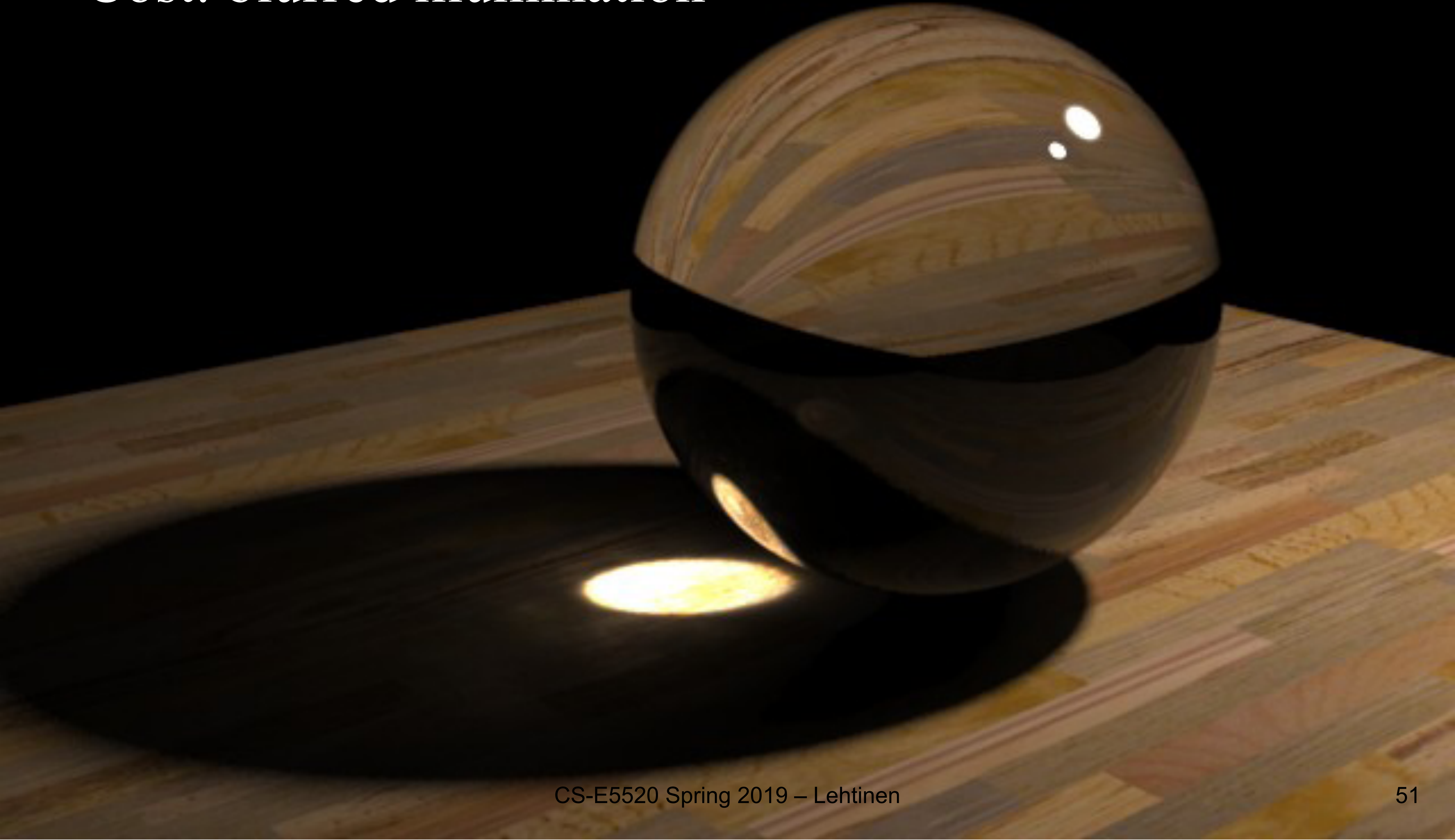
## (Bidir.) Path Tracing is Noisy

- 1000 paths/pixel



# Photon Mapping Does Better

- Cost: blurred illumination



# How Does This Work?

- 1st pass
  - Shoot photons from light source
  - Trace ray, deposit photon on surface
    - Record position, incoming direction, power carried
  - Probabilistically absorb, based on reflectance
    - Russian roulette!
  - If not absorbed: keep going

- In the end, the *density of photons over the surfaces turns out to be proportional to irradiance*
  - Not surprising really is it?

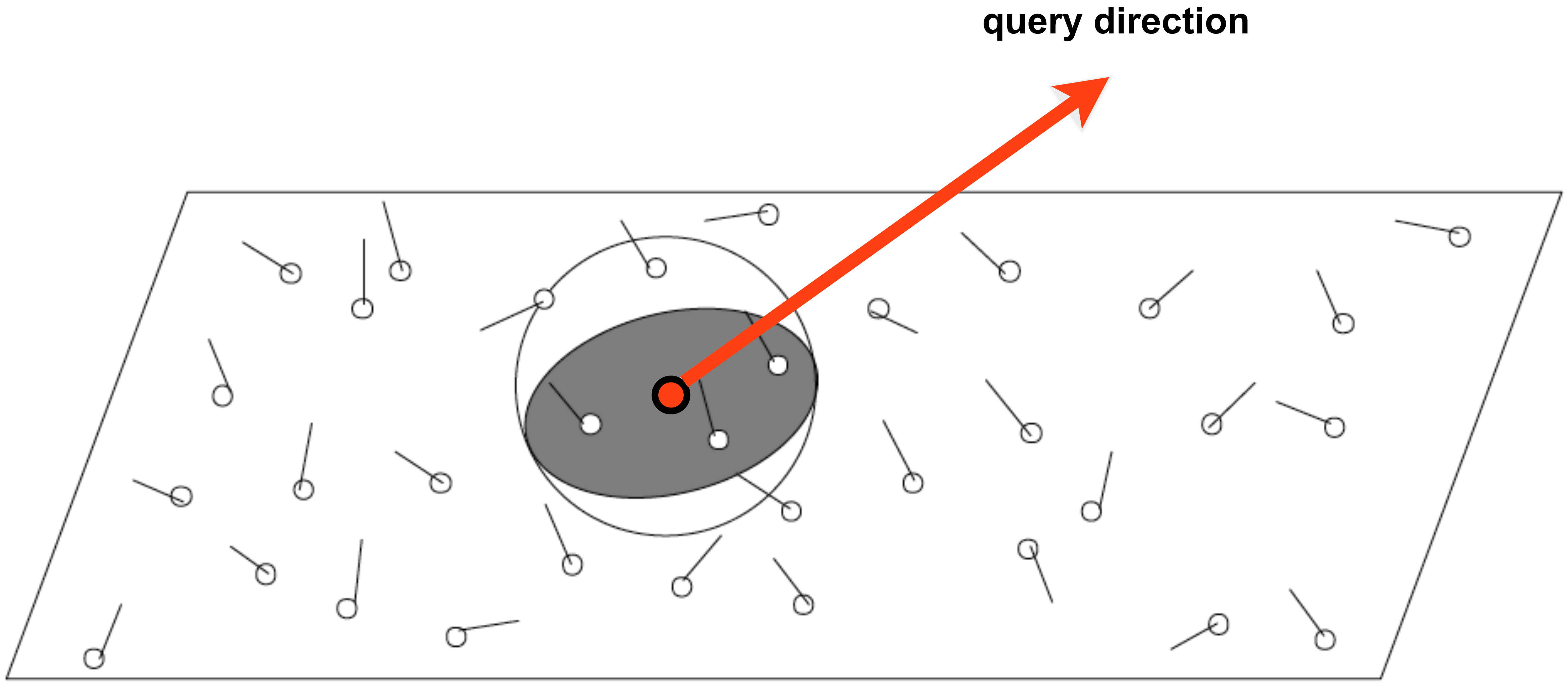
## How Does This Work?

- 2nd pass
  - Estimate irradiance by approximating photon density
- Radiance: $L = \dfrac{\mathrm{d}^2\Phi}{\mathrm{d}A\,\mathrm{d}\omega\ \cos\theta}$
- Reflectance: $L(\omega_o) = \int\limits_\Omega f_r(\omega \to \omega_o) L(\omega) \cos\theta\,\mathrm{d}\omega$

# How Does This Work?

- 2nd pass
  - Estimate irradiance by approximating photon energy density
- Radiance: $L = \frac{\mathrm{d}^2\Phi}{\mathrm{d}A \cancel{\mathrm{d}\omega} \cancel{\cos} \theta}$
- Reflectance: $L(\omega_o) = \int_\Omega f_r(\omega \rightarrow \omega_o) L(\omega) \cancel{\cos\theta} \cancel{\mathrm{d}\omega}$
- Substitute: "$L(\omega_o) = \int_\Omega f_r(\omega \rightarrow \omega_o) \frac{\mathrm{d}^2\Phi}{\mathrm{d}A}$"

# Example



● **query location**

adapted from SIGGRAPH course notes

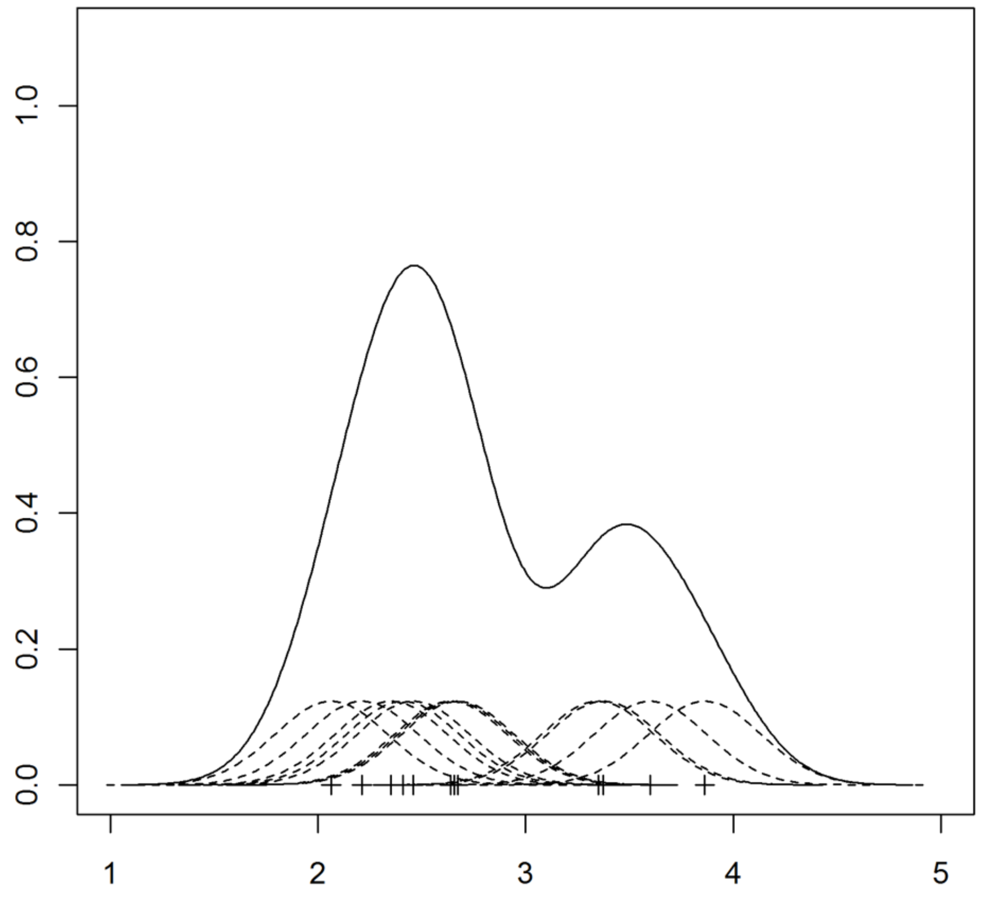
# k-nearest Neighbors Search

$$L_o(\omega_o) \approx \sum_{i=1}^{k} f_r(\omega \to \omega_o) \frac{\Delta\Phi_n(x,\omega)}{\Delta A}$$

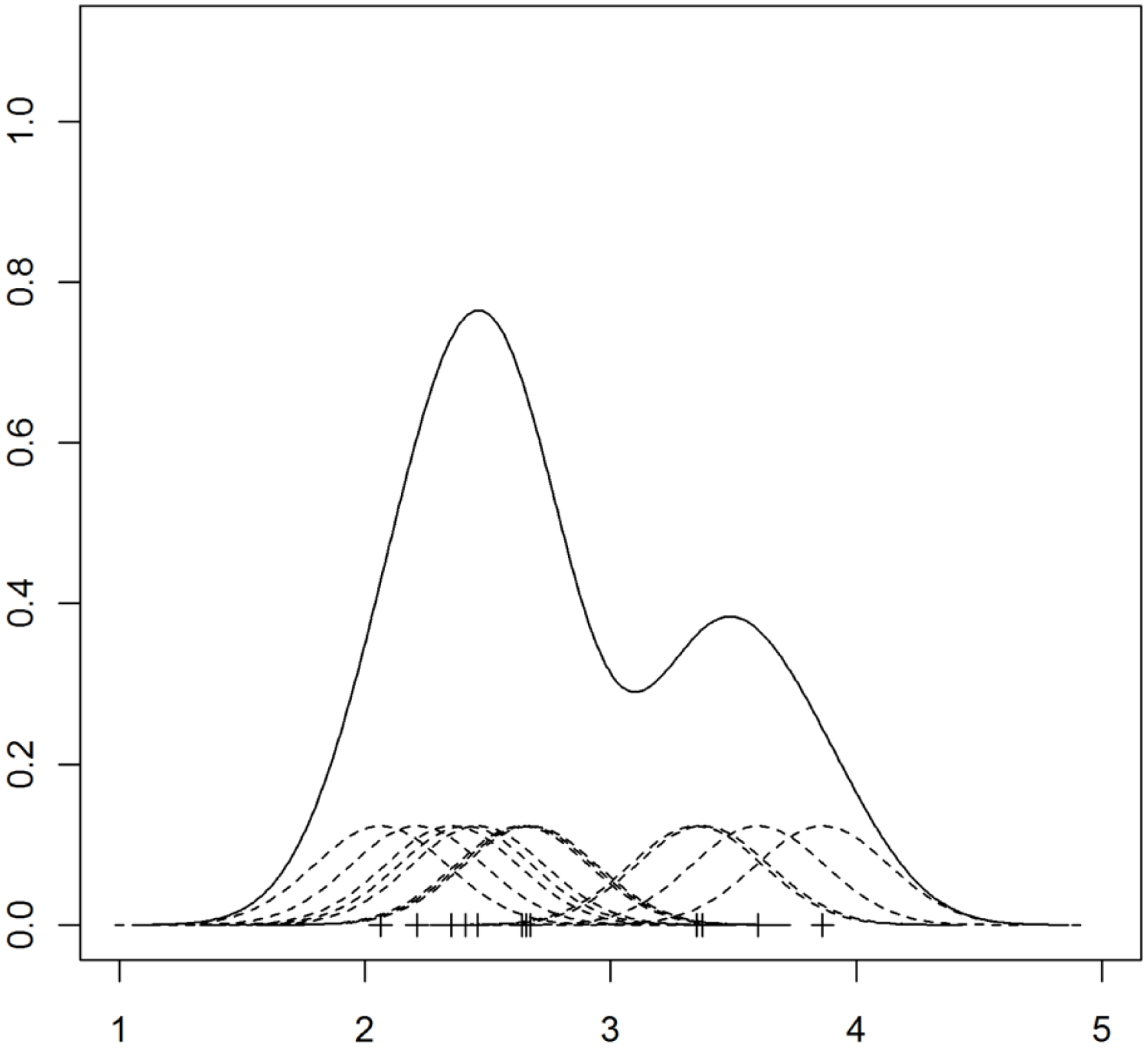
- *k*-nearest neighbors density estimation
  - find the *k* nearest photons (easy using a BVH)
  - estimate how large the area is they fall upon
    - E.g. assume they fall upon a disk whose radius is the distance of the *k*th nearest photon from the query point
  - sum up their power
  - divide by area

# Another Way: Kernel Density Estimation

- k-nearest neighbors is not the only method for estimating density from point samples
- Slapping a smooth kernel on each sample and summing the values is another approach
  - can be used with photon mapping too

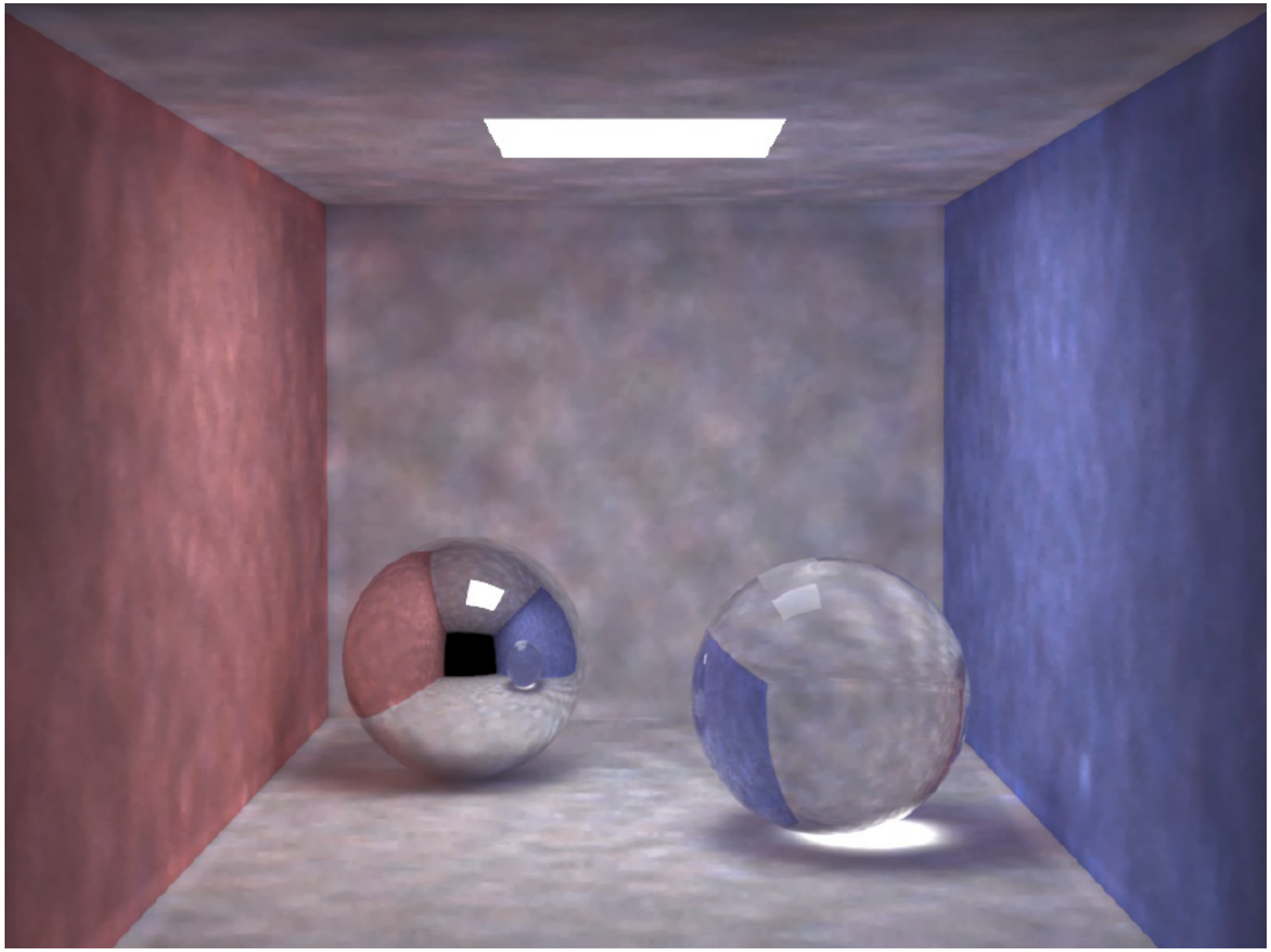


# Kernel Density Estimation



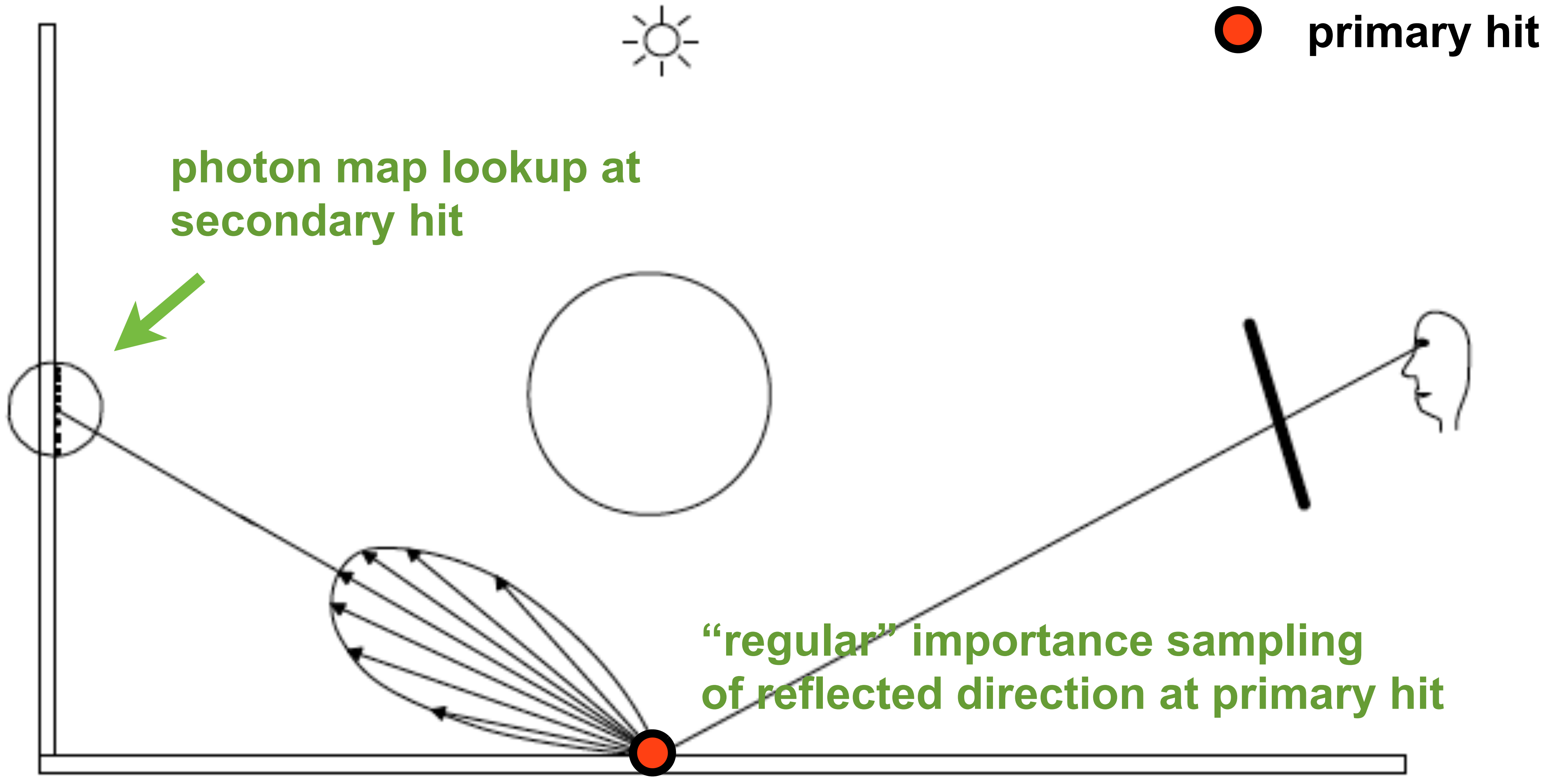
## Issues

- Need lots of photons, otherwise estimate has low-frequency noise



# "Fix": Final Gathering + IC

- Trace secondary rays as usual, use photon map for approximating lighting incident to primary hits
  - Of course, more expensive, but hides artifacts
  - Combine with irradiance cache and usual direct light sampling


primary hit
photon map lookup at secondary hit
"regular" importance sampling of reflected direction at primary hit

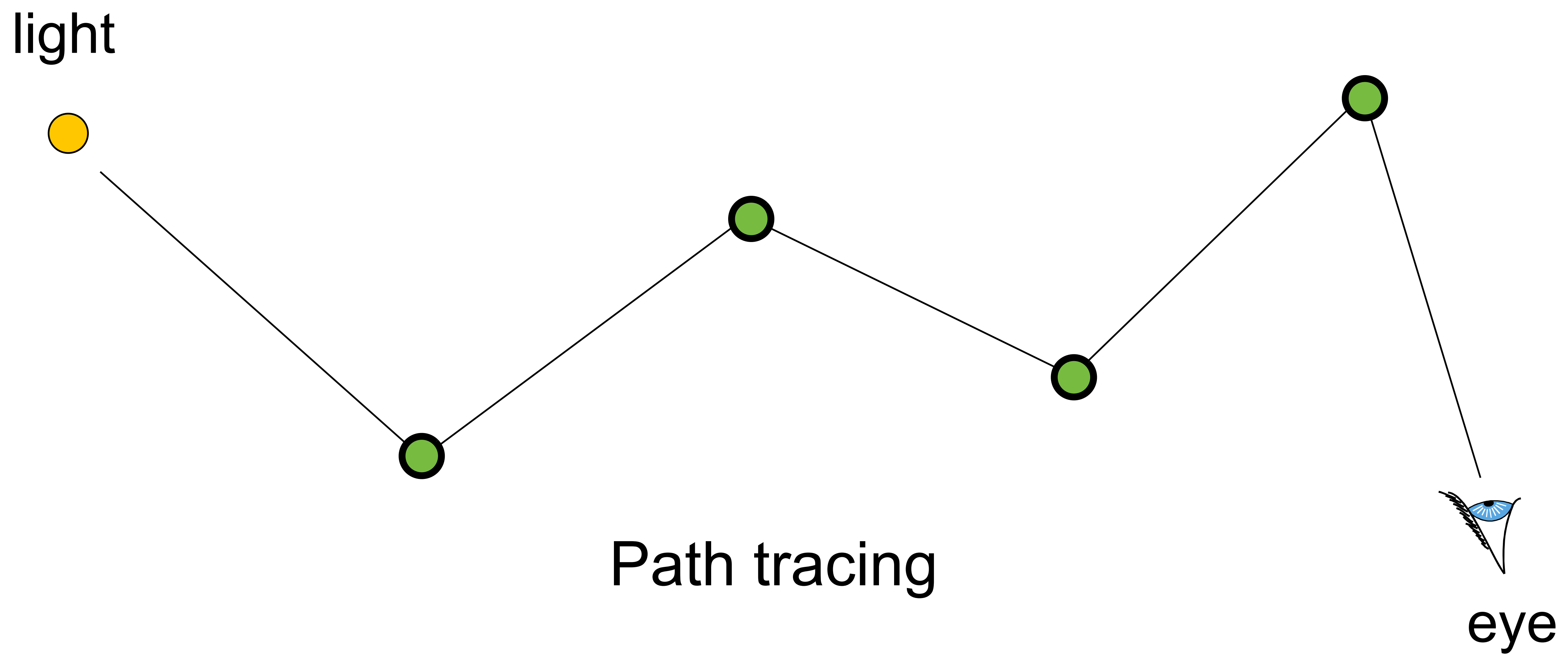
## Example

- Henrik Wann Jensen "The Light of Mies van der Rohe"
  - Model by Stephen Duck

- Further reading on Photon Mapping: SIGGRAPH 2007 course notes

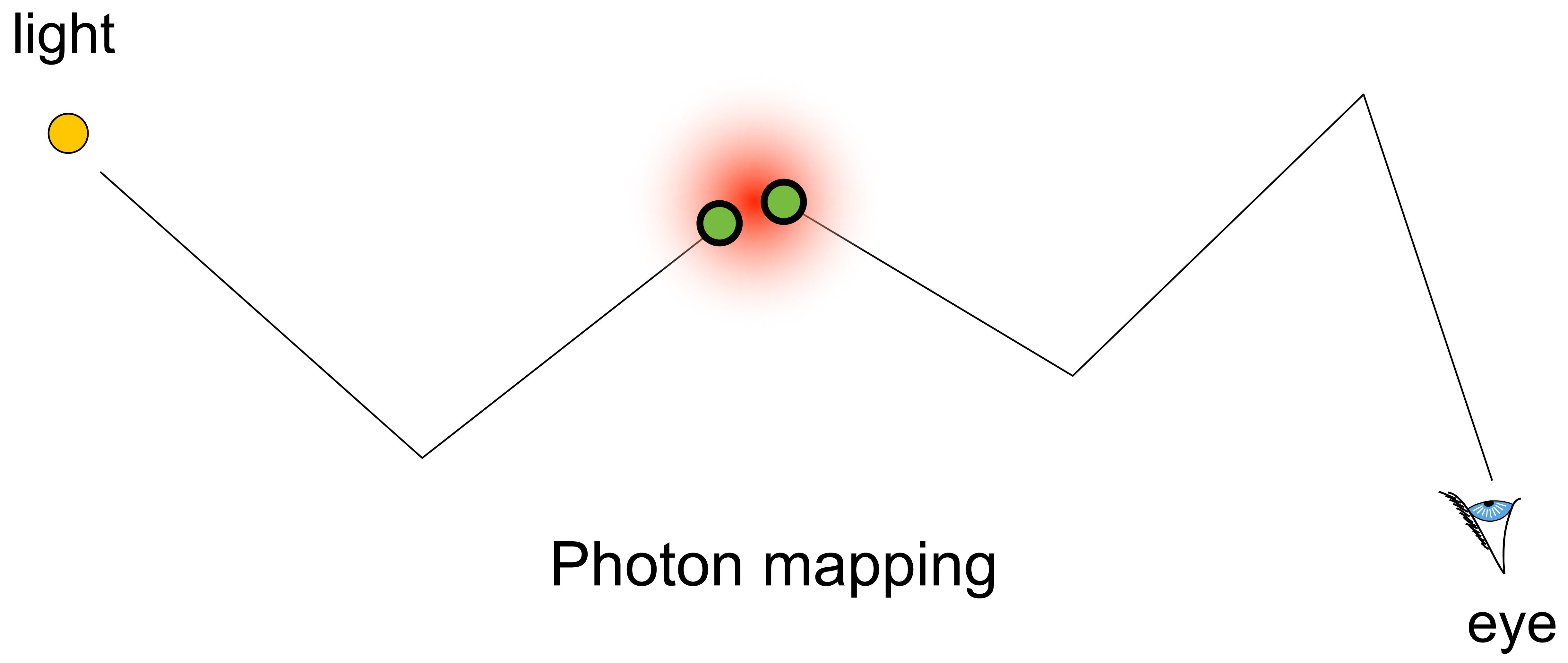
# Photon Mapping vs. Path Tracing

- What are we doing when we perform density estimation?



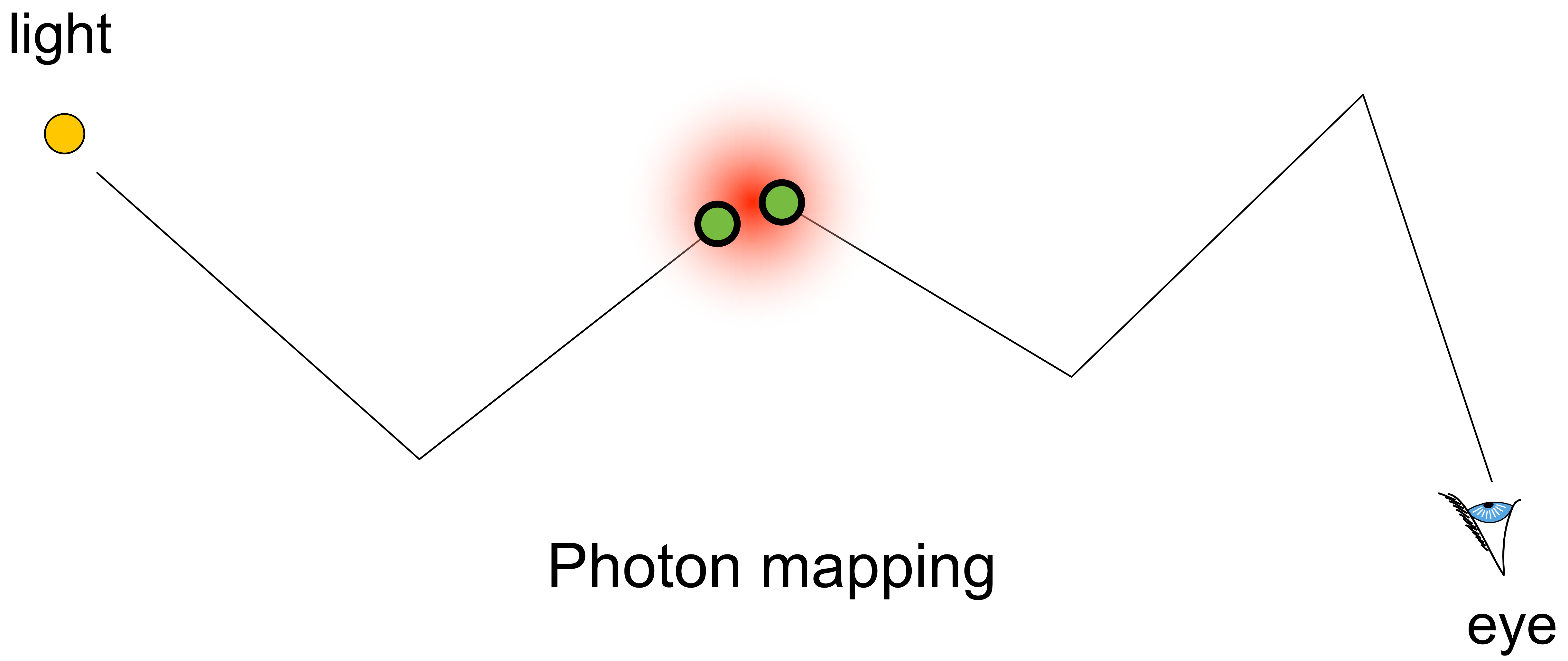
# Photon Mapping vs. Path Tracing

- We allow paths that do not actually connect, only come close



# Photon Mapping vs. Path Tracing

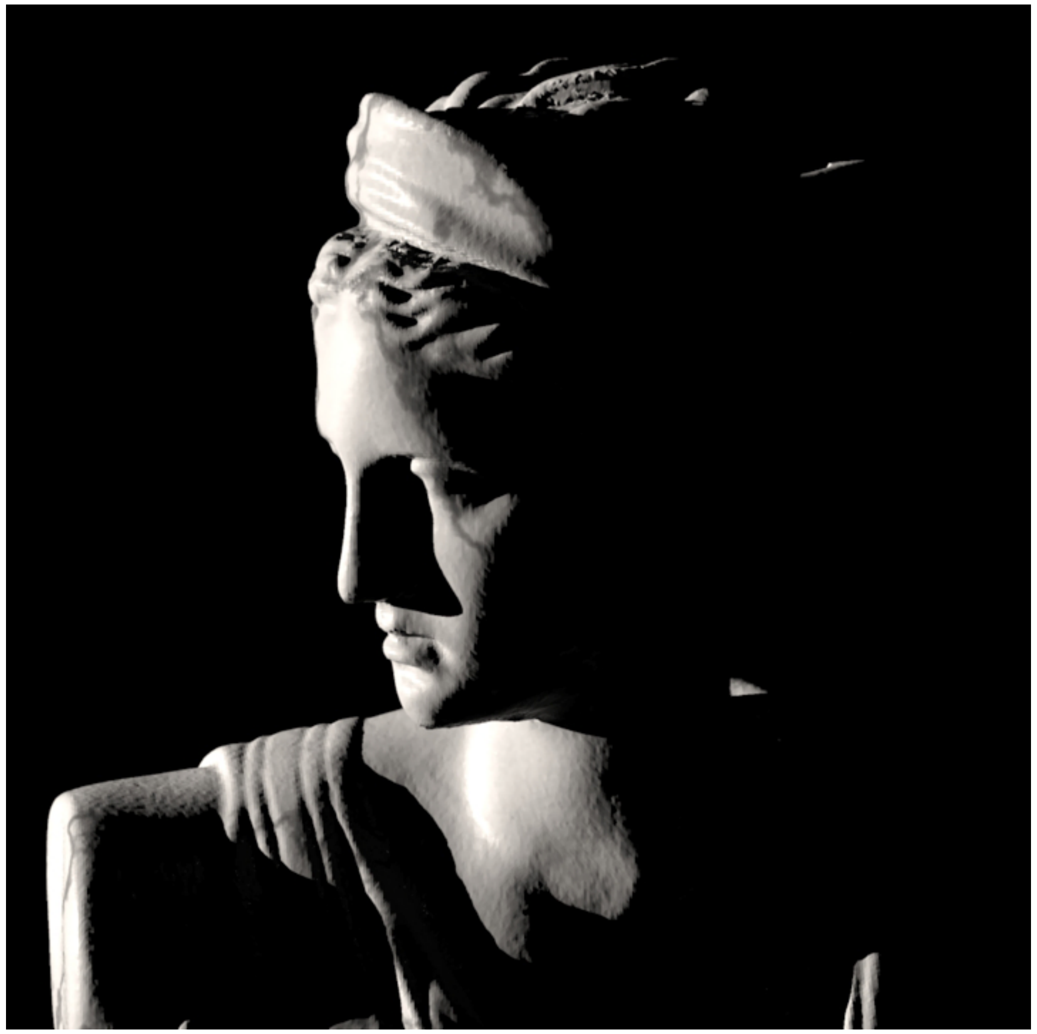
- This operation has recently been cast in a path space formulation in two independent works (Hachisuka et al. and Georgiev et al.) – **highly recommended reading!**

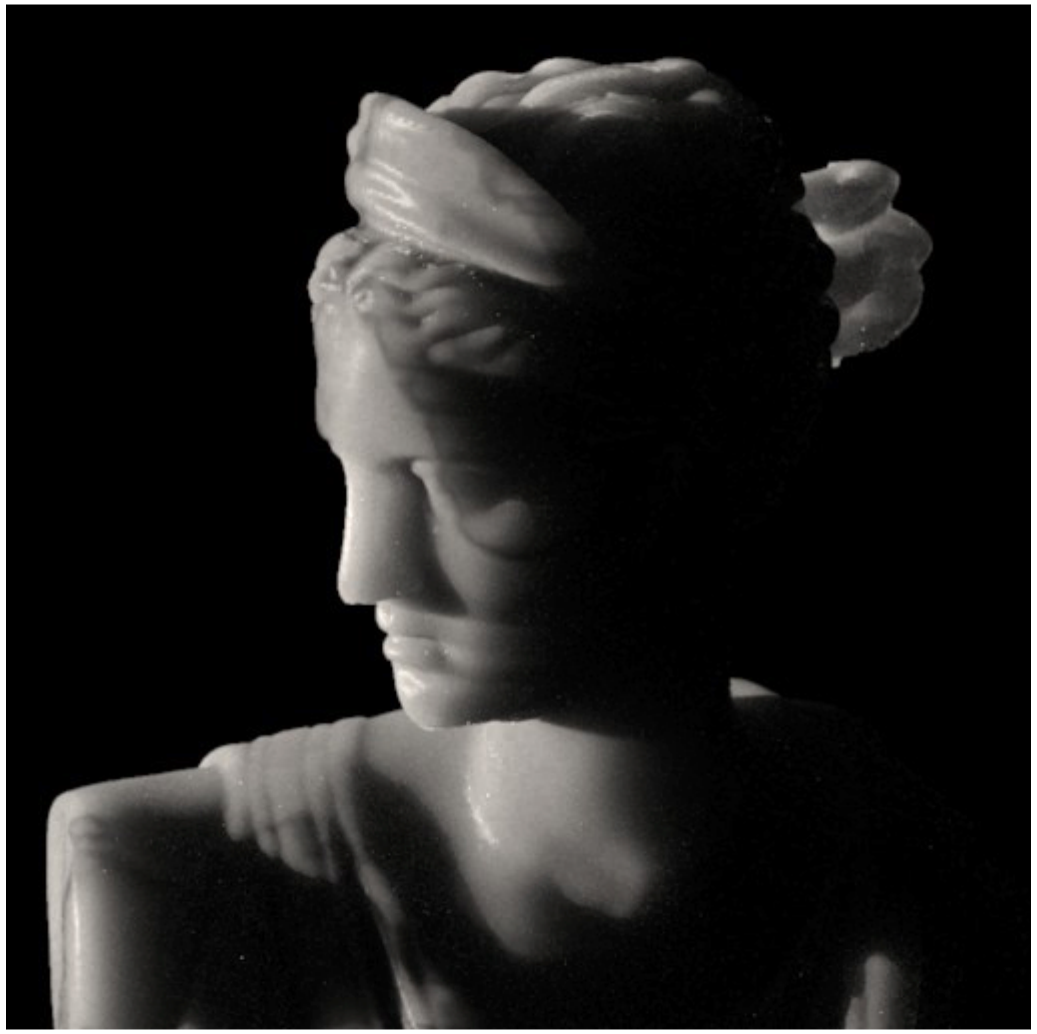


# More Global Illumination Coolness

- Many materials exhibit *subsurface scattering*
  - Light doesn’t just reflect off the surface
  - Light enters, scatters around, and exits at another point
  - Examples: Skin, marble, milk

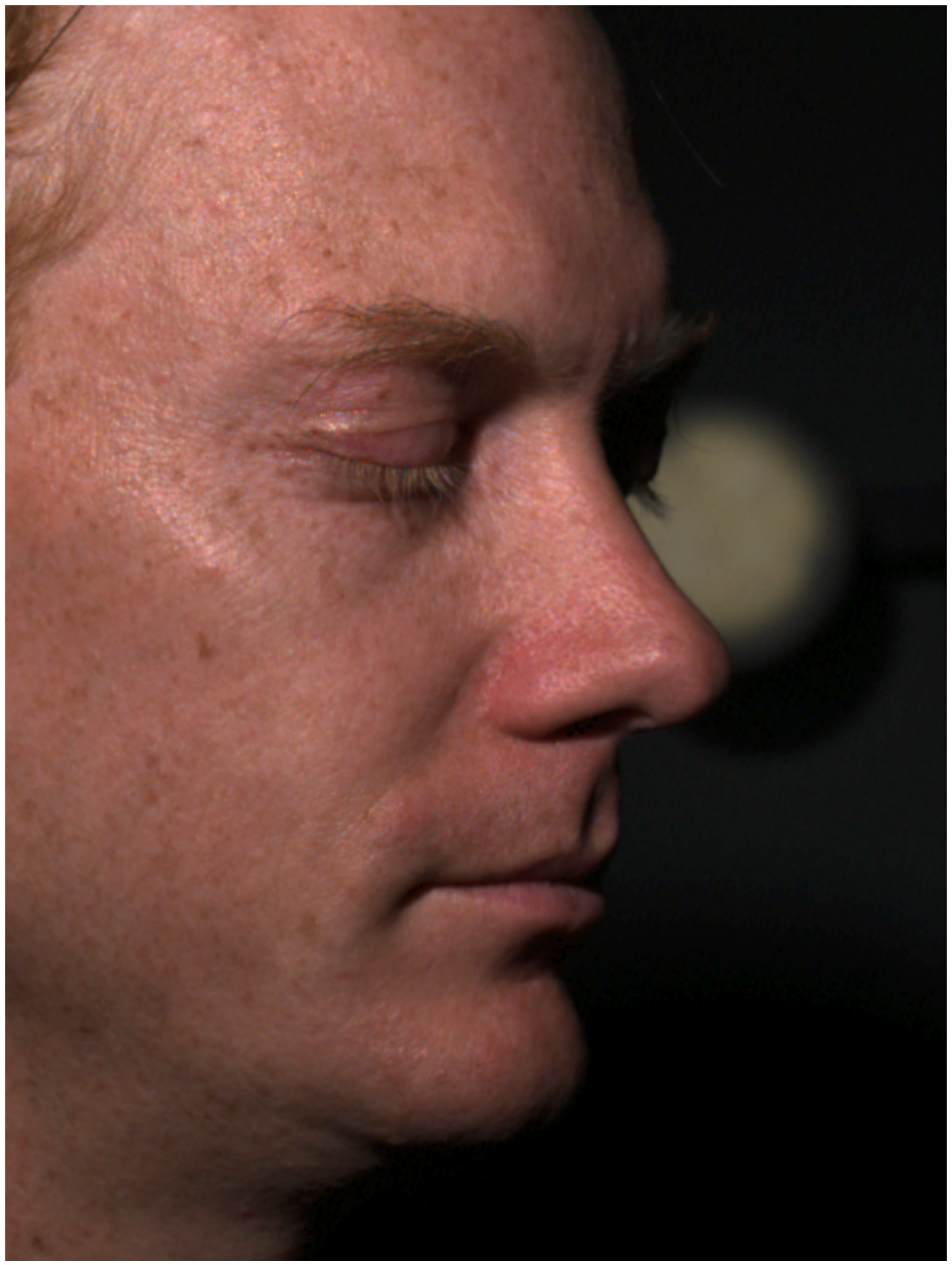
Images: Jensen et al.

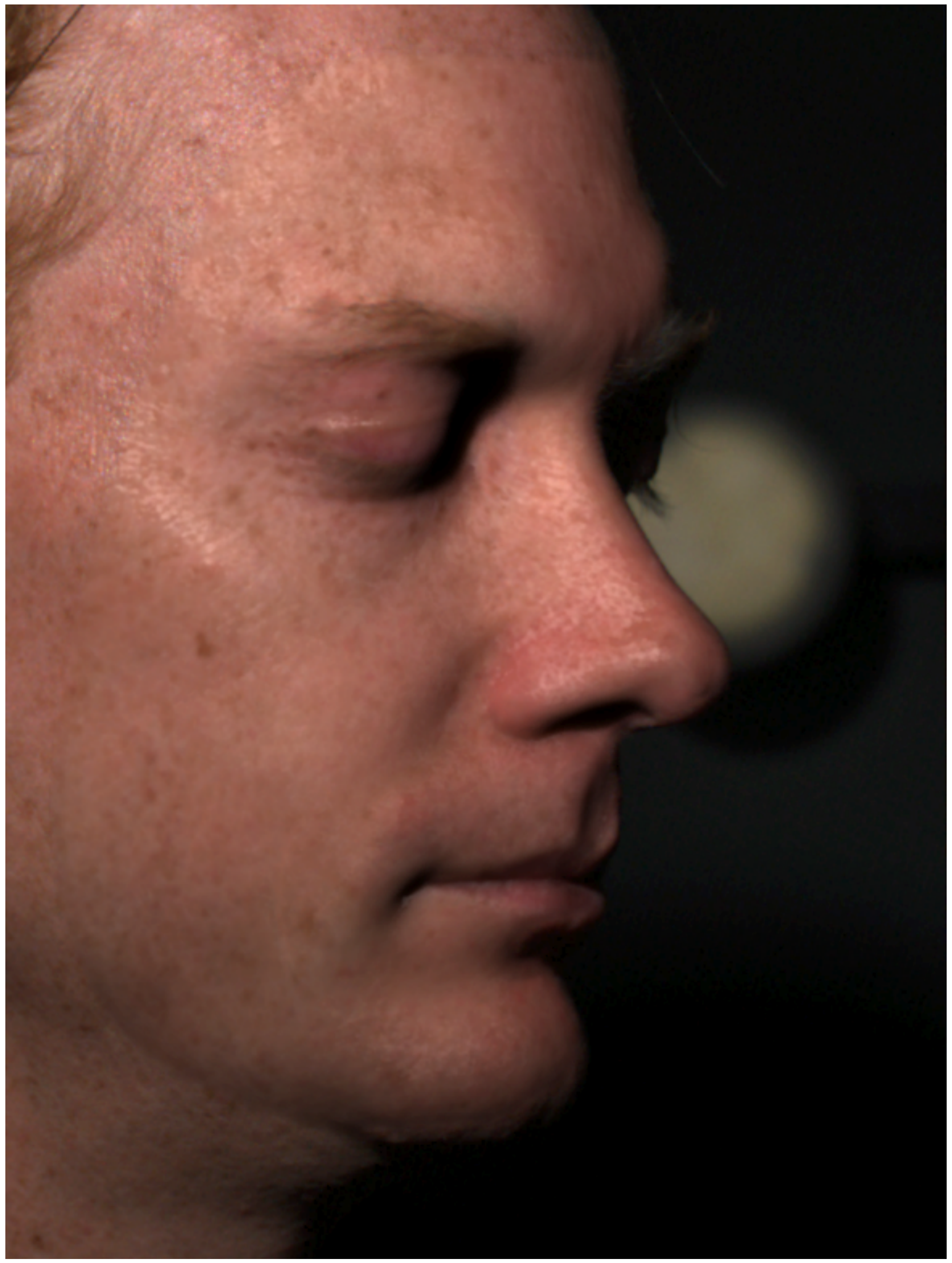




# More Subsurface Scattering

Weyrich et al. 2006


Photograph

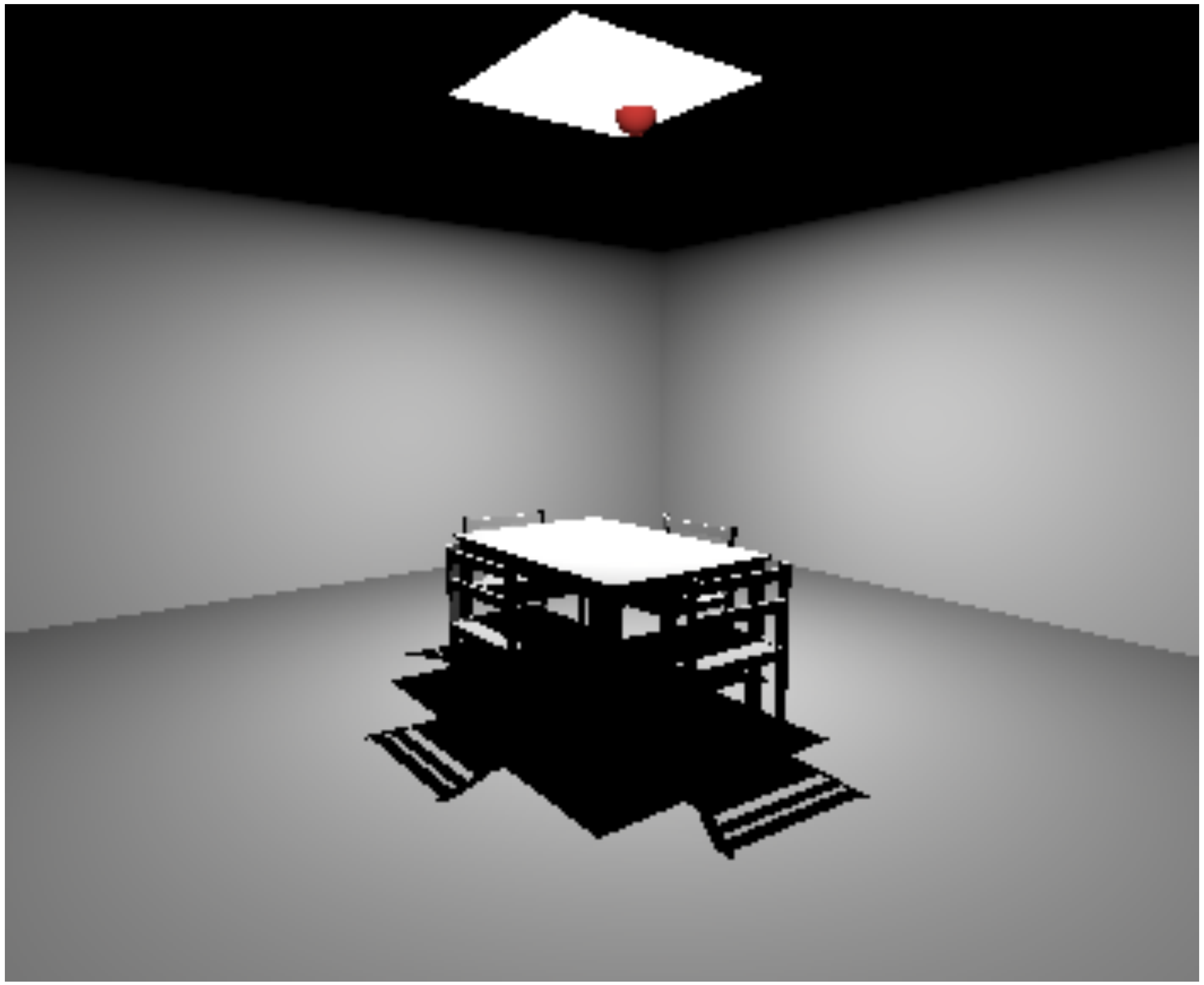

Rendering

# Instant Radiosity

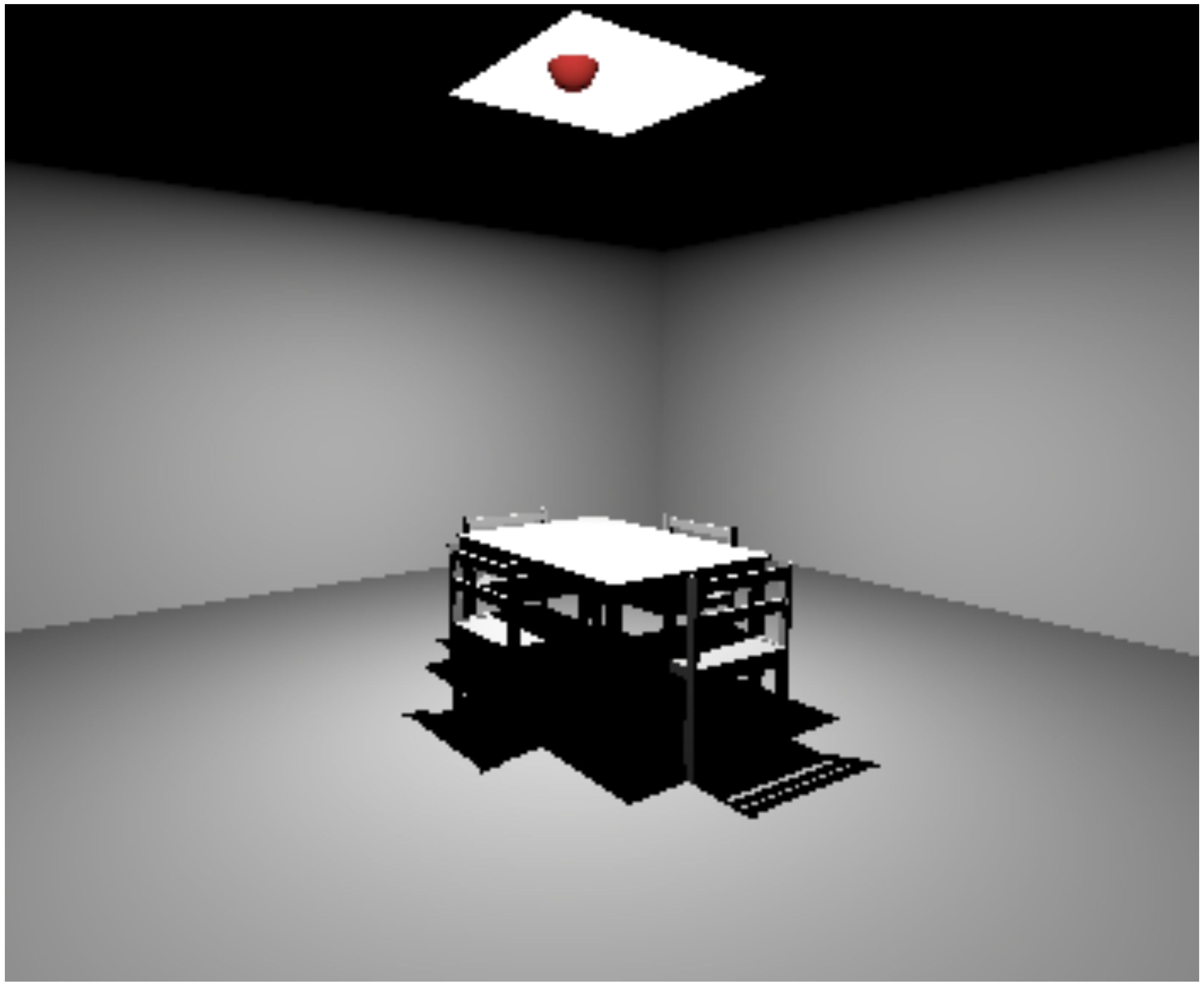
- Almost like photon mapping...
- ...except that we use the “photons” as light sources and render them into the pictures using shadow mapping
  - Secondary lights called “Virtual Point Lights” (VPL)
  - Lots of research since Alex Keller’s SIGGRAPH 97 paper
  - Including some of our own

- You know this already!

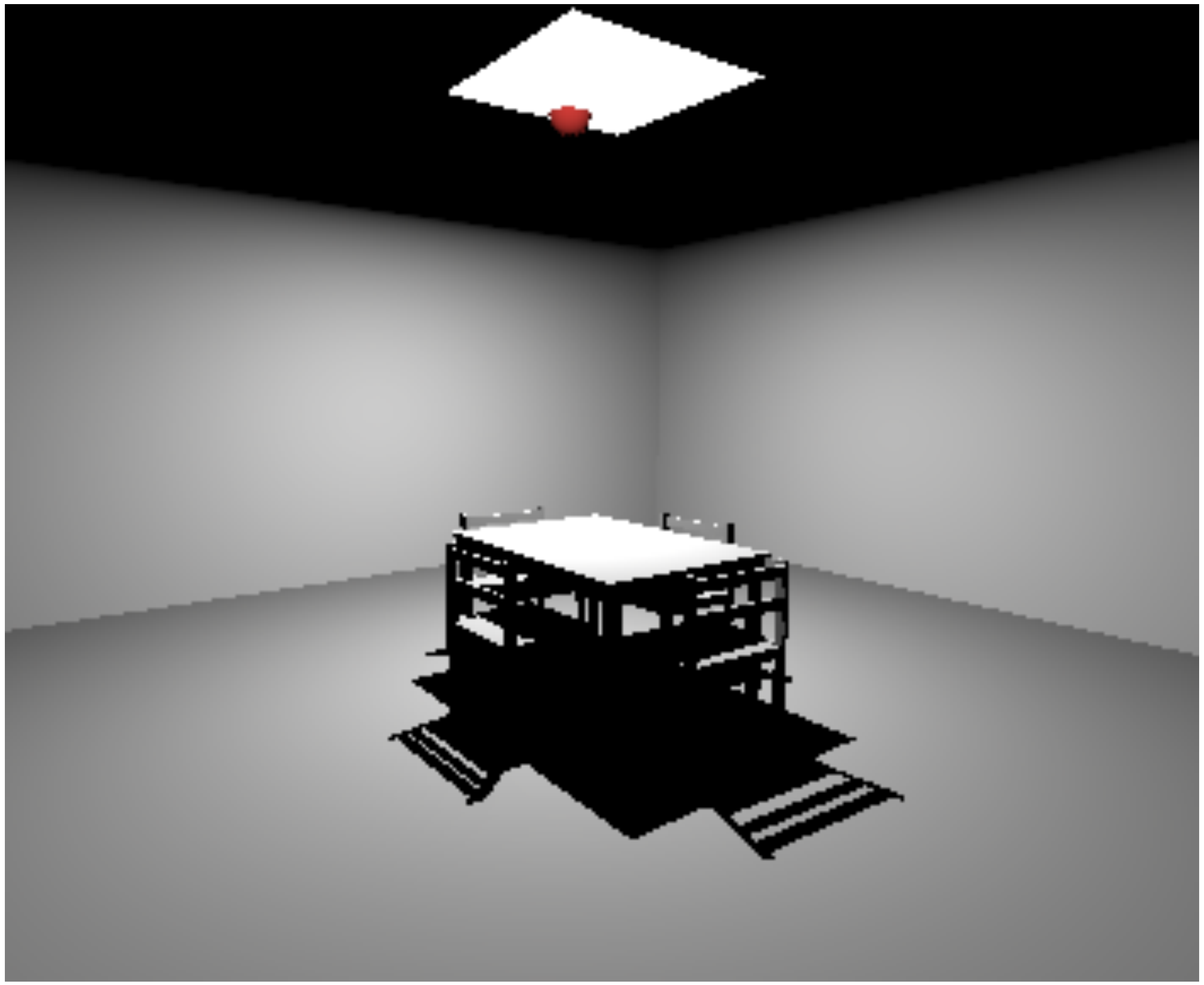
## Direct Light



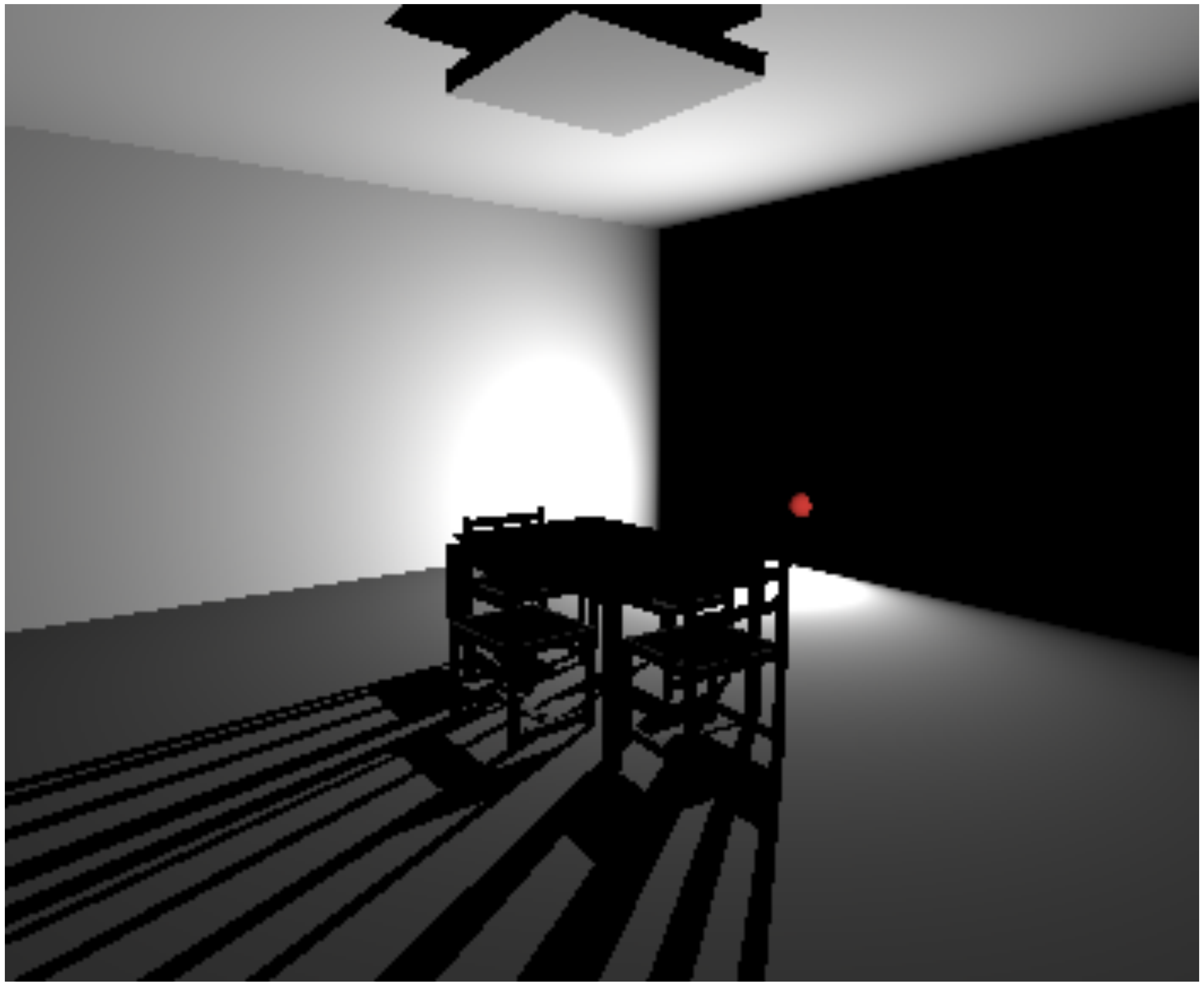
# Direct Light



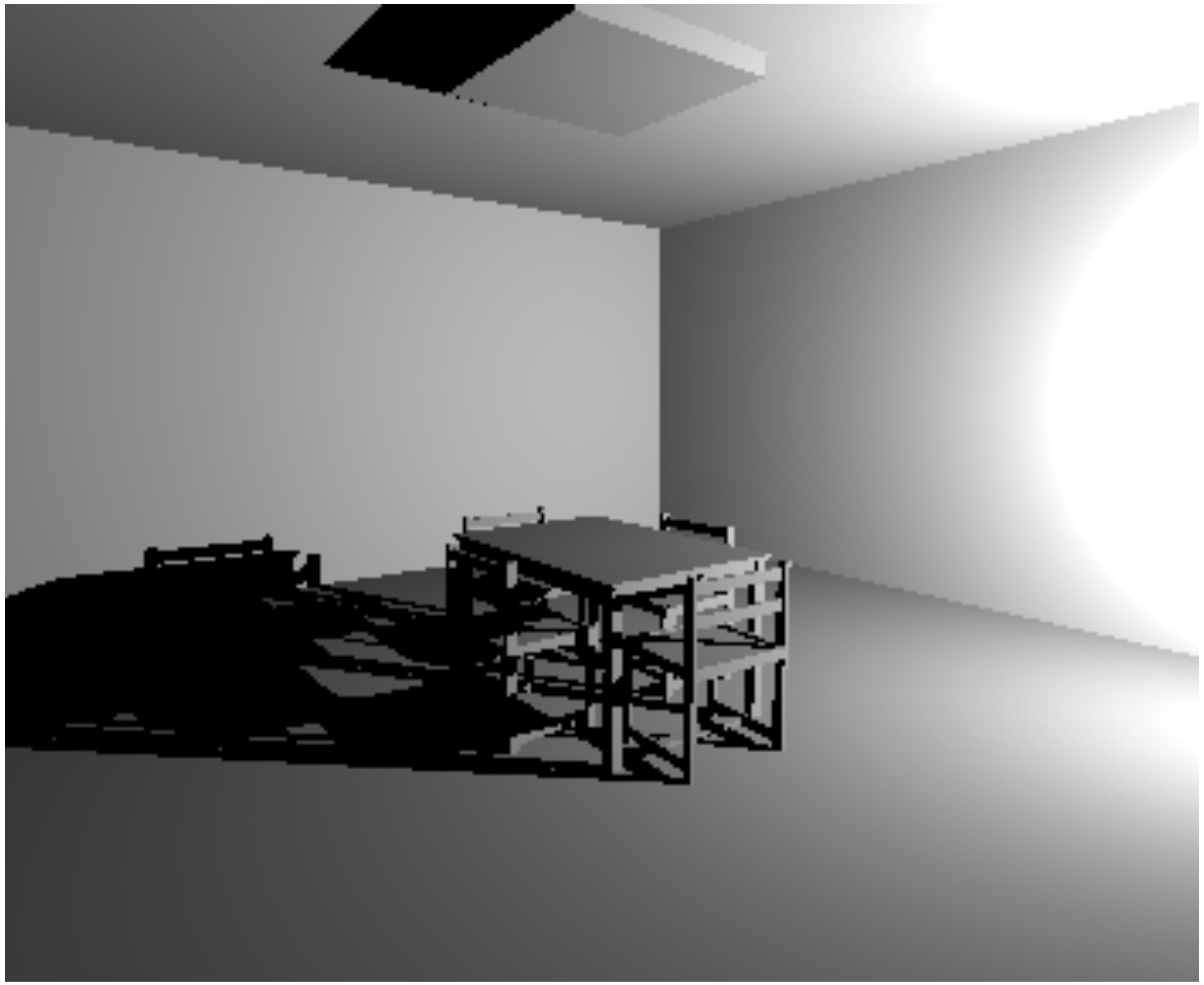
# Direct Light



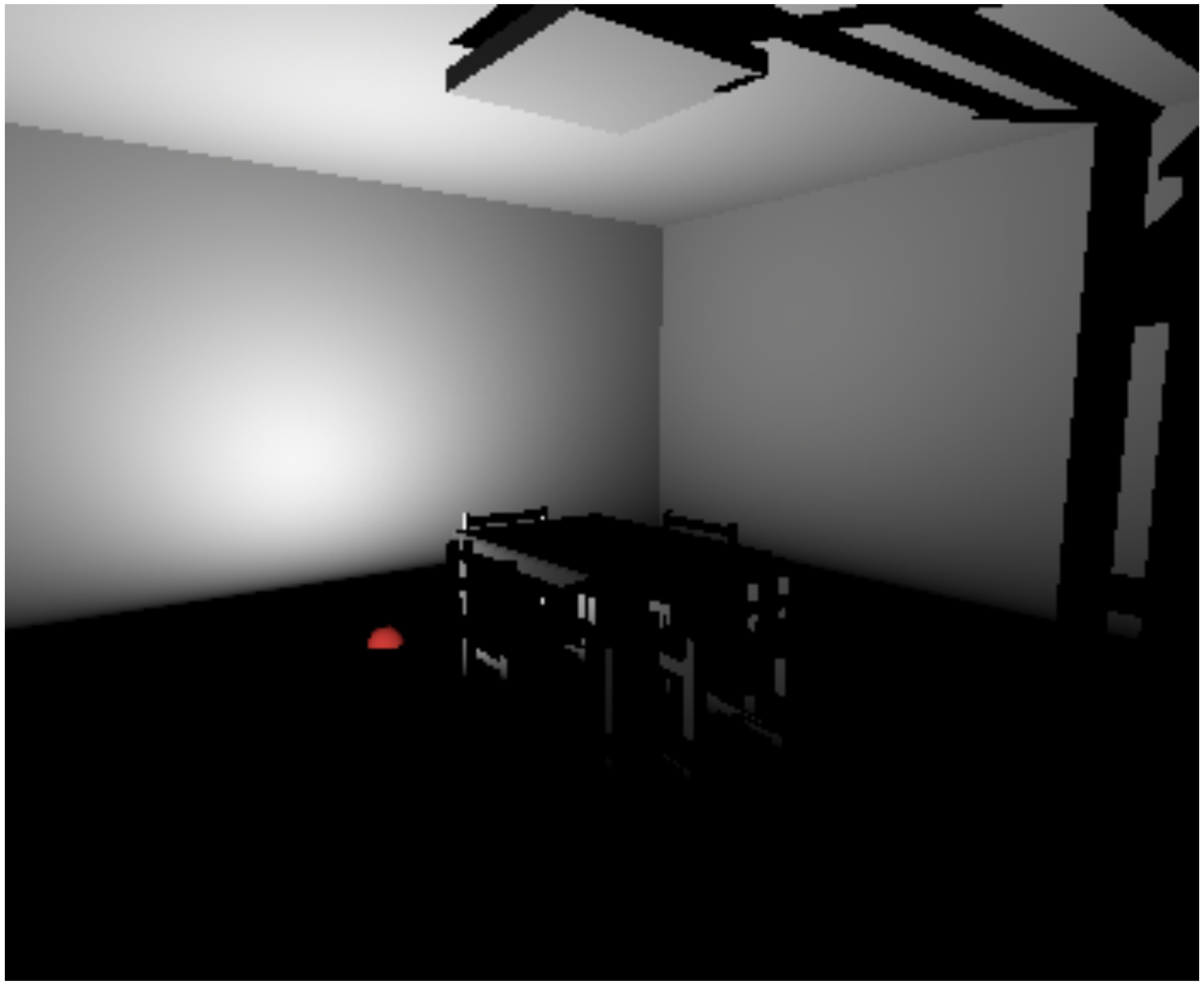
# 1st Indirect Bounce



## 1st Indirect Bounce

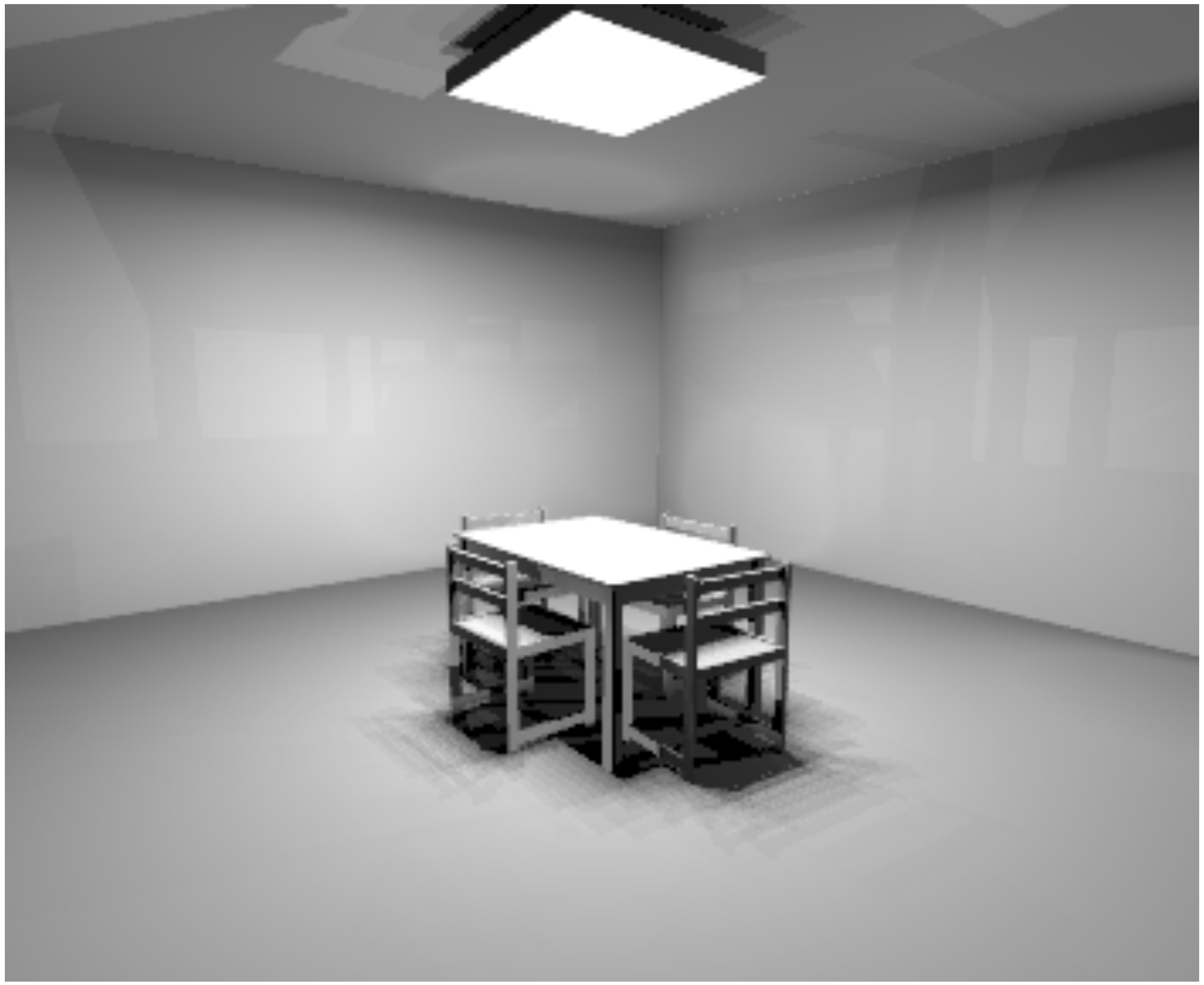


# 1st Indirect Bounce

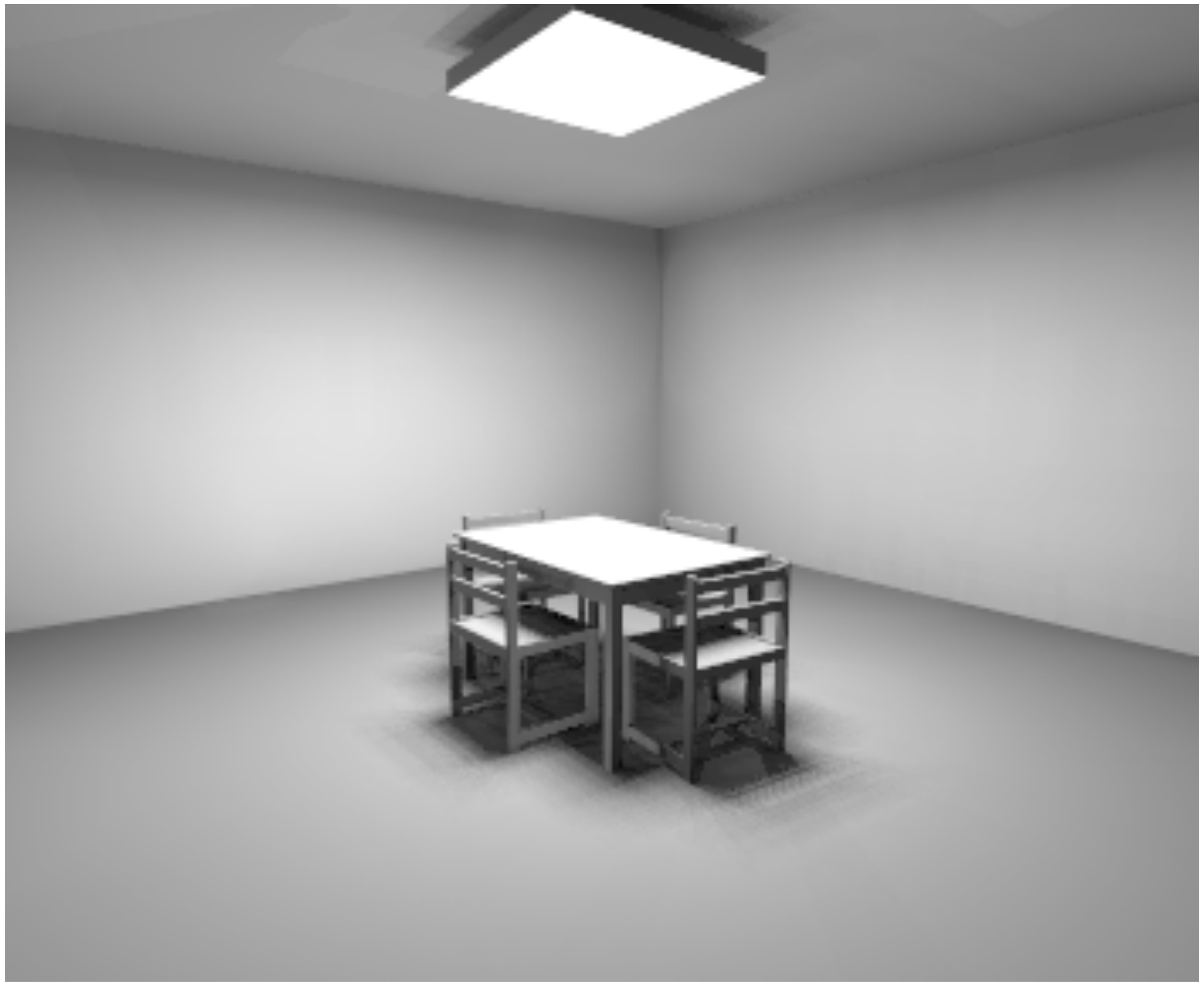


...and so on

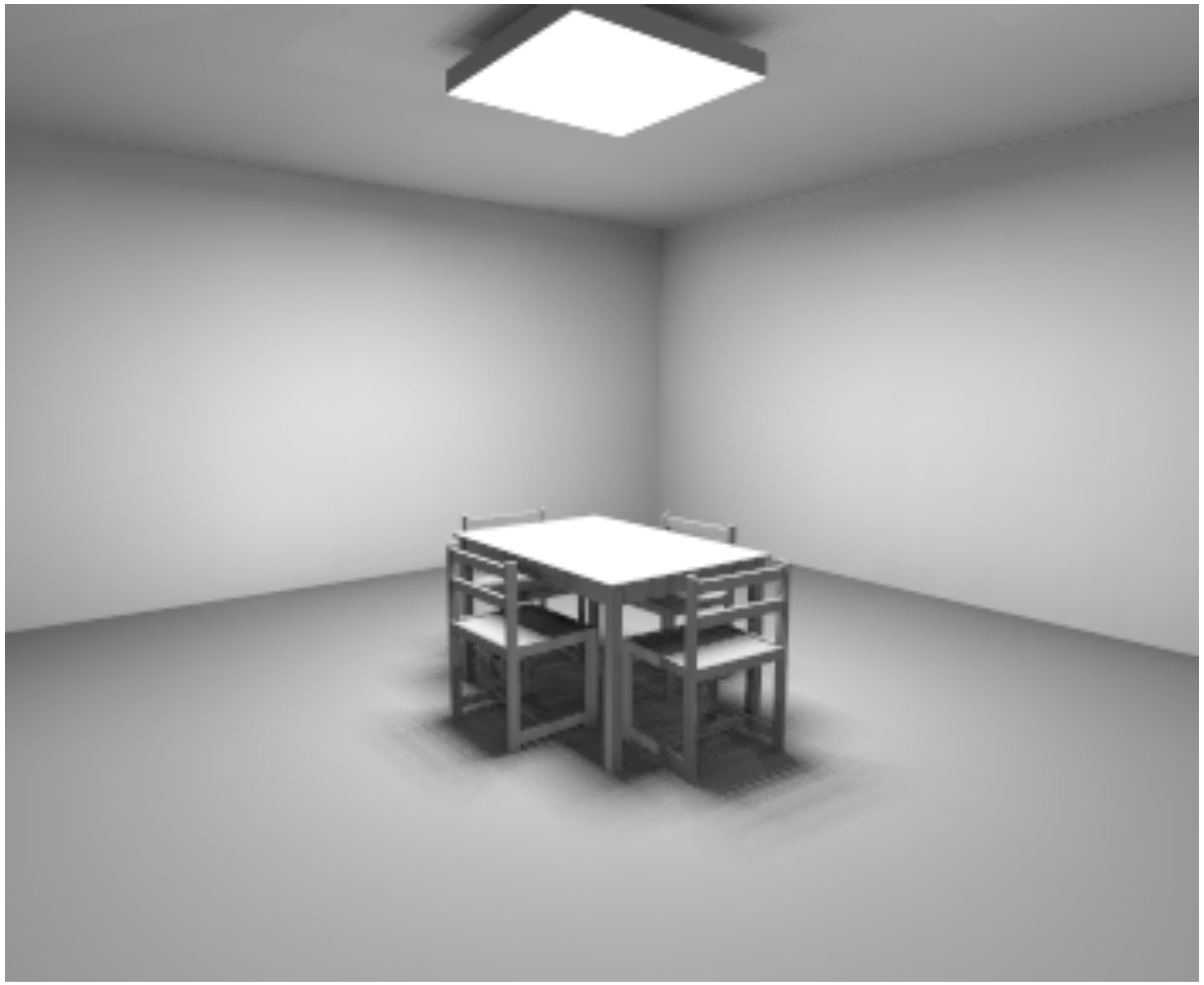
## Sum, N=10



Sum, N=32



# Sum, N=64

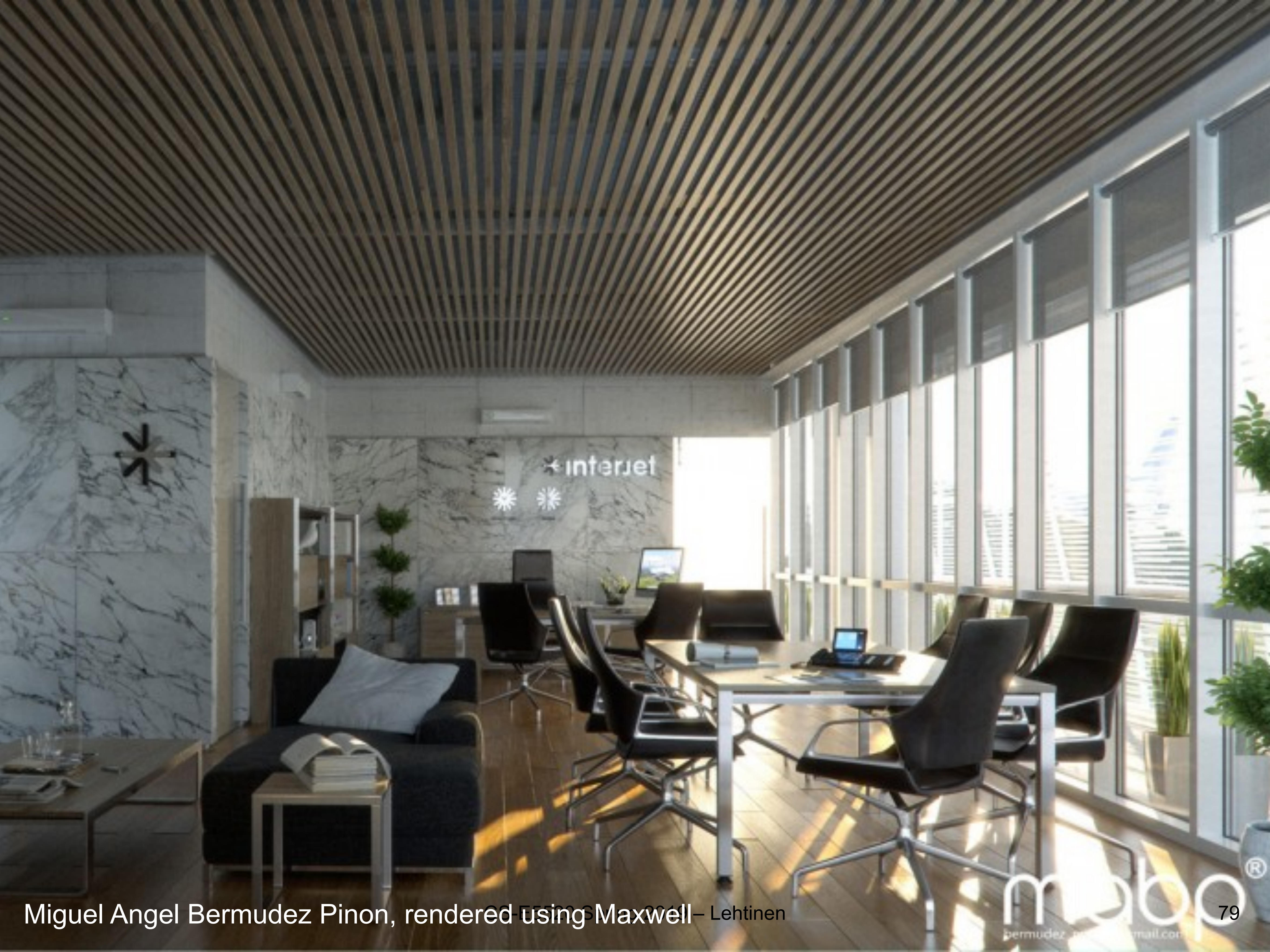


## Simple, Really

- All you need is a shadow mapper
  - Using a few dozen VPLs is real-time

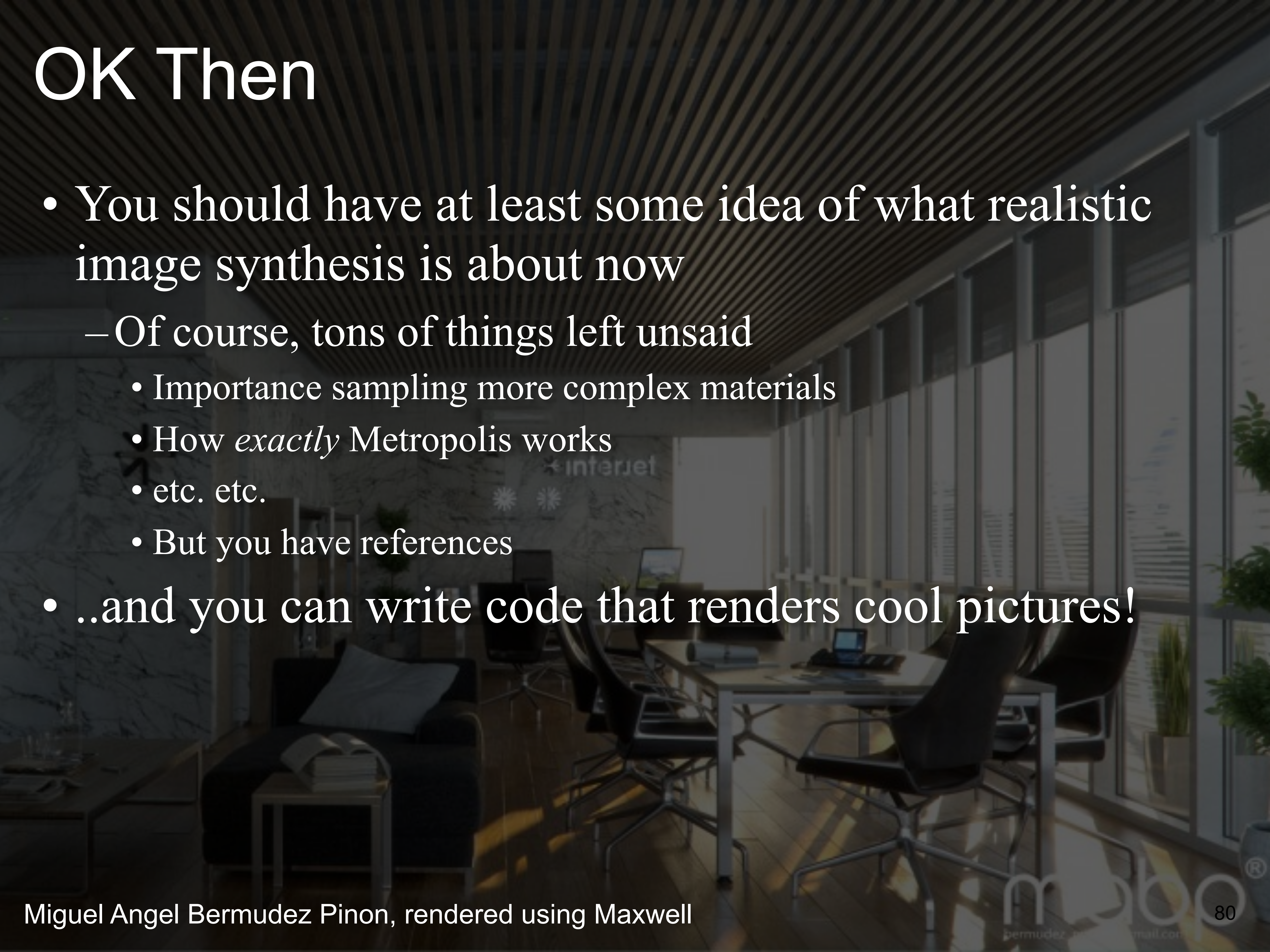
# Simple, Really

- All you need is a shadow mapper
  - Using a few dozen VPLs is real-time
- But problems with animation (flickering) unless you use tons of VPLs
  - But every VPL has to have its own shadow map
  - The lookups aren't free either
- Exploiting temporal coherence
  - Laine, Saransaari, Kontkanen, Lehtinen, Aila EGSR2007
  - Video (you've seen this before)



Miguel Angel Bermudez Pinon, rendered using Maxwell

## OK Then

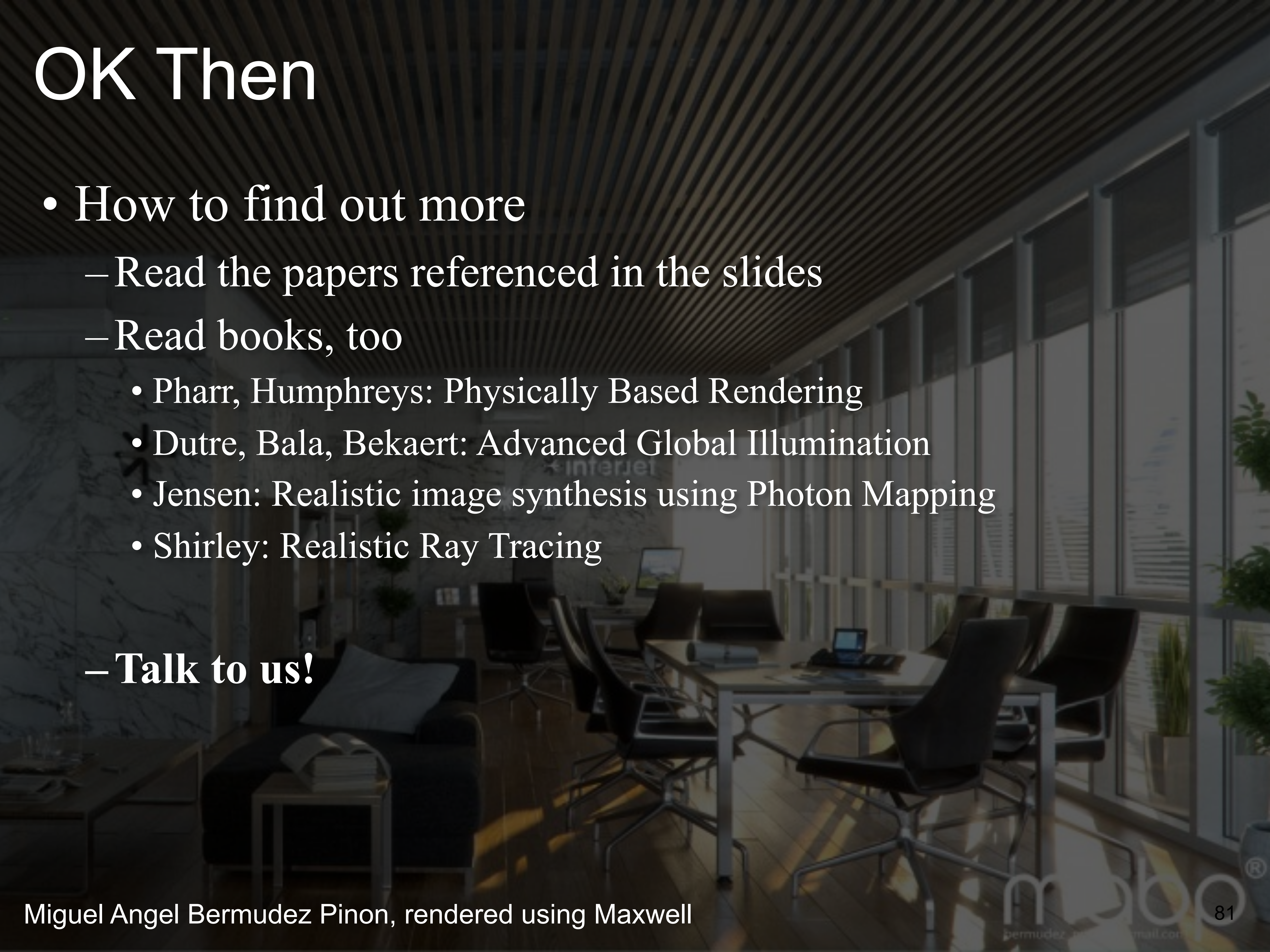


- You should have at least some idea of what realistic image synthesis is about now
  - Of course, tons of things left unsaid
    - Importance sampling more complex materials
    - How *exactly* Metropolis works
    - etc. etc.
    - But you have references
- ..and you can write code that renders cool pictures!

Miguel Angel Bermudez Pinon, rendered using Maxwell

## OK Then

- How to find out more
  - Read the papers referenced in the slides
  - Read books, too
    - Pharr, Humphreys: Physically Based Rendering
    - Dutre, Bala, Bekaert: Advanced Global Illumination
    - Jensen: Realistic image synthesis using Photon Mapping
    - Shirley: Realistic Ray Tracing
  - **Talk to us!**


Miguel Angel Bermudez Pinon, rendered using Maxwell

## …hope you've had fun!

Miguel Angel Bermudez Pinon, rendered using Maxwell