Reconstruction I

Philipp Slusallek   *Karol Myszkowski*
Gurprit Singh

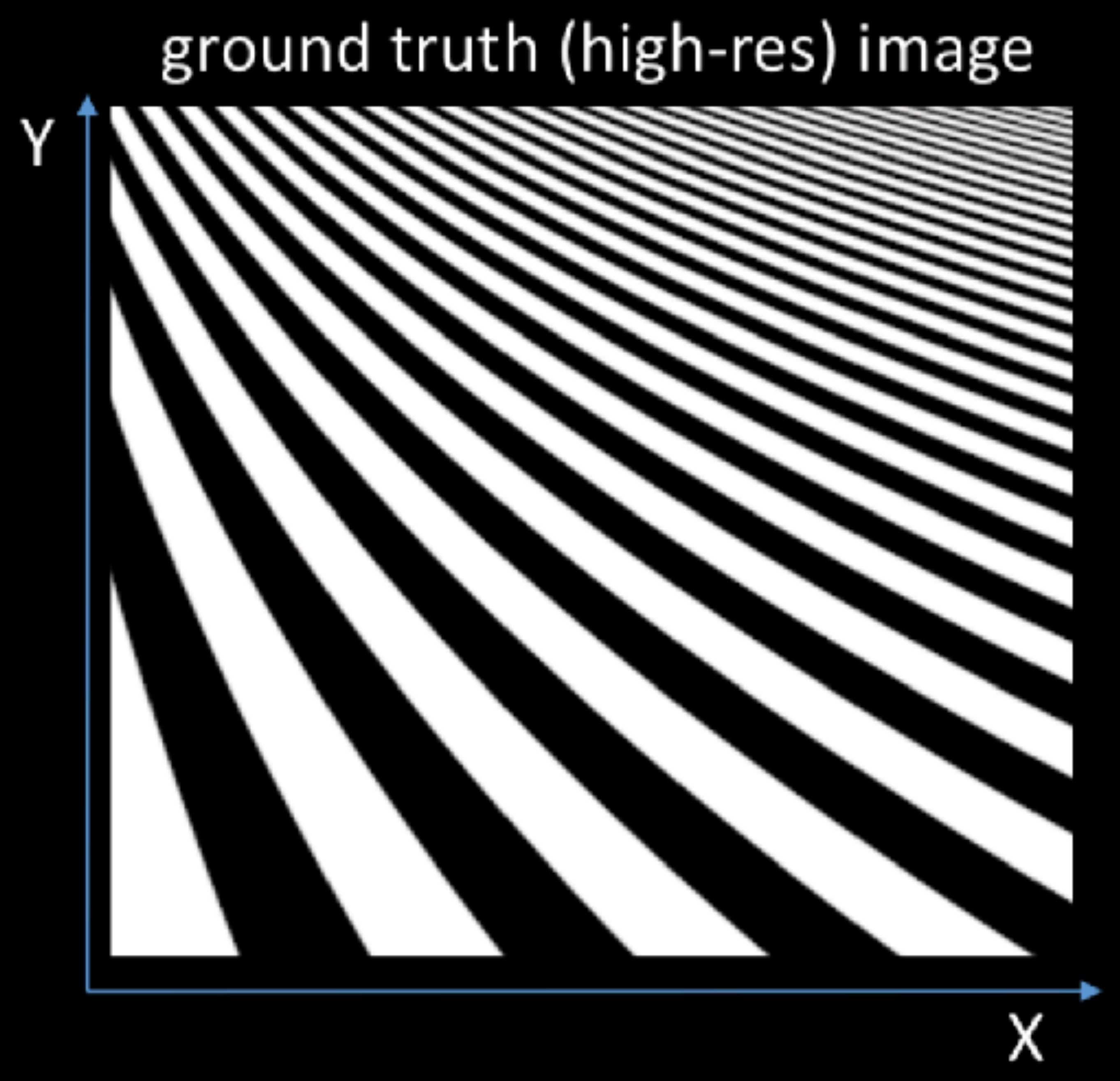
À la Carte

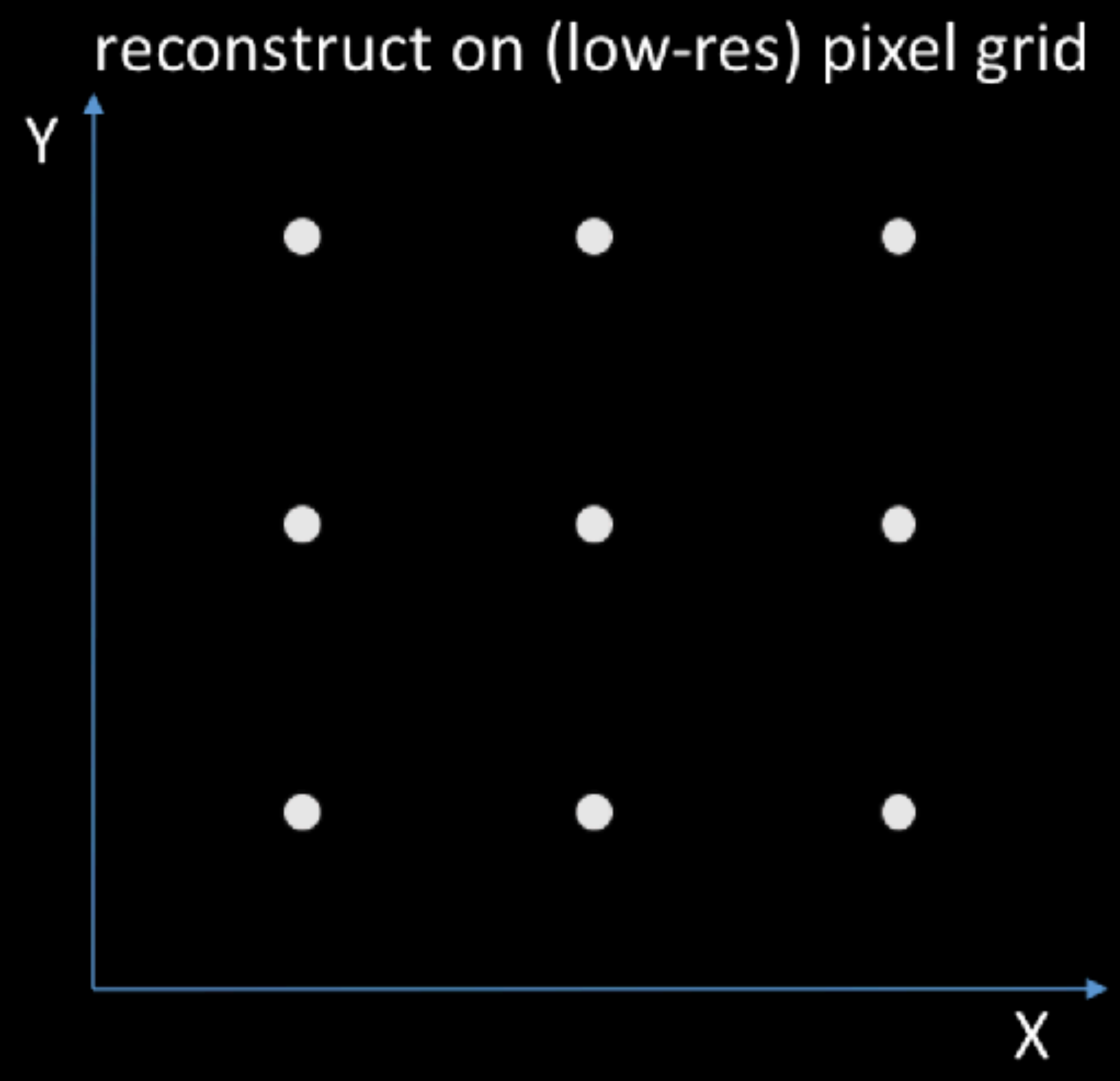
- Reconstruction vs Integration
- Multi-dimensional Sampling and Reconstruction
- Temporal Light Field Reconstruction
- Random Parameter Filtering of Monte Carlo Noise

Reconstruction vs Integration

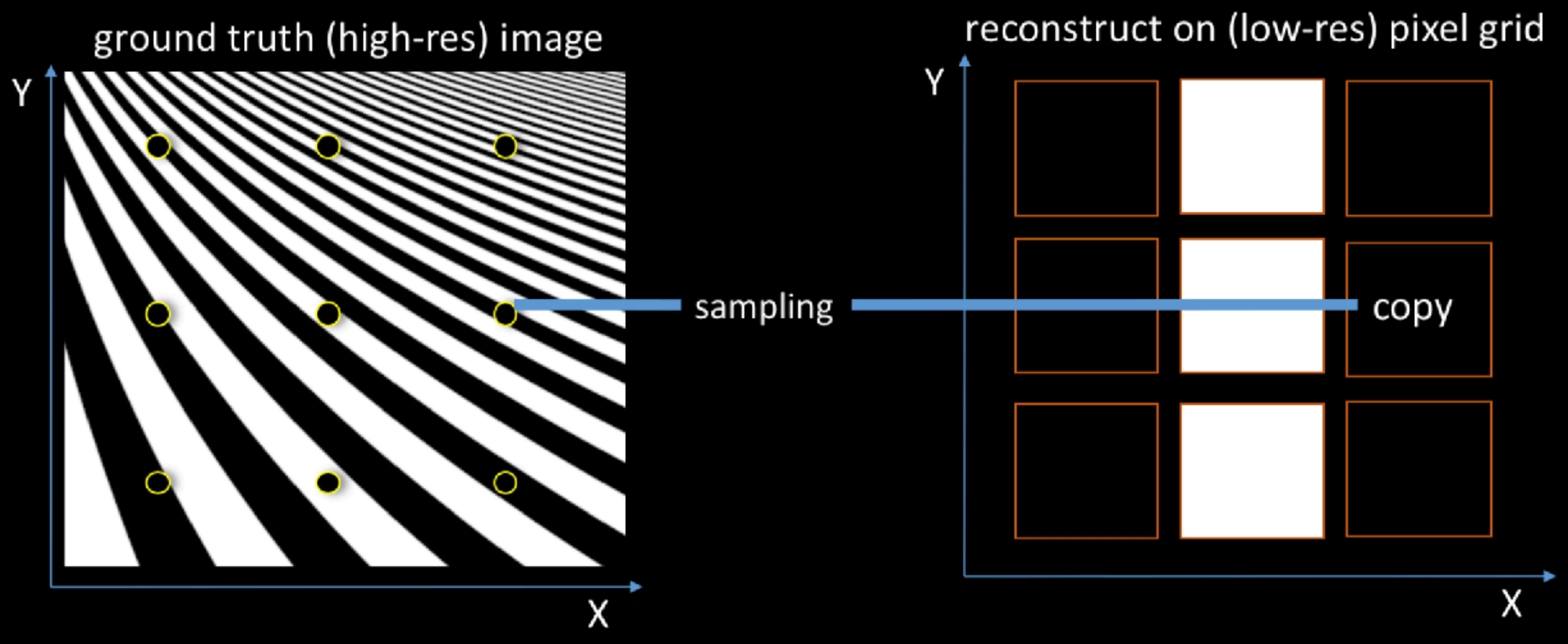
Slides courtesy: Kartic Subr

Reconstruction: estimate image samples

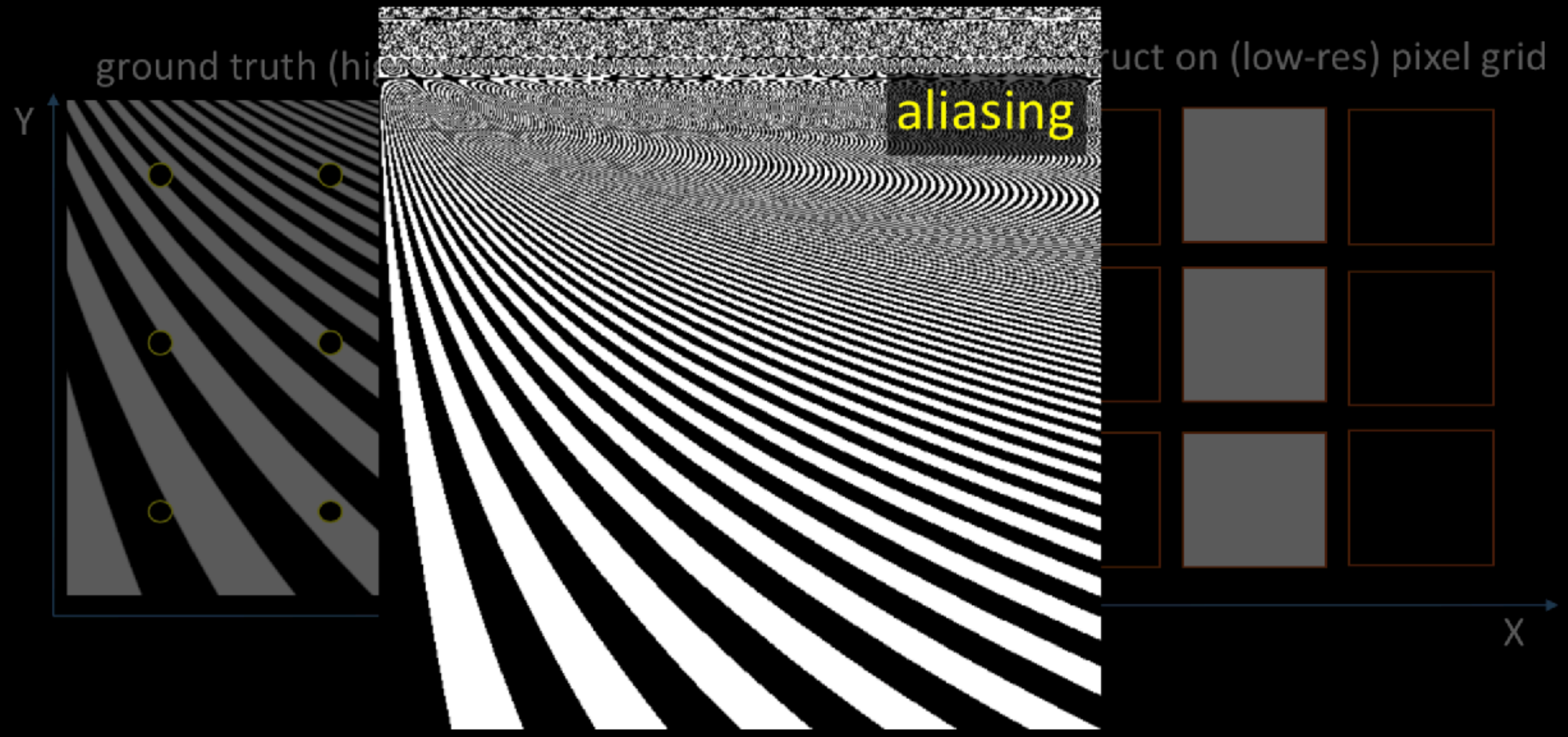




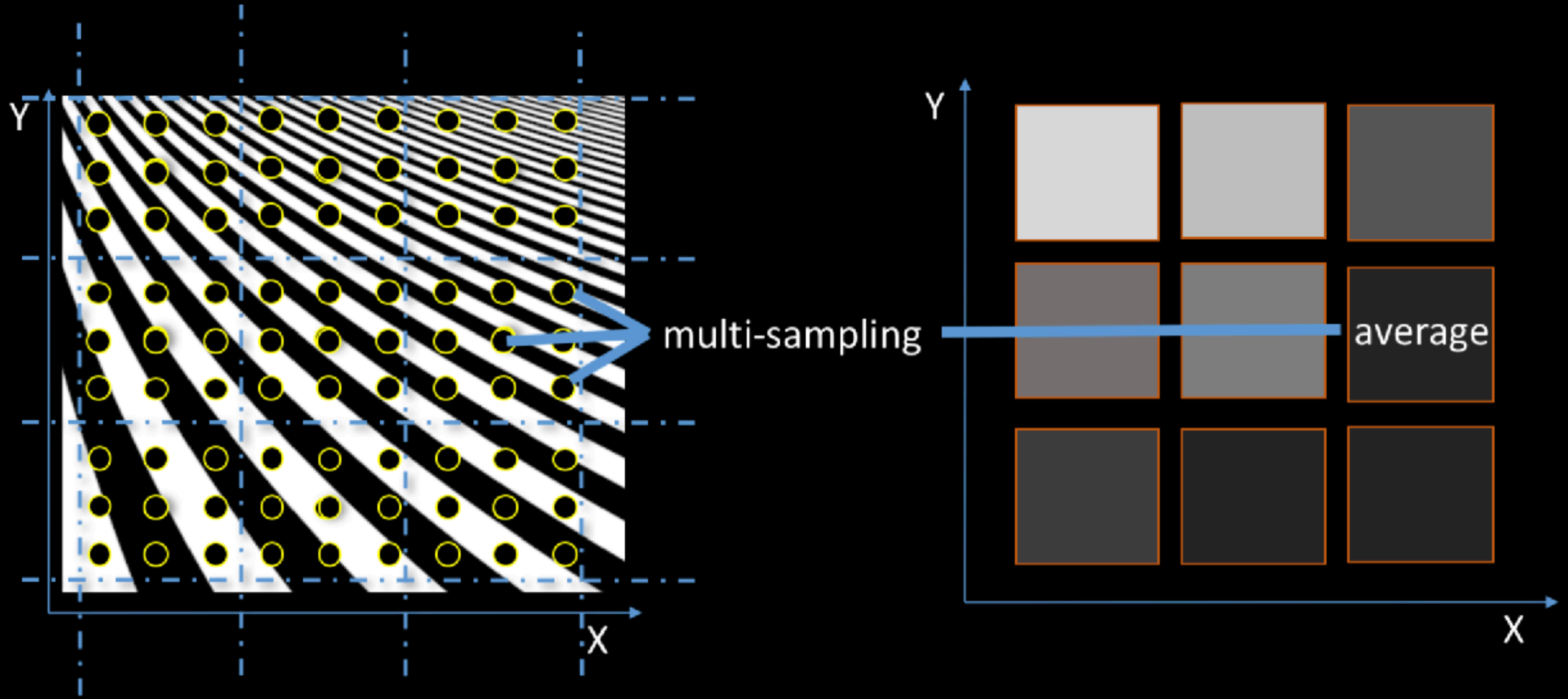
Naïve method: sample image at grid locations



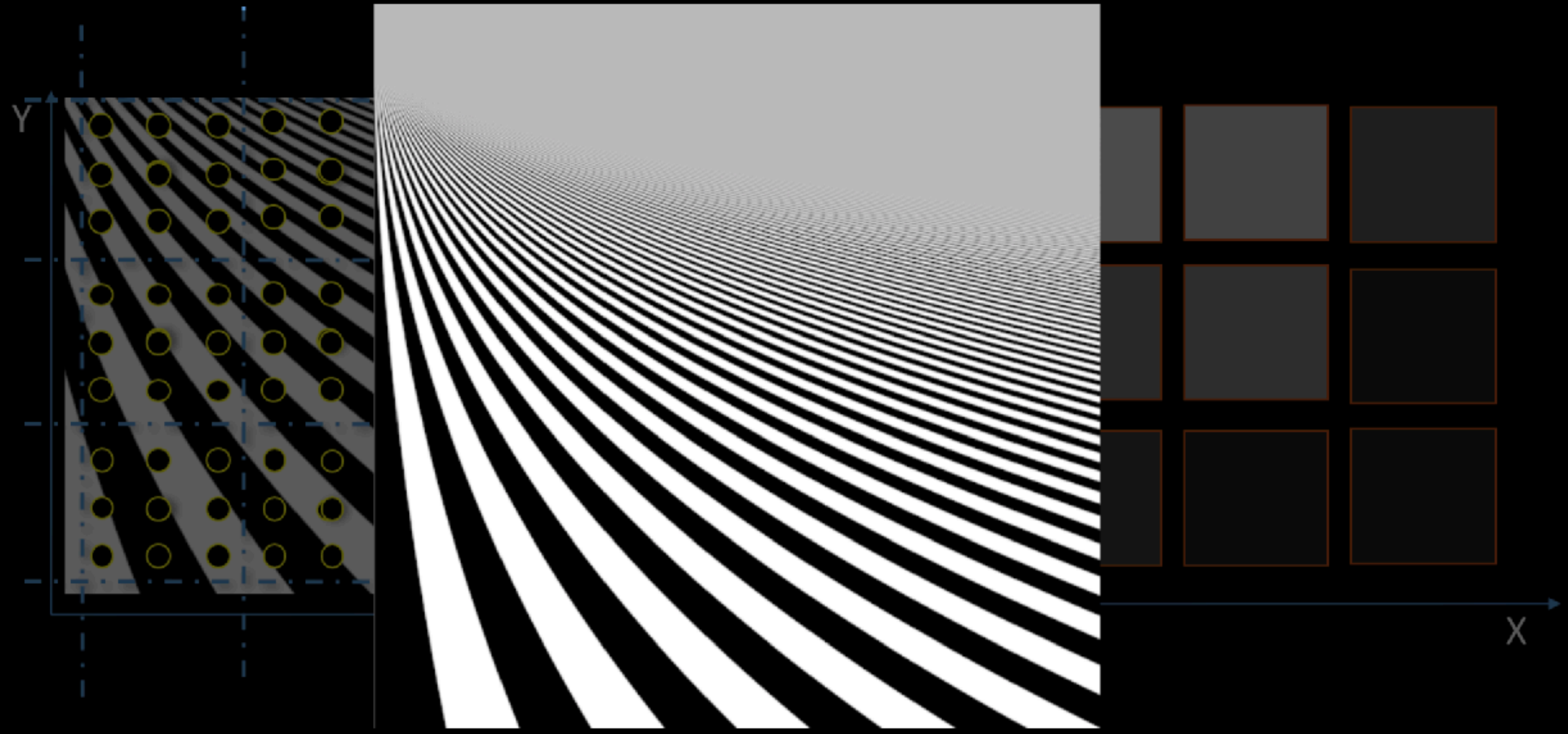
Naïve method: when sampling is increased



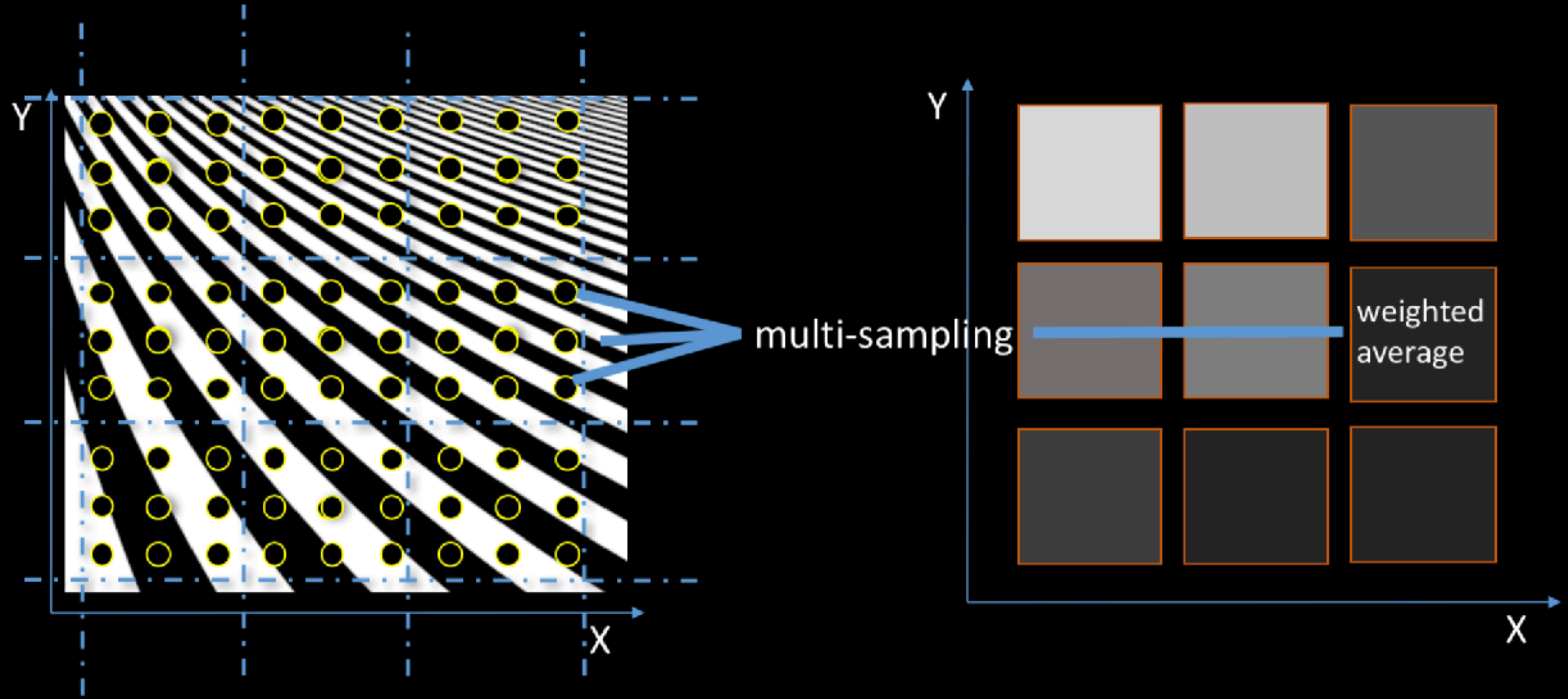
Antialiasing: assuming `square' pixels



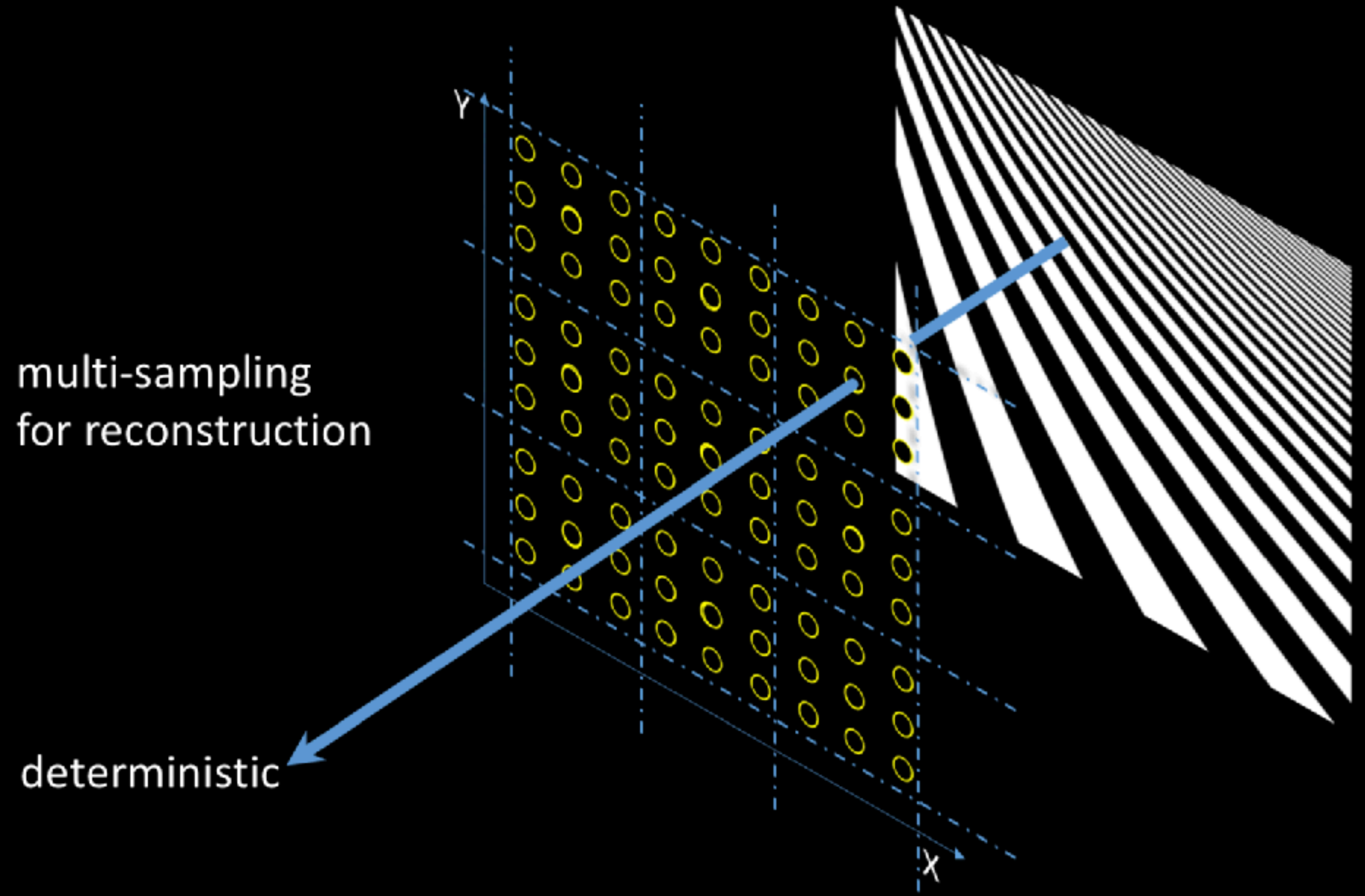
Antialiasing is costly due to multi-sampling



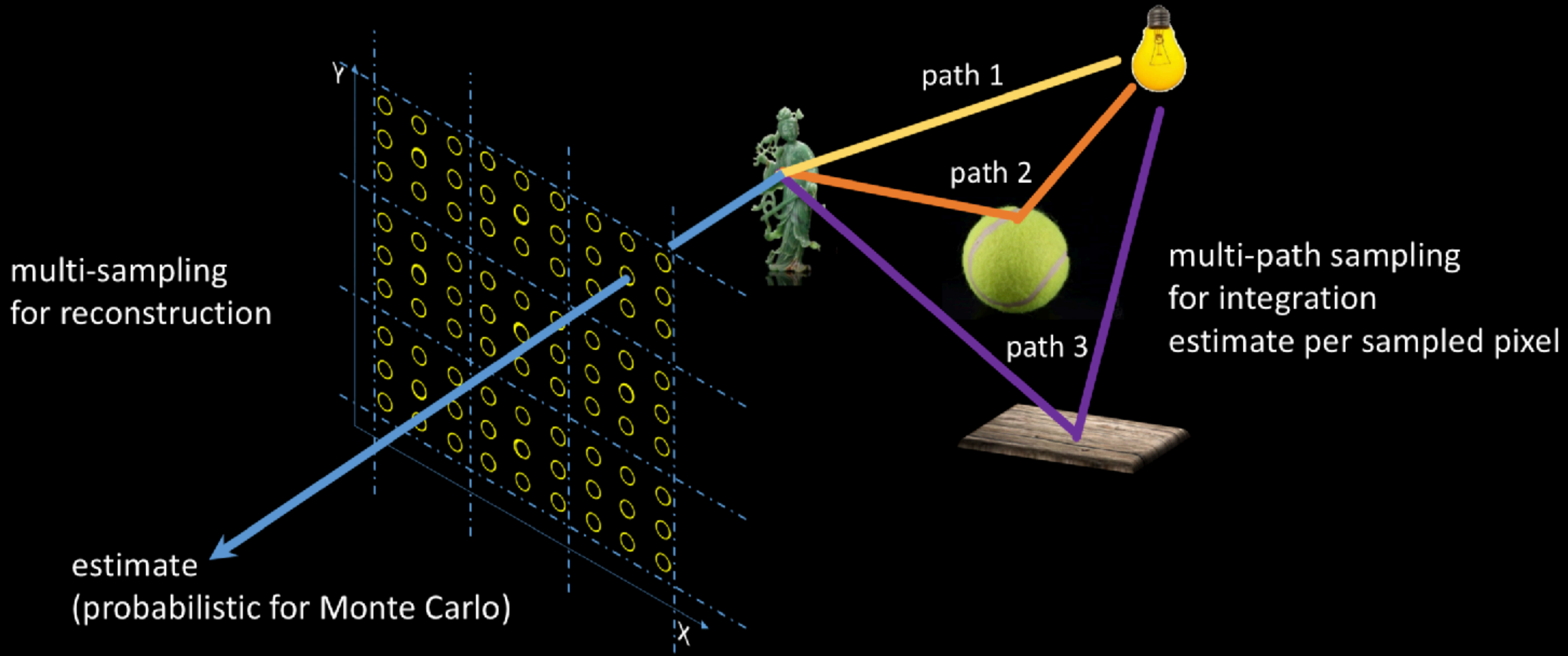
Antialiasing using general reconstruction filter



Rendering: Reconstructing integrals

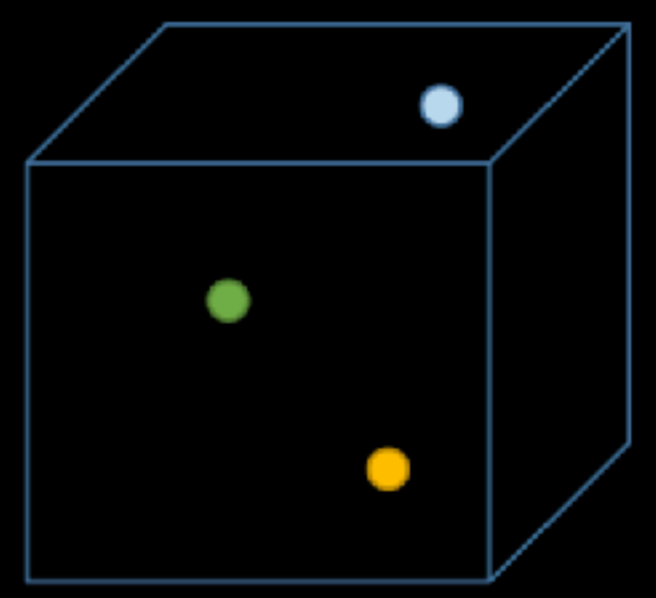


Rendering: Reconstructing integrals

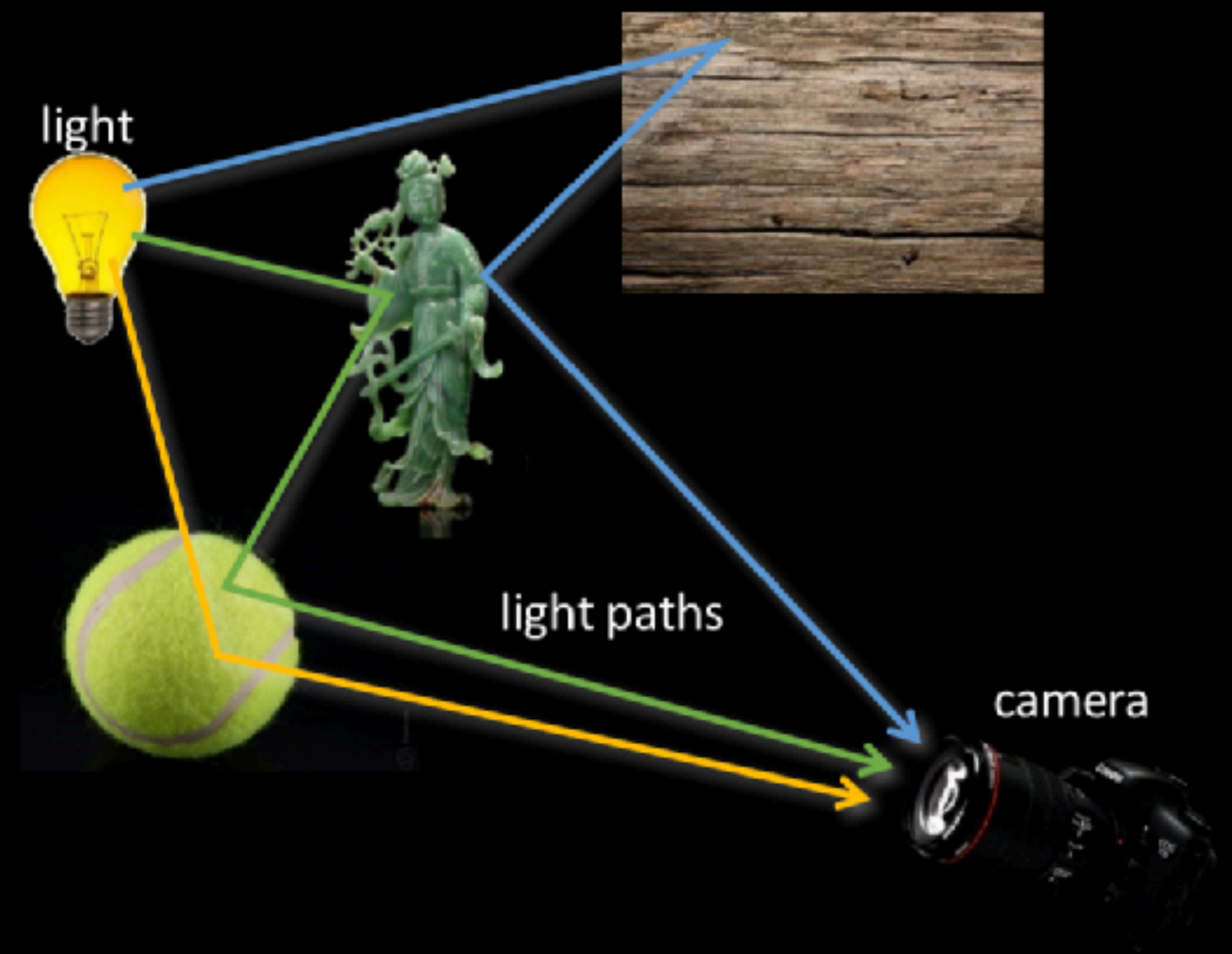


Function-space view: Sampling in path space

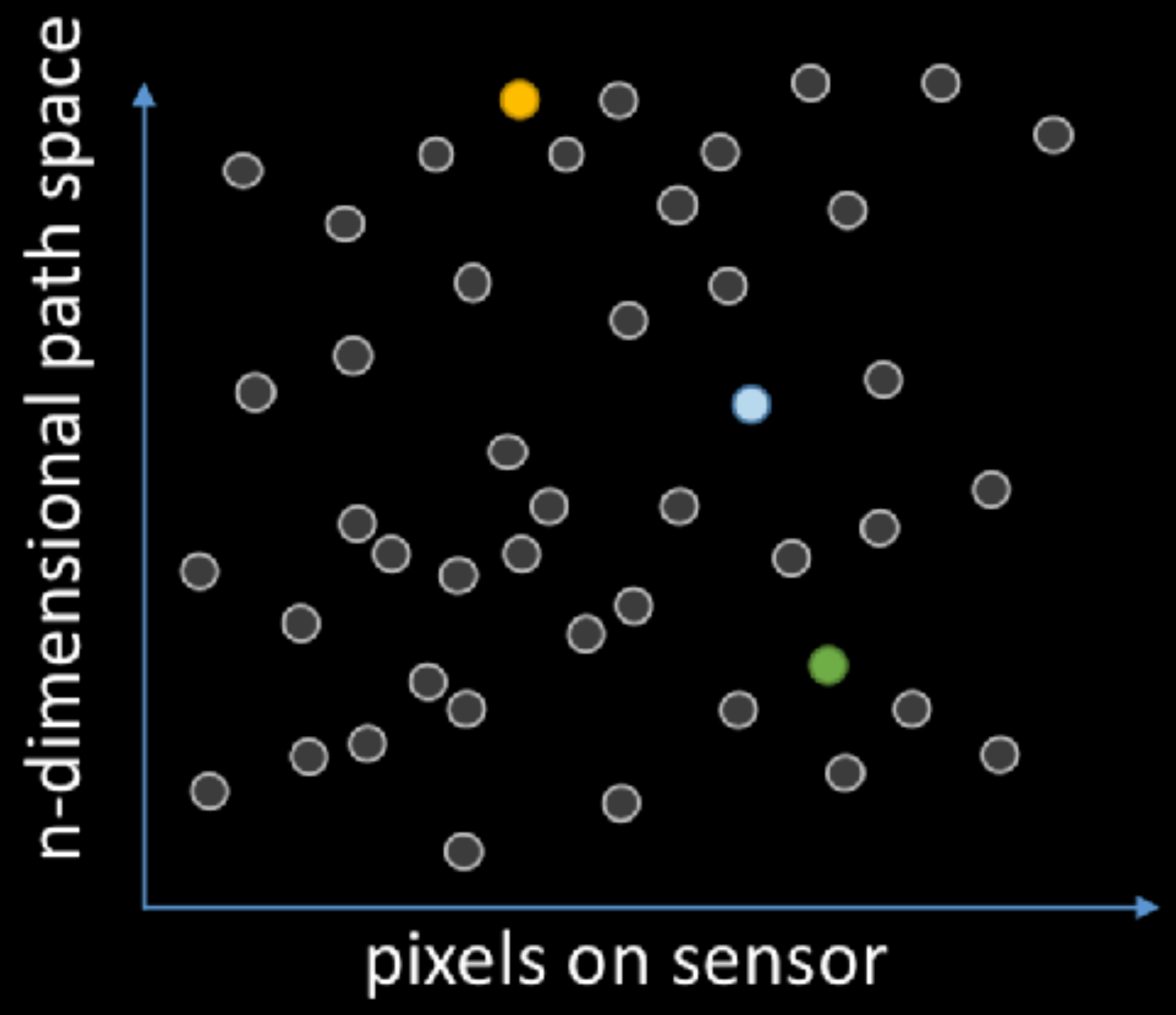
each sample represents a path
and has an associated radiance value



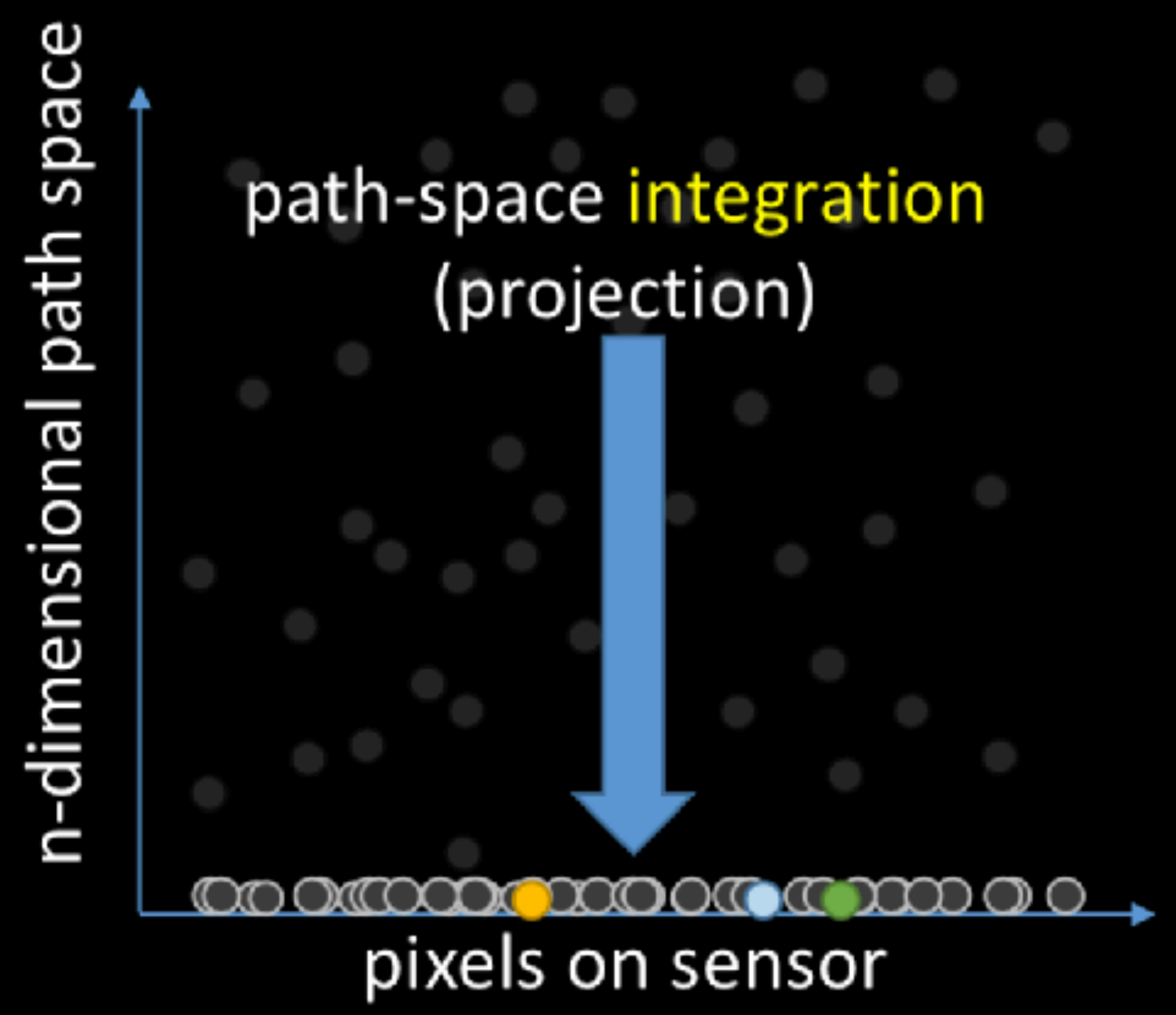
n-dimensional path space

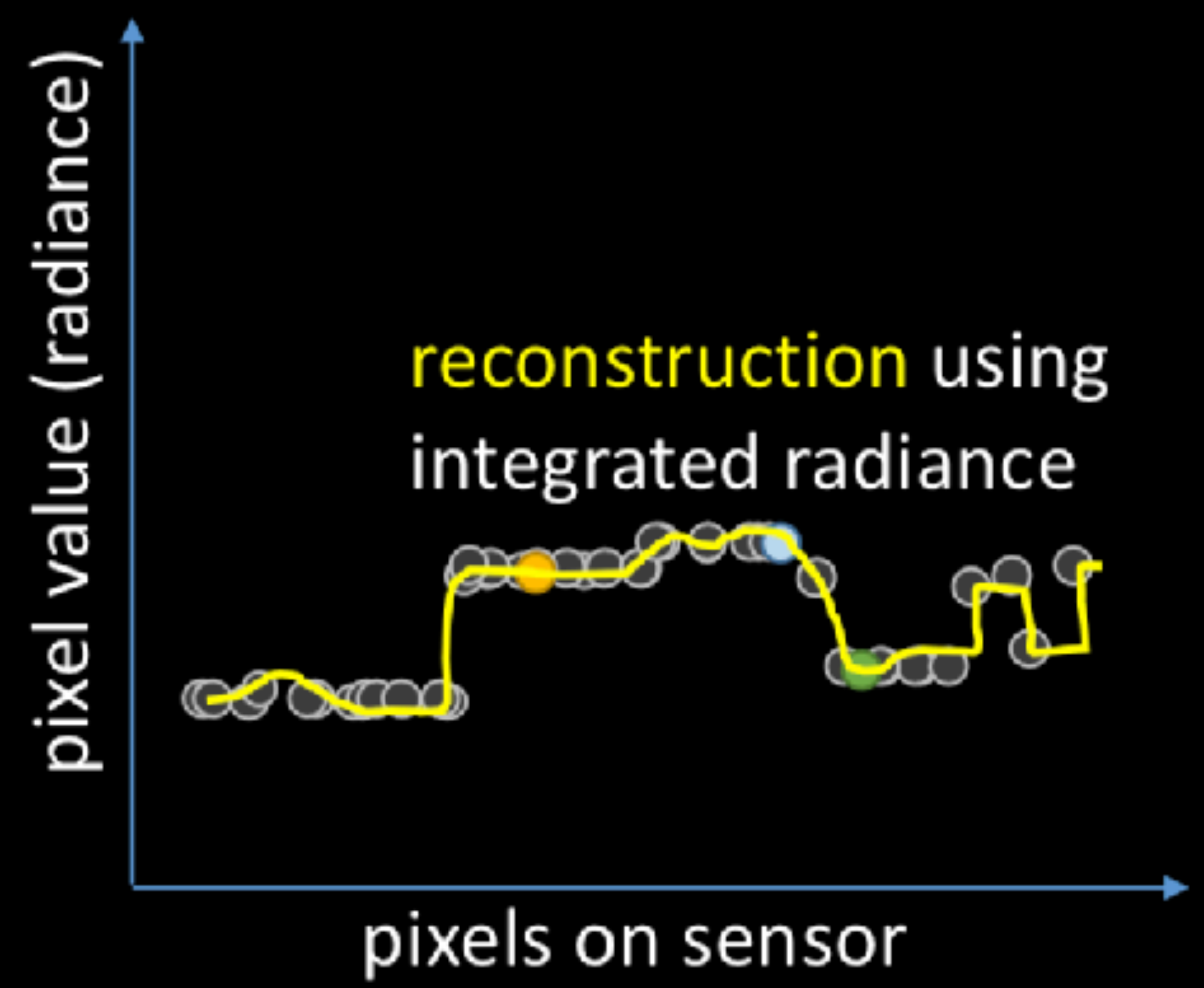


Sample locations shown in path-pixel space



Rendering = integration + reconstruction





Multi-dimensional Adaptive Sampling and Reconstruction

Hachisuka et al. [2008]

Slides courtesy: Toshiya Hachisuka

Temporal Light-Field Reconstruction for Rendering Distribution Effects

Lehtinen et al. [2011]

Slides courtesy: Jakko Lehtinen

Pinhole image

Requires dense sampling of 5D function:

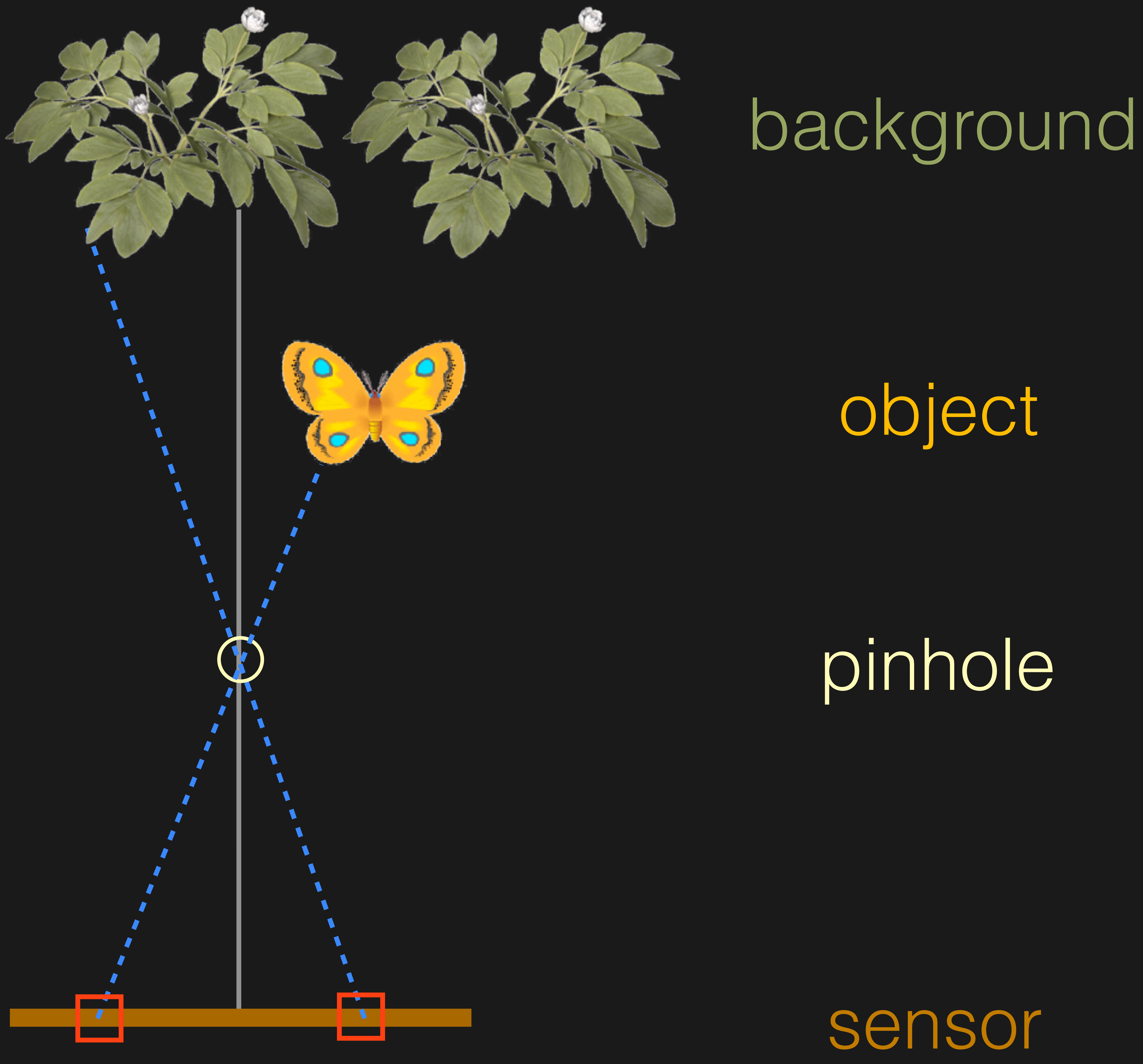
- Pixel area (2D)
- Lens aperture (2D)
- Time (1D)

With motion blur and depth of field

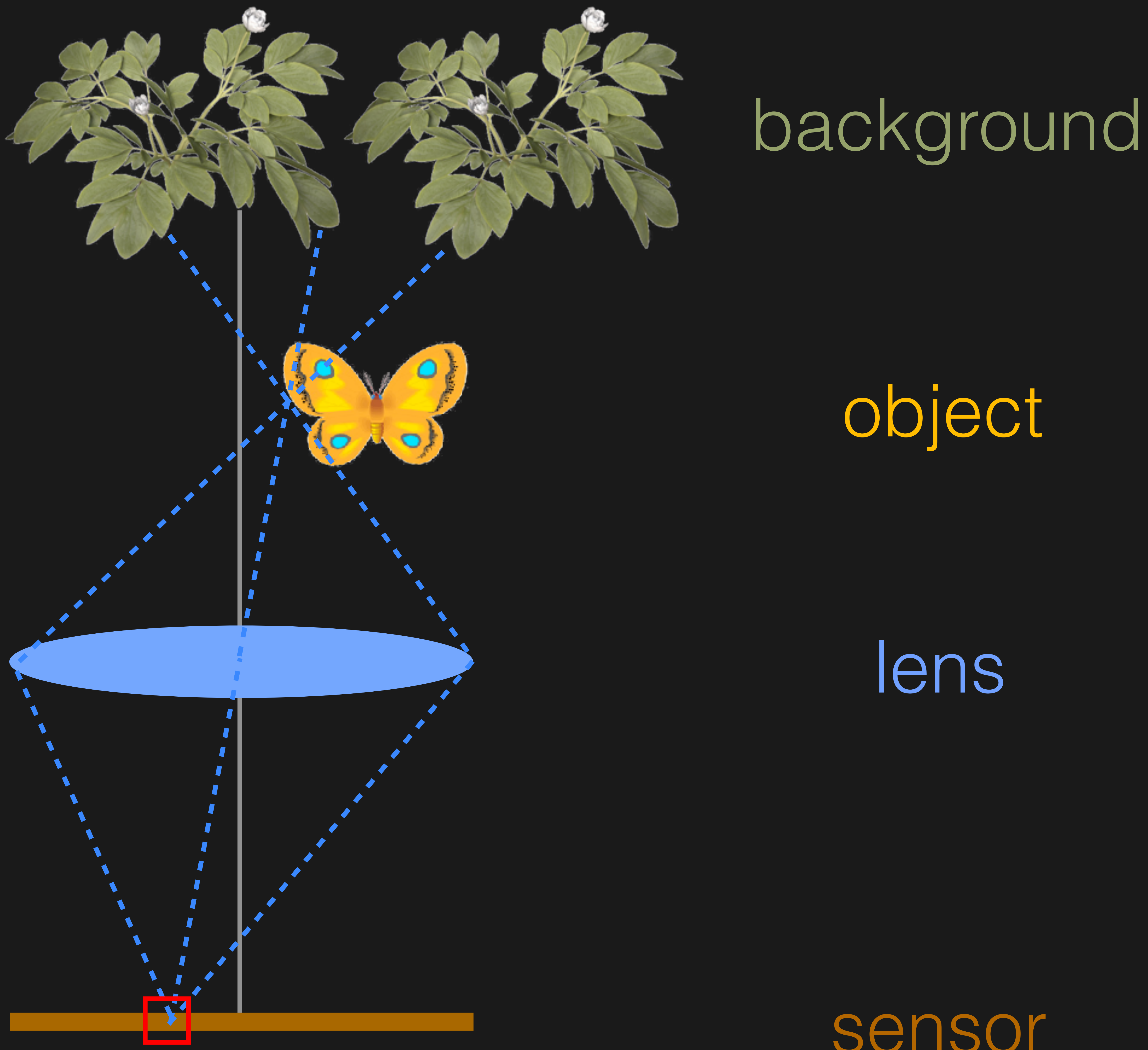
Motion blur and depth of field 1 sample per pixel


Our reconstruction

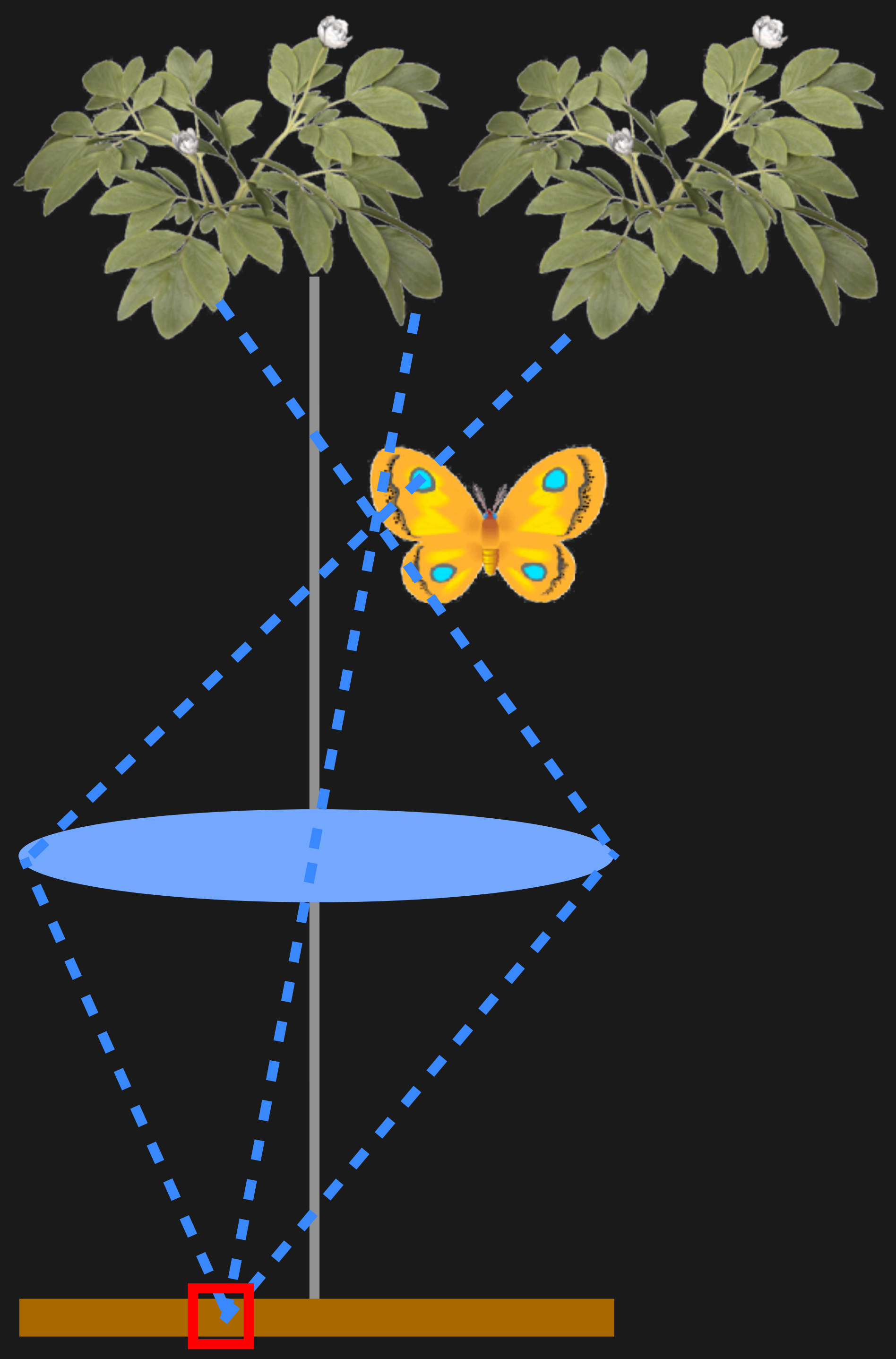
Pinhole camera model



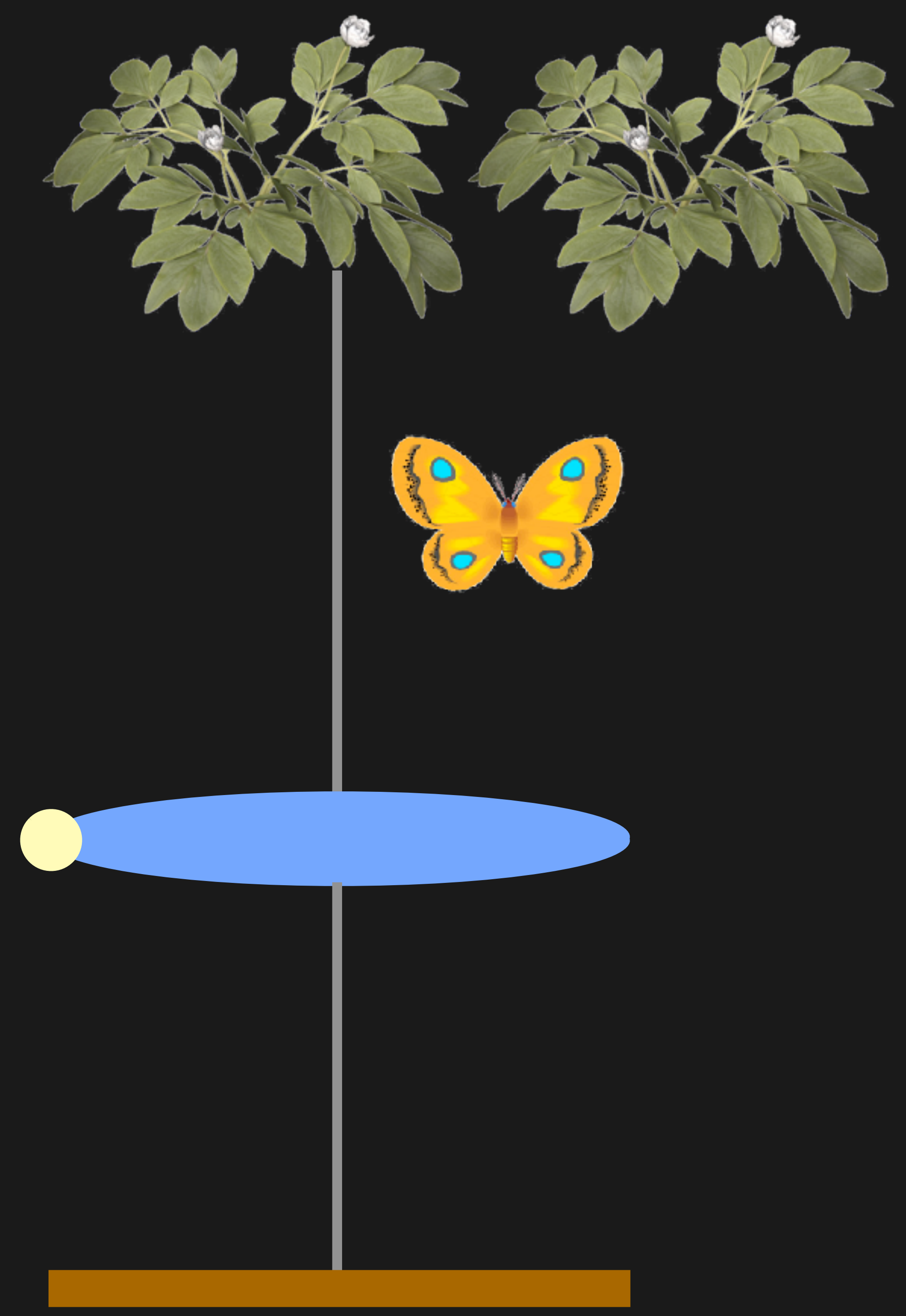
Thin lens camera model



Depth of field



Depth of field




1 scanline


Lens u
Screen x


1 pixel
Lens u
Screen x

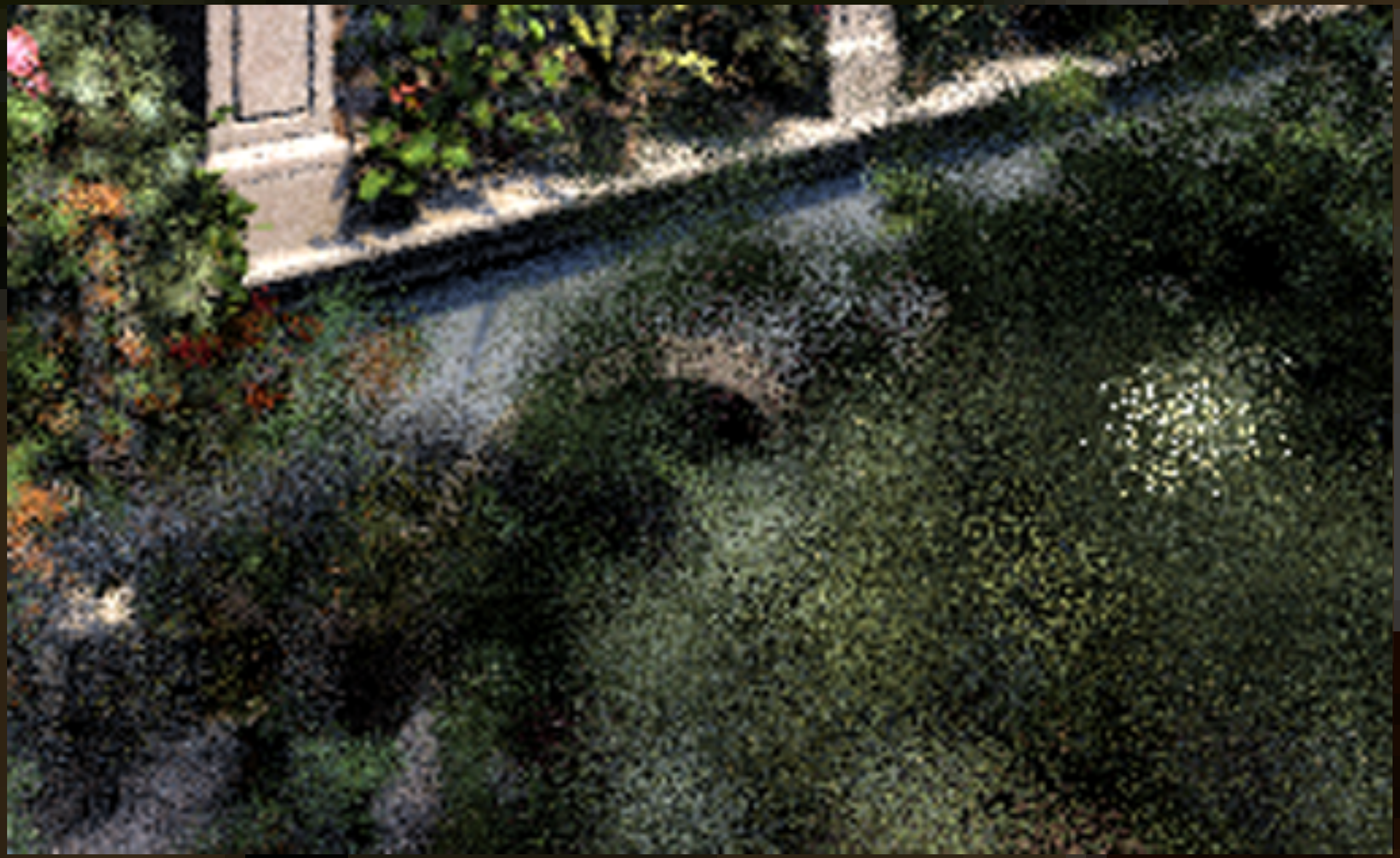

Lens u
Screen x

Light field [Levoy 1996]

Monte Carlo sampling

Low sample density leads to noise

1 pixel



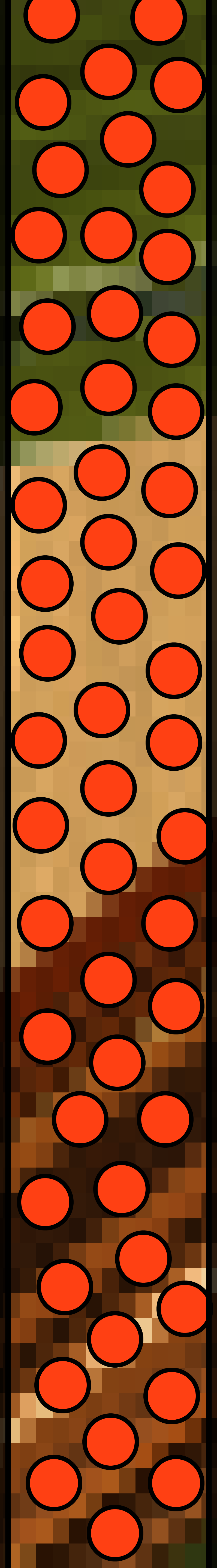
Lens u

Screen x

Monte Carlo sampling

Need many samples to capture the signal:

computationally expensive

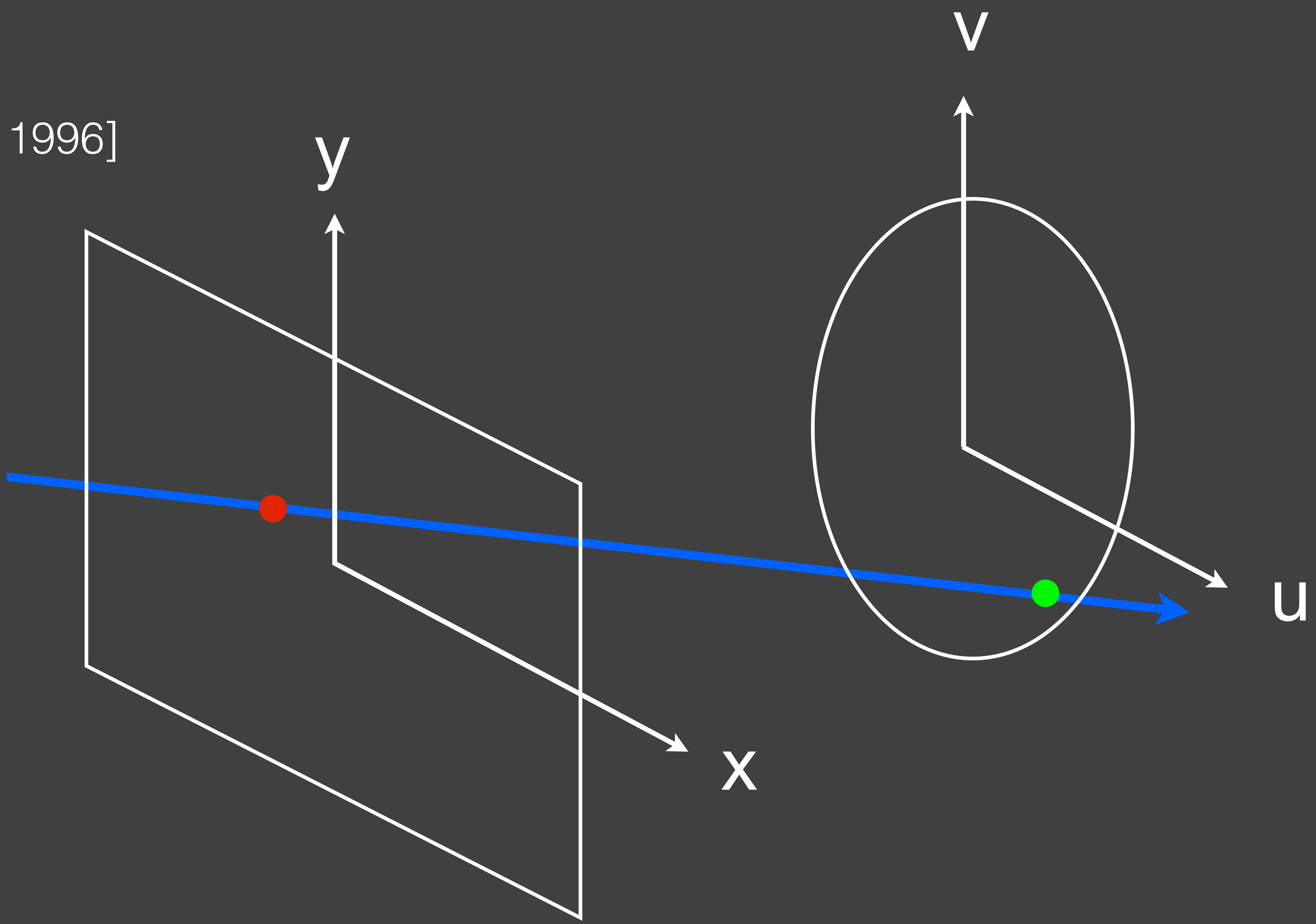


Temporal light fields

Traditional light field is 4D [Levoy 1996]

x,y over sensor (2D)
u,v over lens (2D)

Add time dimension for moving geometry (5D)




Lens u
Screen x


Lens u
Screen x

The Integrand is Anisotropic

[Chai00, Durand05, Hachisuka08, Soler09, Egan09, ...]

Multi-dimensional Adaptive Sampling [Hachisuka 08]

Frequency Analysis and Sheared Reconstruction [Egan 09]

Our approach

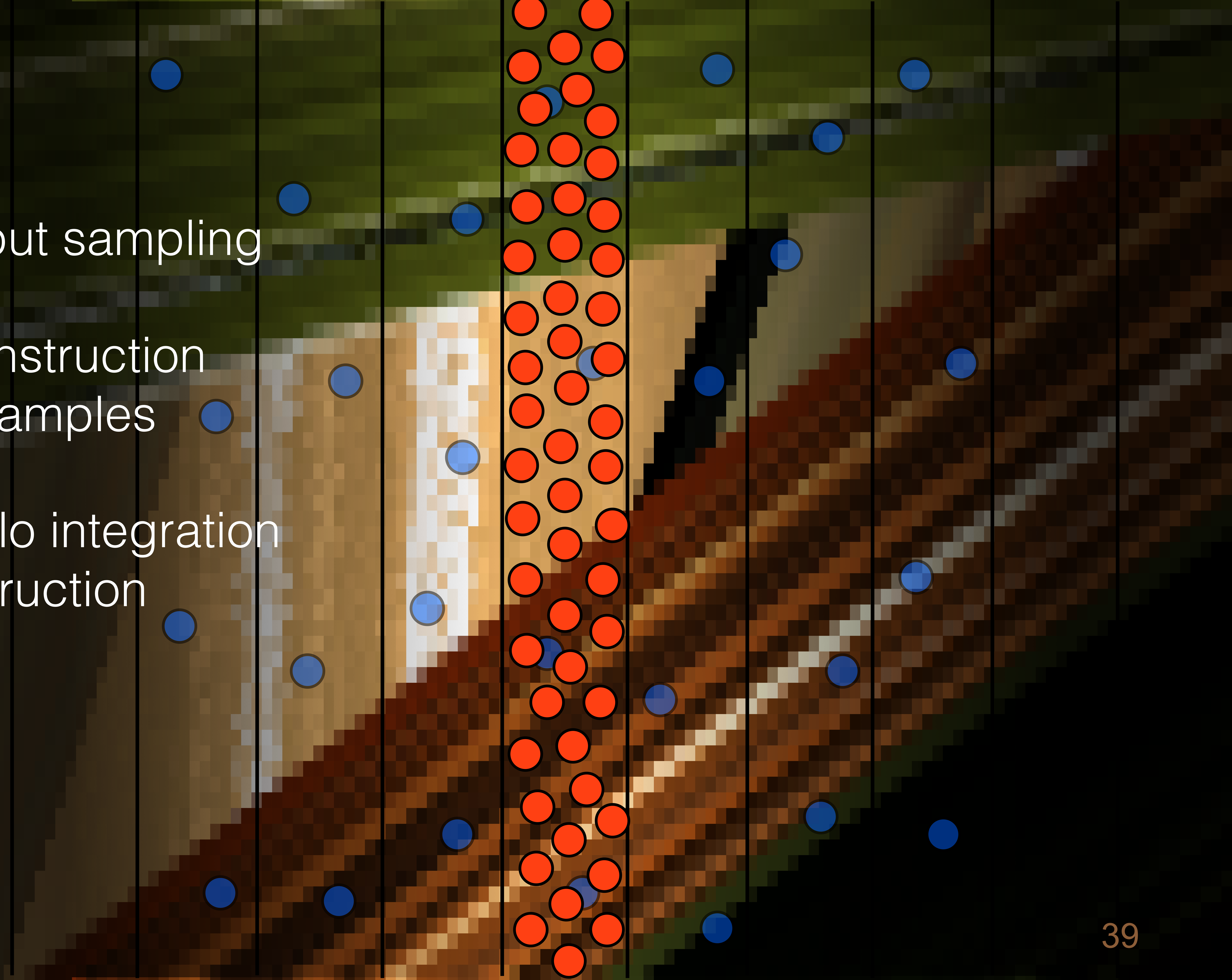
Start with **sparse** input sampling

Our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

Standard Monte-Carlo integration using dense reconstruction



Our input has **slope information**

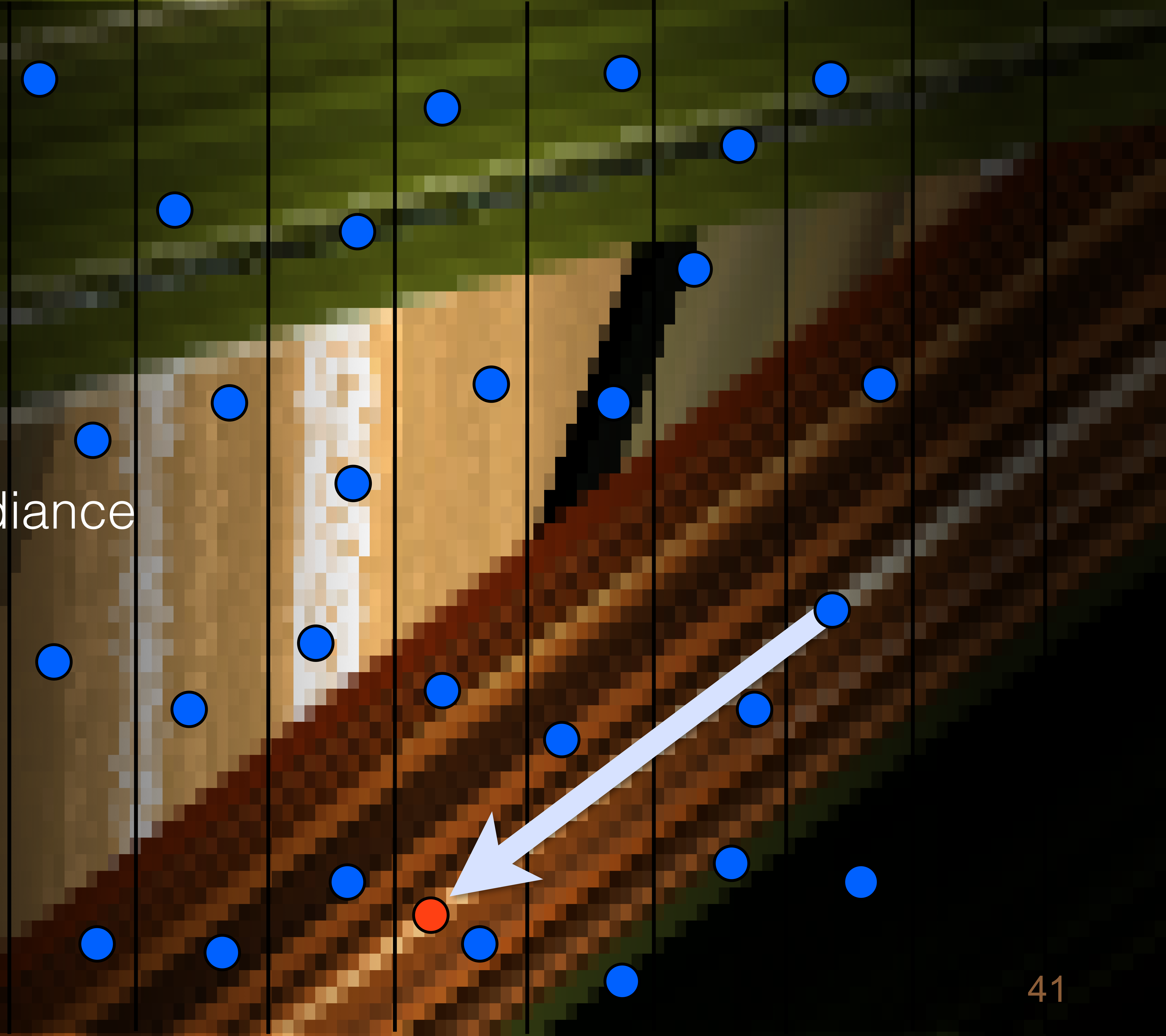
For defocus, proportional to inverse **depth 1/z** [Chai00]

For motion, proportional to inverse **velocity 1/v** [Egan09]

Easy to output from any renderer.

What is the radiance at the red location?

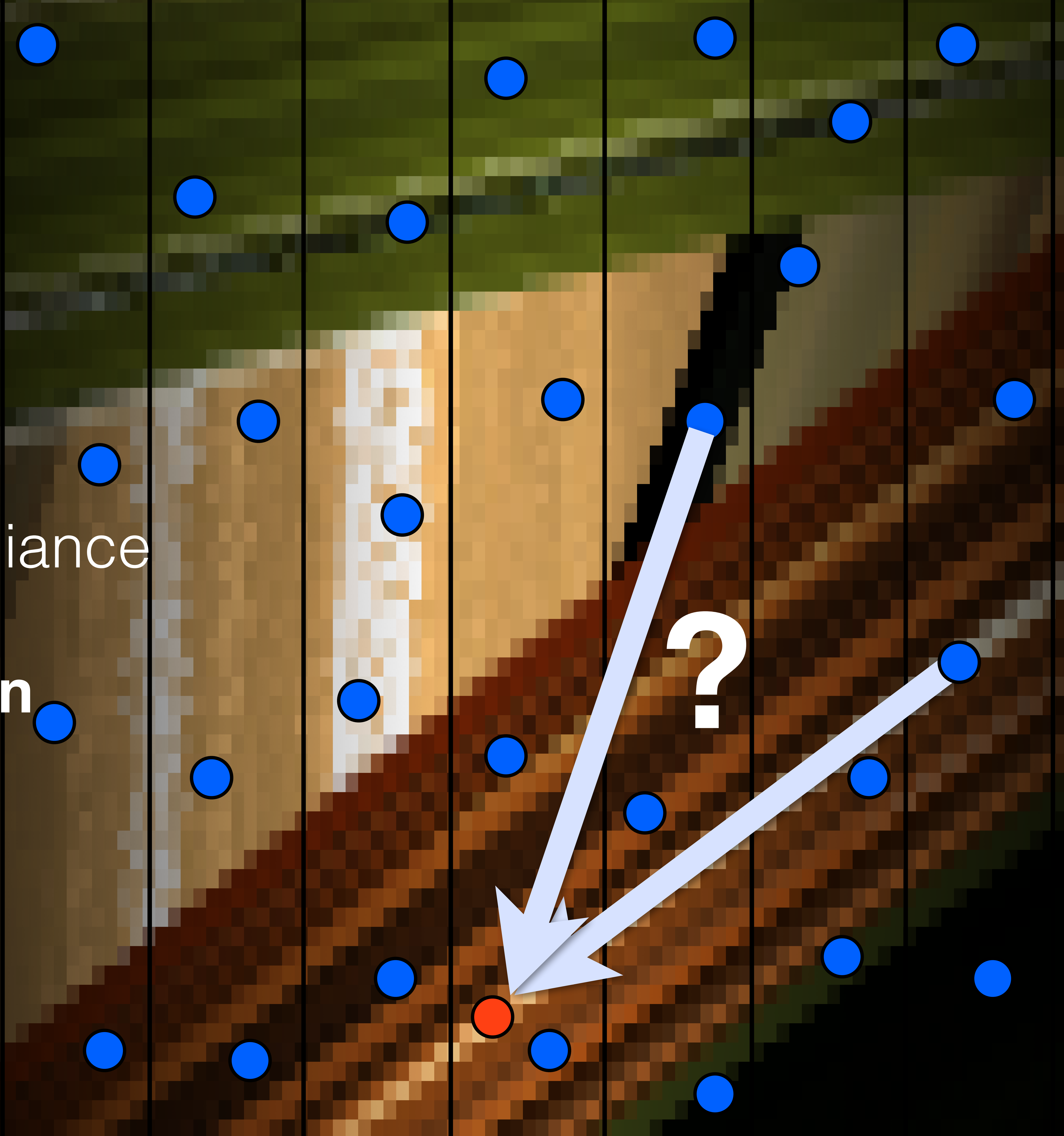
Use slope to **reproject** radiance



What is the radiance at the red location?

Use slope to **reproject** radiance

Must account for **occlusion**



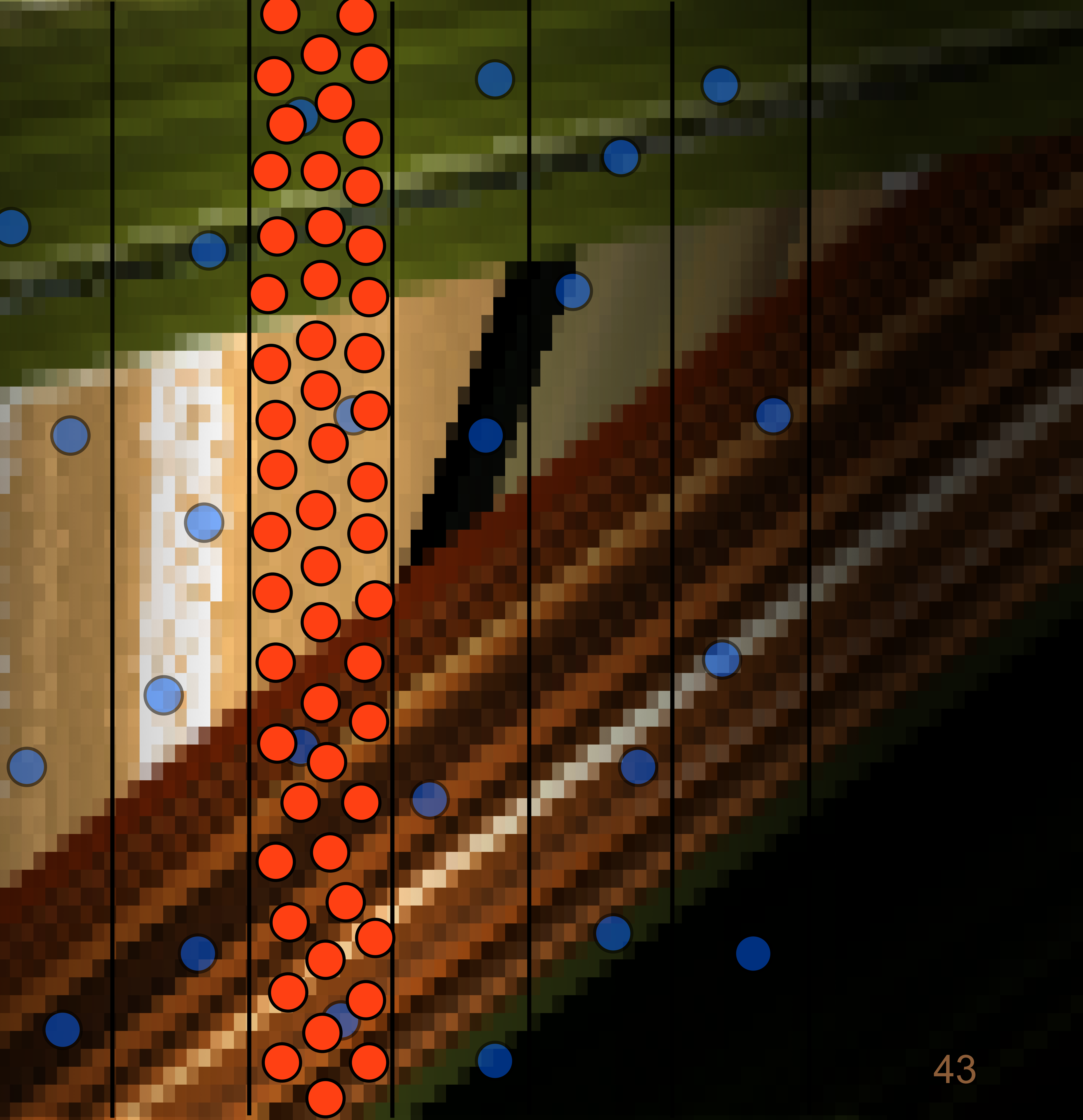
Recap: our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

- Use **slopes** to reproject
- Account for **visibility**

Standard Monte-Carlo integration using dense reconstruction



Reprojection and filtering

Simplify visibility by reprojecting into **screen space**.

Reproject to u, v, t of reconstruction location.

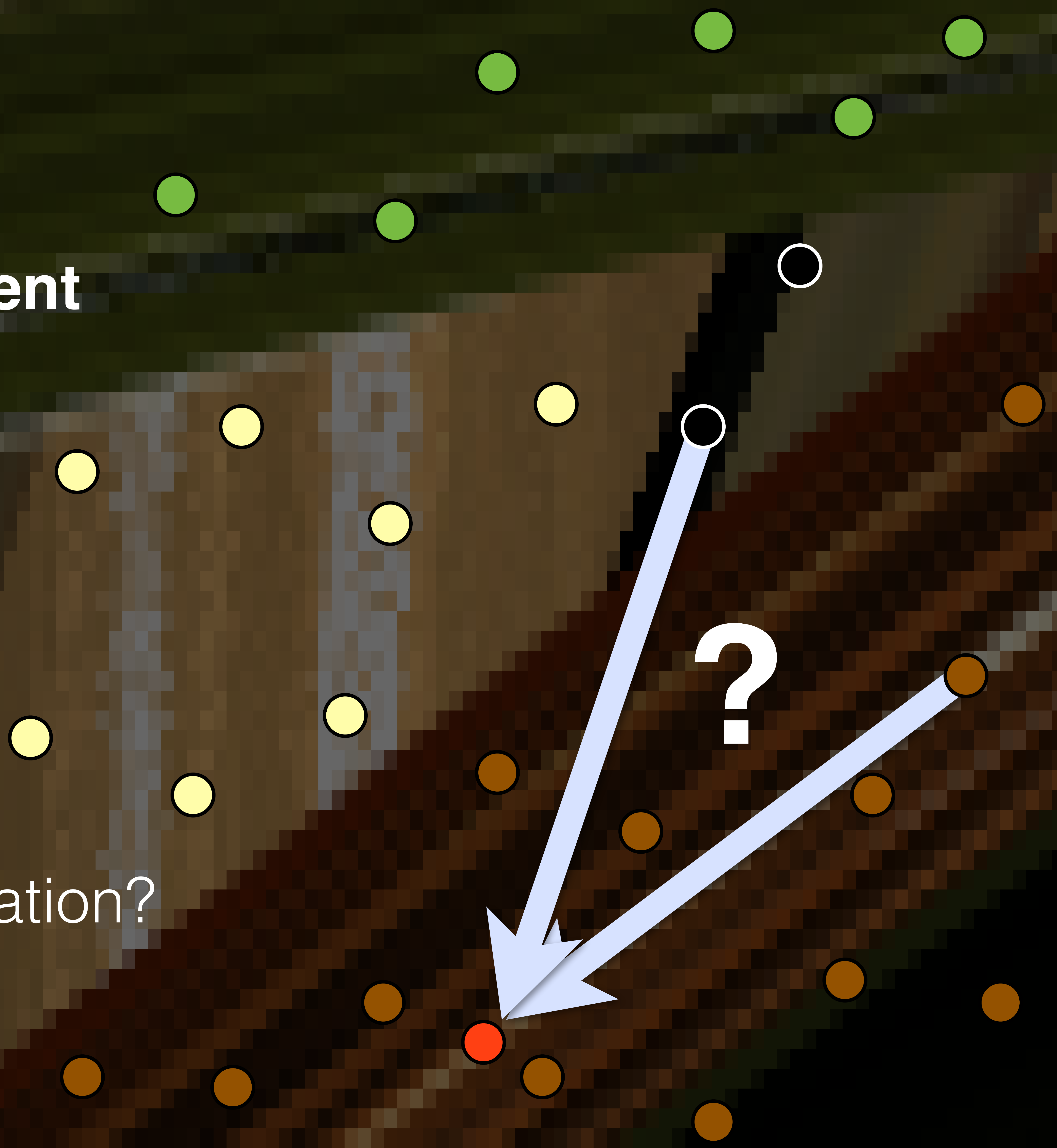
Pixel filter over **visible** samples.

Visibility

Cluster samples into **apparent surfaces** to resolve *z*-order

SameSurface algorithm

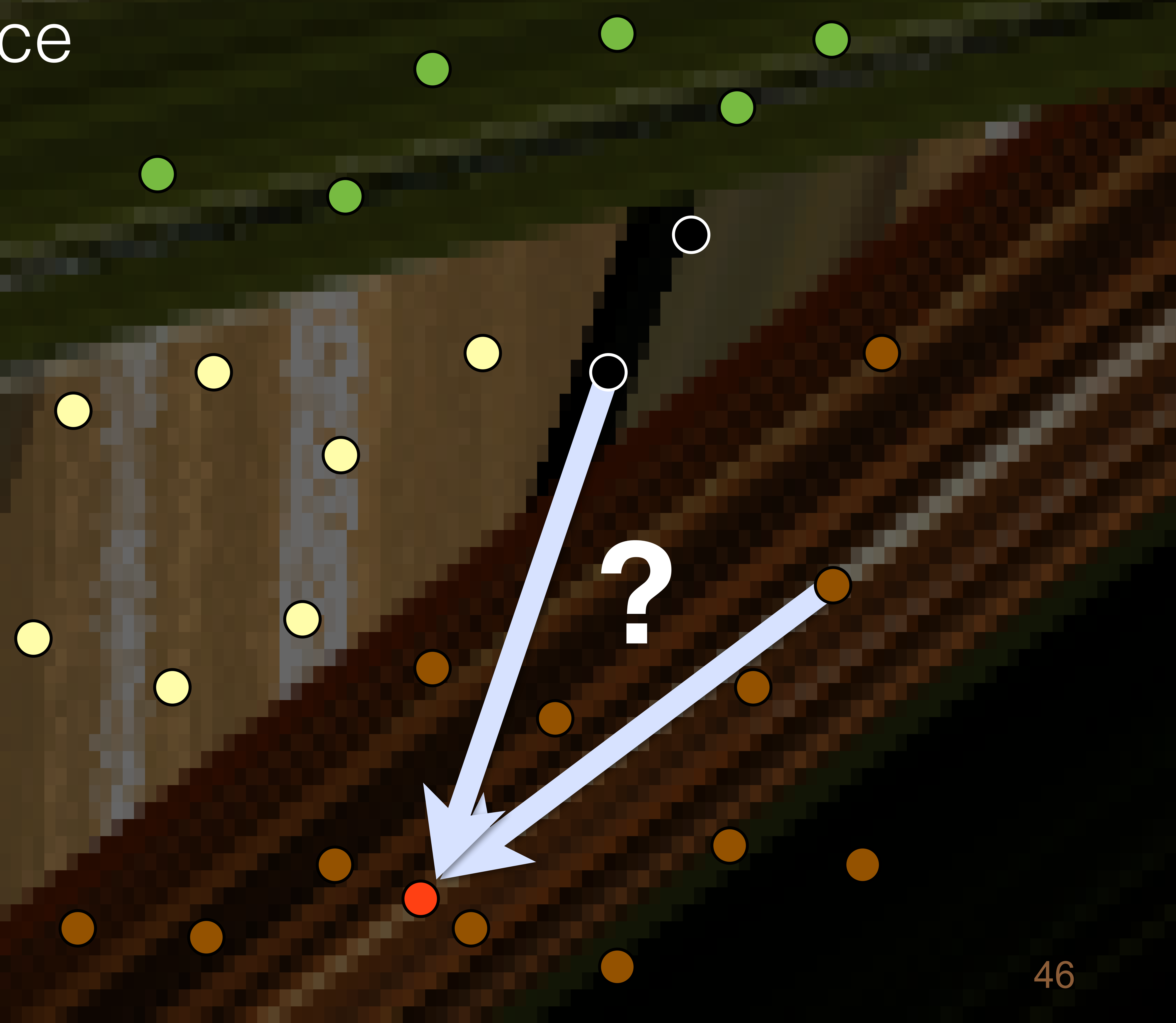
Determining **coverage**:
Does the apparent surface cover my reconstruction location?



Visibility: SameSurface

Input:

sparse **points** with **slopes**



Visibility: SameSurface

The trajectories of samples originating from a single **apparent surface** never intersect.

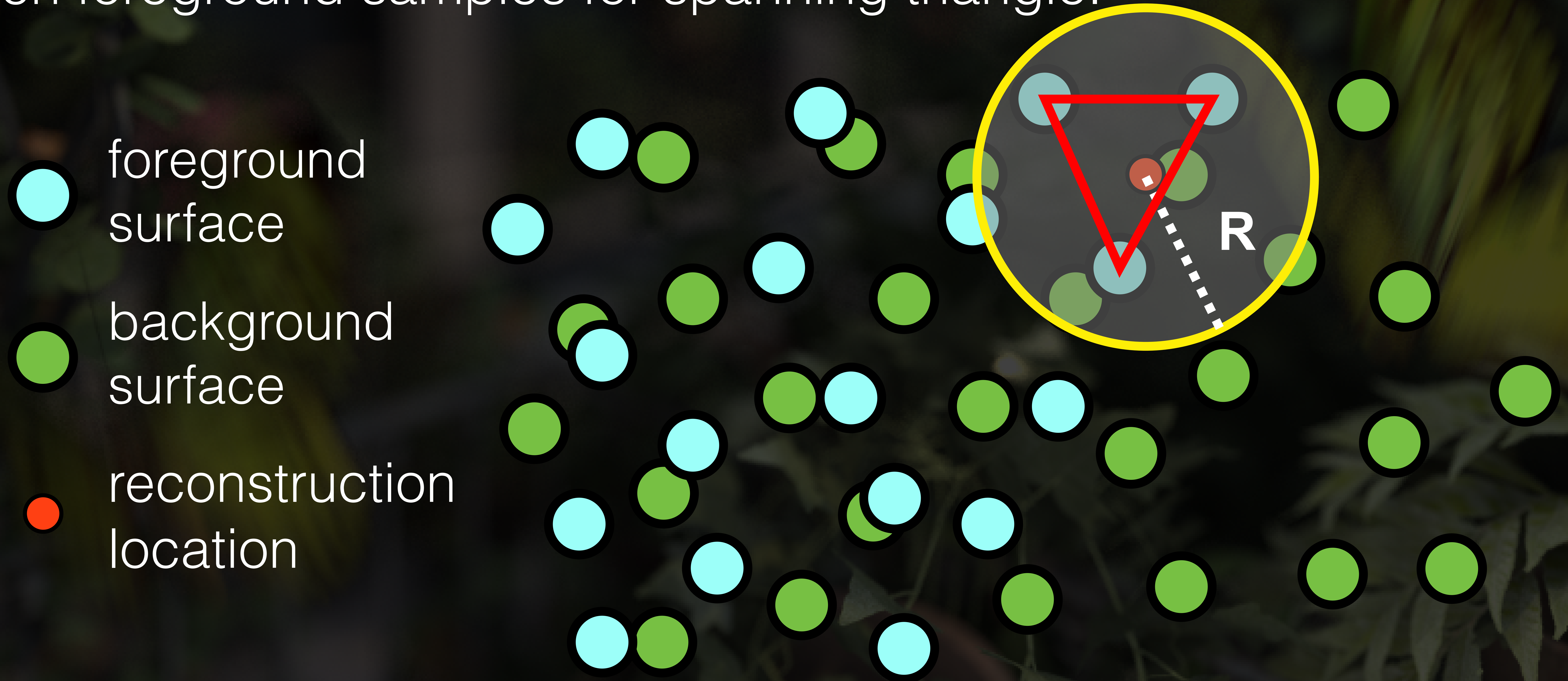
Visibility: SameSurface

Visibility events
show up as **intersections**

Visibility: Coverage

Does foreground apparent surface cover reconstruction location?

Search foreground samples for spanning triangle.



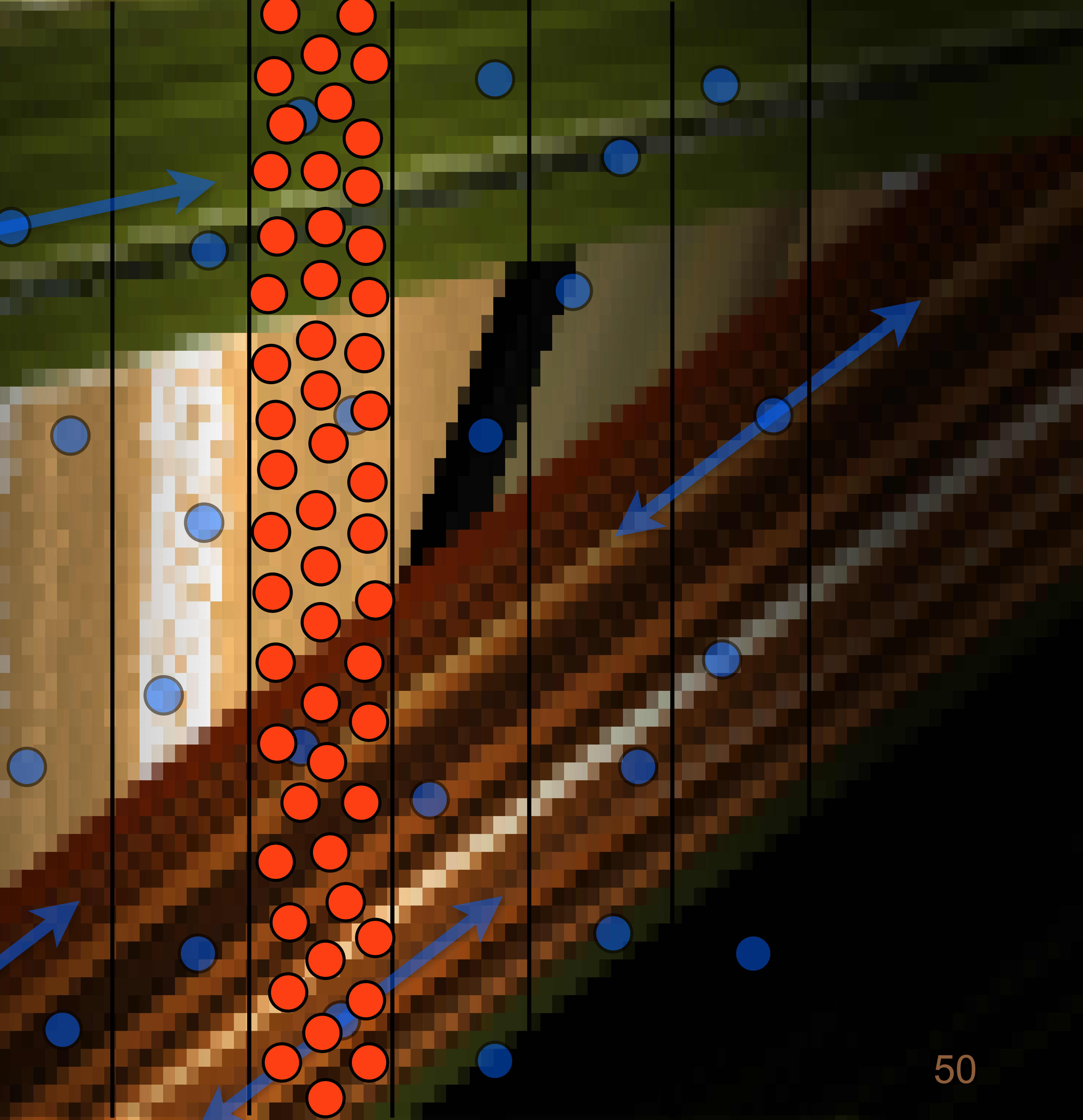
Recap: our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

- Use **slopes** to reproject
- Account for **visibility**

Standard Monte-Carlo integration using dense reconstruction



Observations

We only need sample radiance, depth, and velocity (i.e., **slopes**).
Reconstruction is **independent** of the original renderer.

We can **discard** the scene.

Observations

We only need sample radiance, depth, and velocity (i.e., **slopes**).
Reconstruction is **independent** of the original renderer.

We can **discard** the scene.

Need efficient **sample search**:

Fast motion and large defocus can lead to a single sample contributing to **hundreds** of pixels.

Build a **hierarchy** over input samples.

Extension to soft shadows

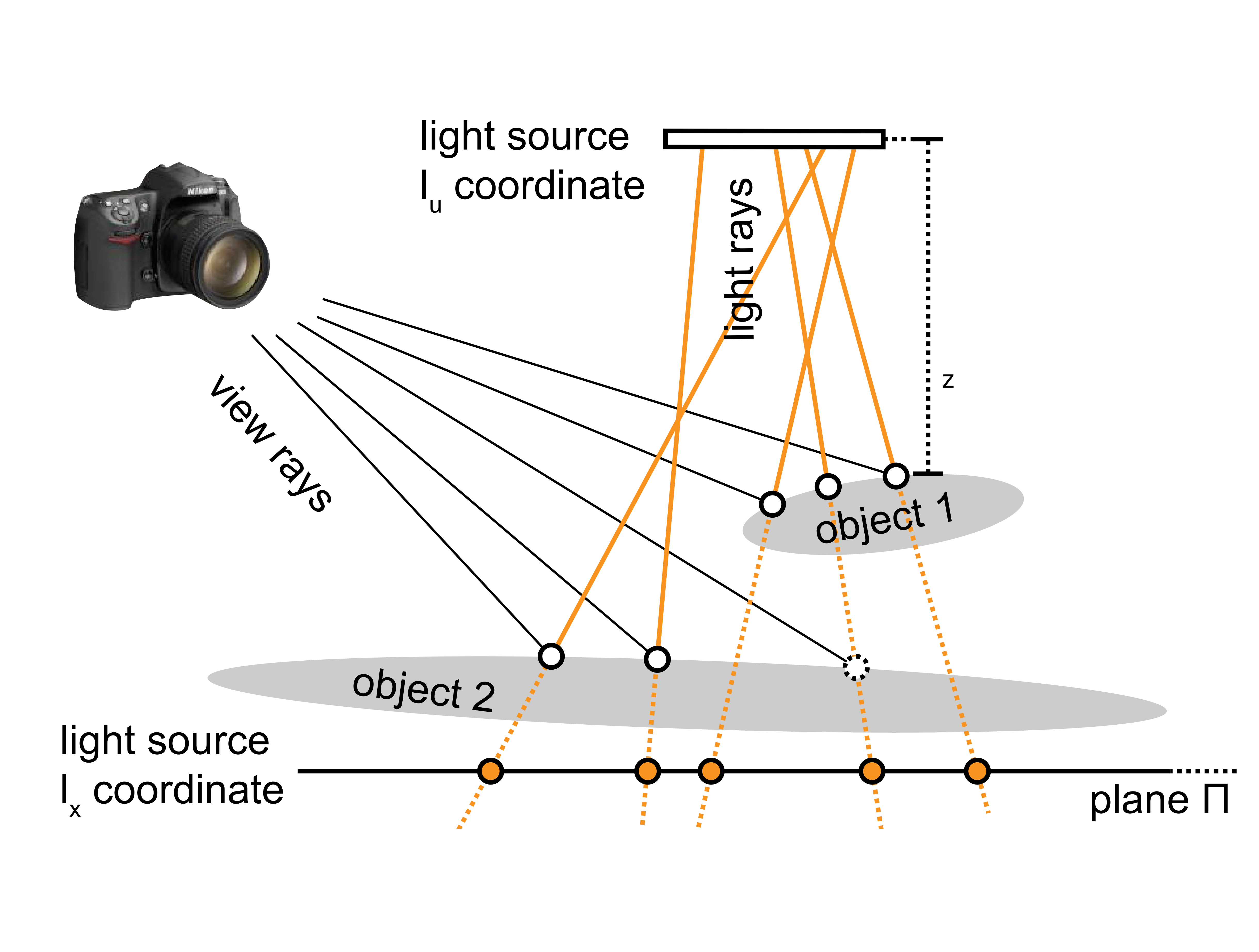
An **area light** is very much like a **lens**.

lens ~ light, sensor ~ virtual plane
Reconstruct **z** instead of radiance

Egan et al. [2010] reconstruct
far field **binary visibility** only.

7D path-tracing style reconstruction
avoiding **combinatorial explosion**

Reconstruct scene point (**5D**)
Reconstruct shadow **z** shade (**2D**)



Results

Implementation

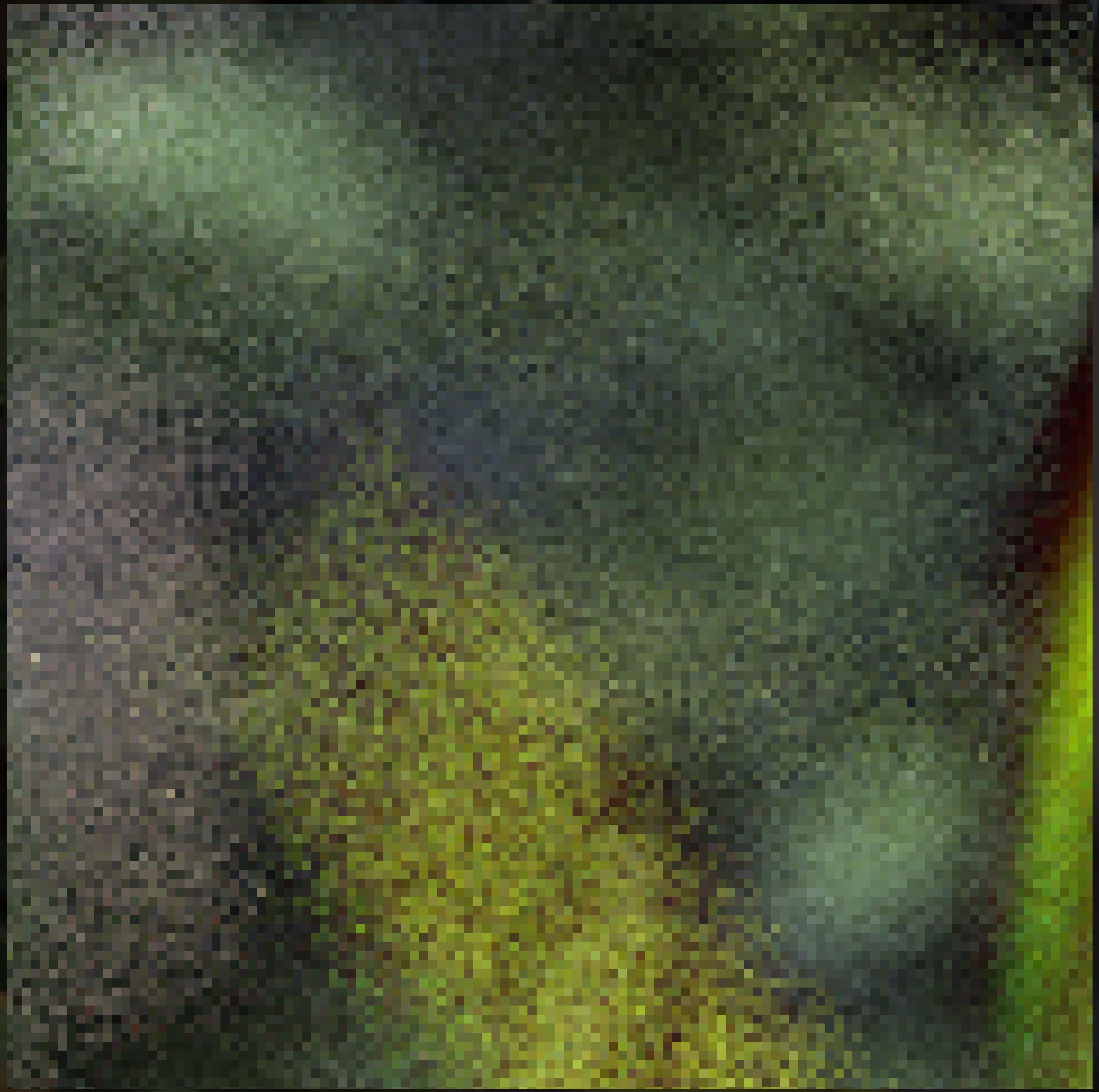
Multithreaded **CPU**
GPU, excluding hierarchy construction

Common sample buffer format accepts outputs from:
- **PBRT**
- **Pixie** (Open source RenderMan)
- Custom ray tracer

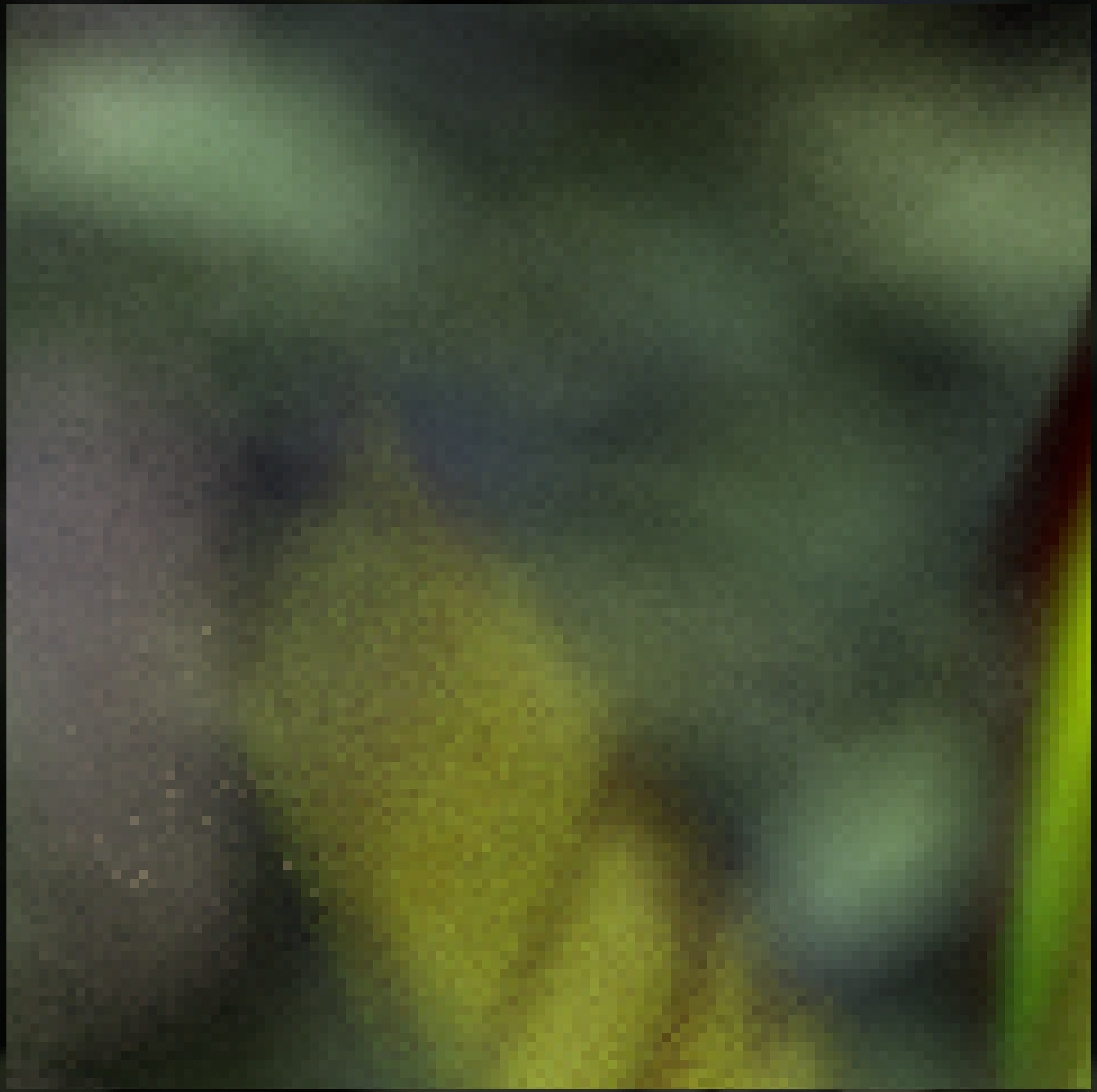
Code will be made available


Input: 16 spp
1072 sec (PBRT)

Our result: 16 spp + reconstruction at 128spp
1072 sec (PBRT) + 10 sec (reconstruction)


Input: 16 spp


Our result at 128 spp
using same input

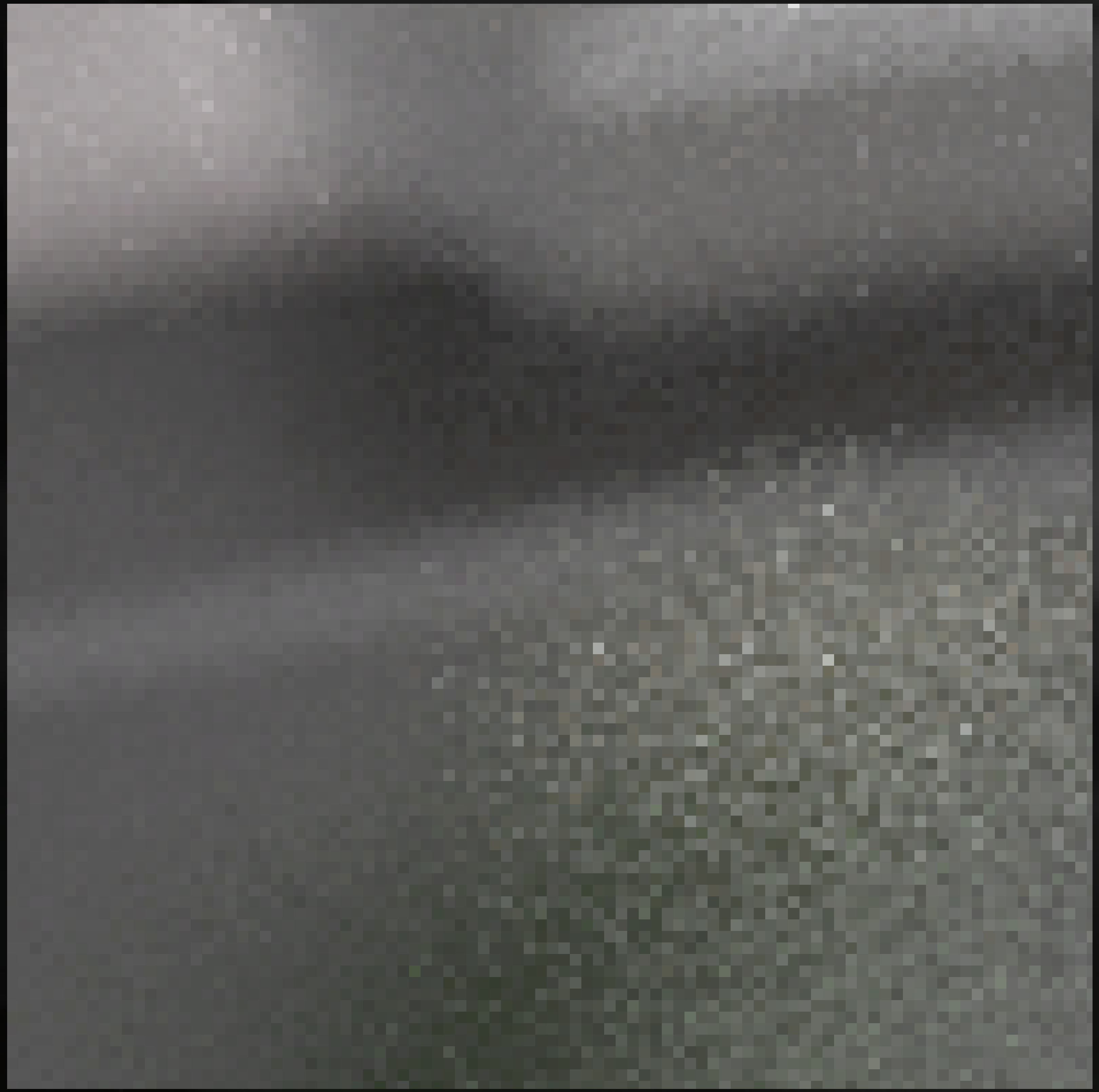

Reference: 256 spp
(16x time)

Our result: 16 spp + reconstruction at 128spp
1072 sec (PBRT) + 10 sec (reconstruction)

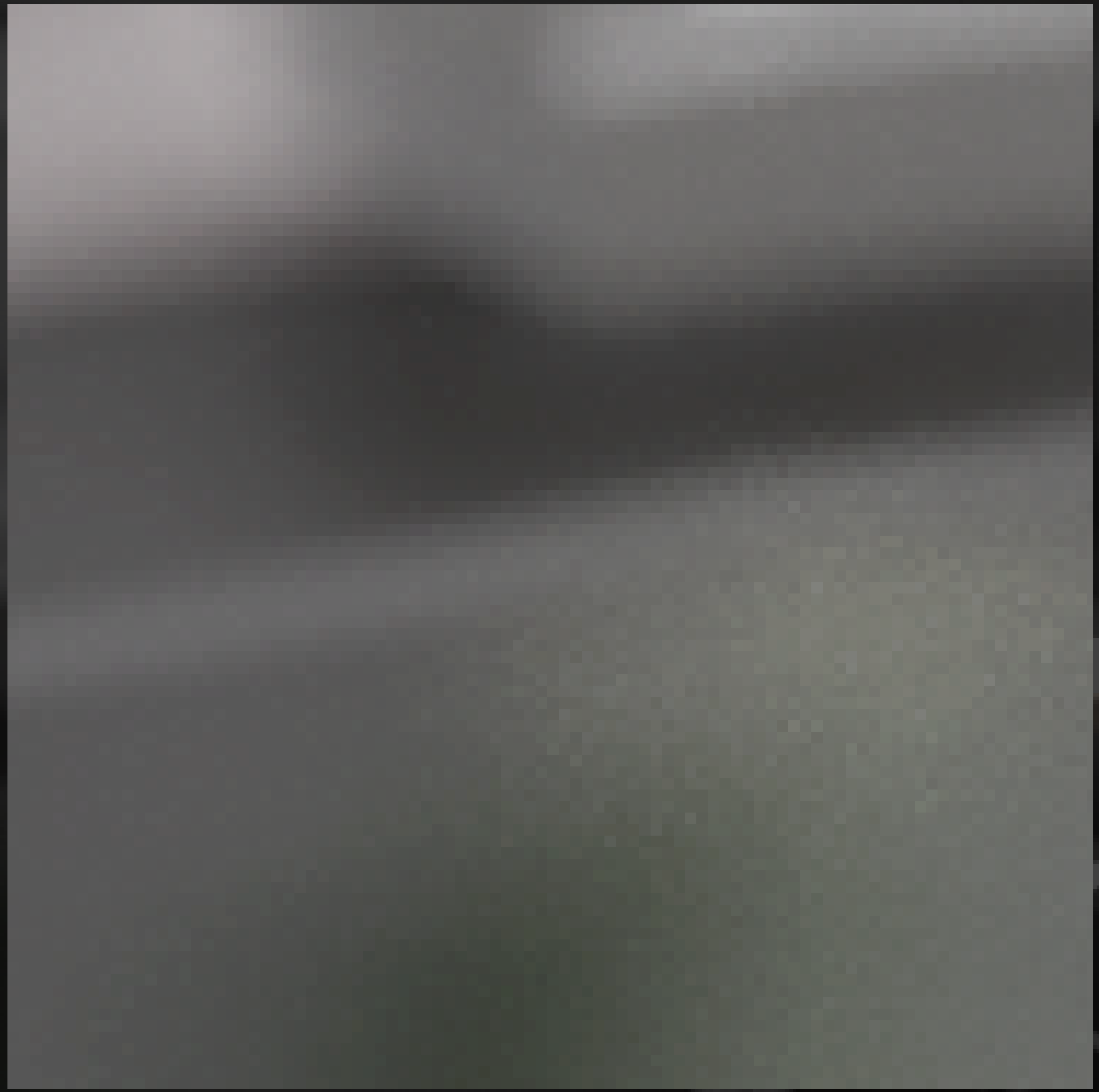

Input: 16 spp
771 sec (PBRT)

Our result: 16 spp + reconstruction at 128spp
771 sec (PBRT) + 10 sec (reconstruction)

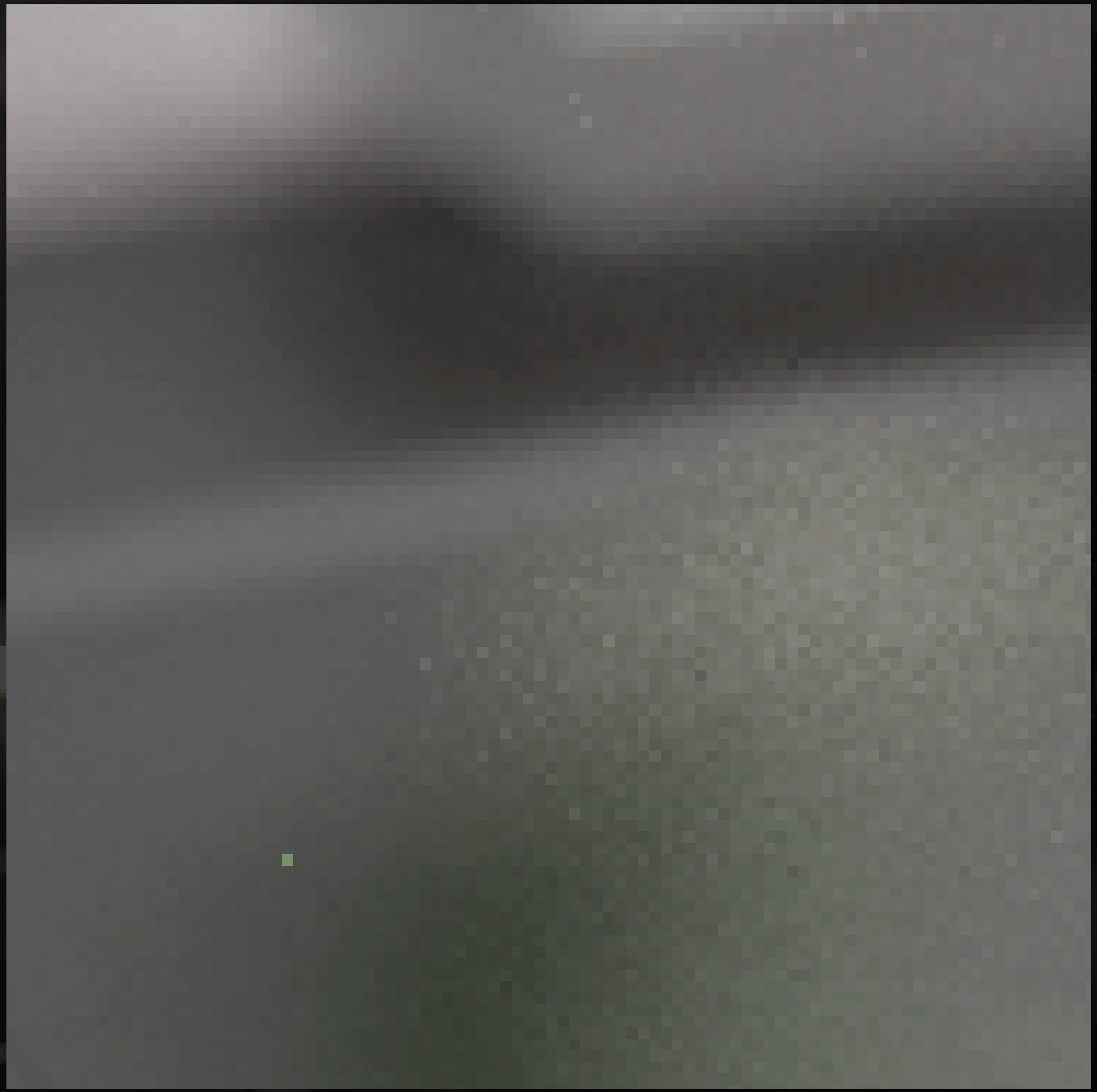
Our result: 16 spp + reconstruction at 128spp
771 sec (PBRT) + 10 sec (reconstruction)



Input: 16 spp

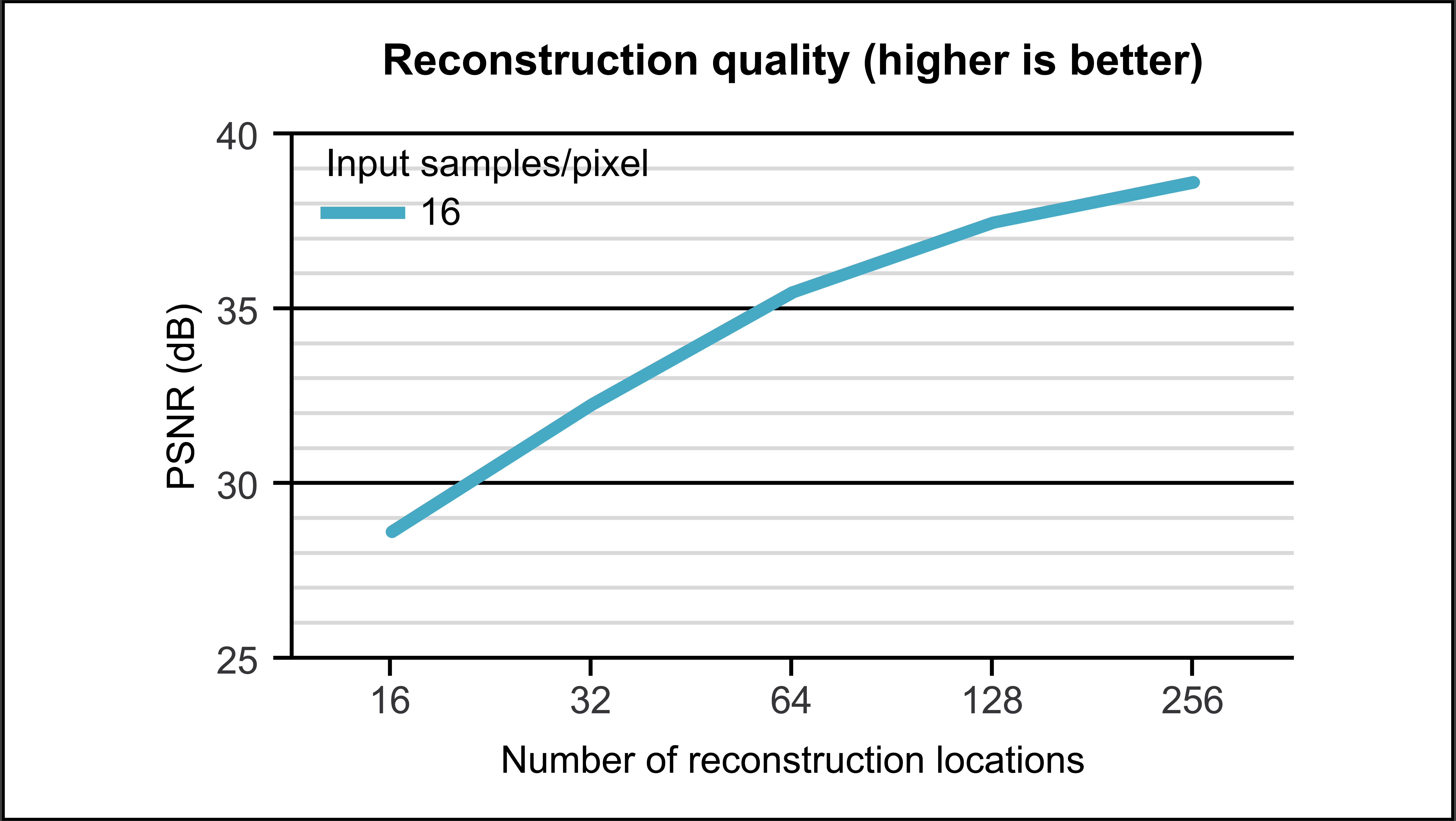


Our result at 128 spp
using same input



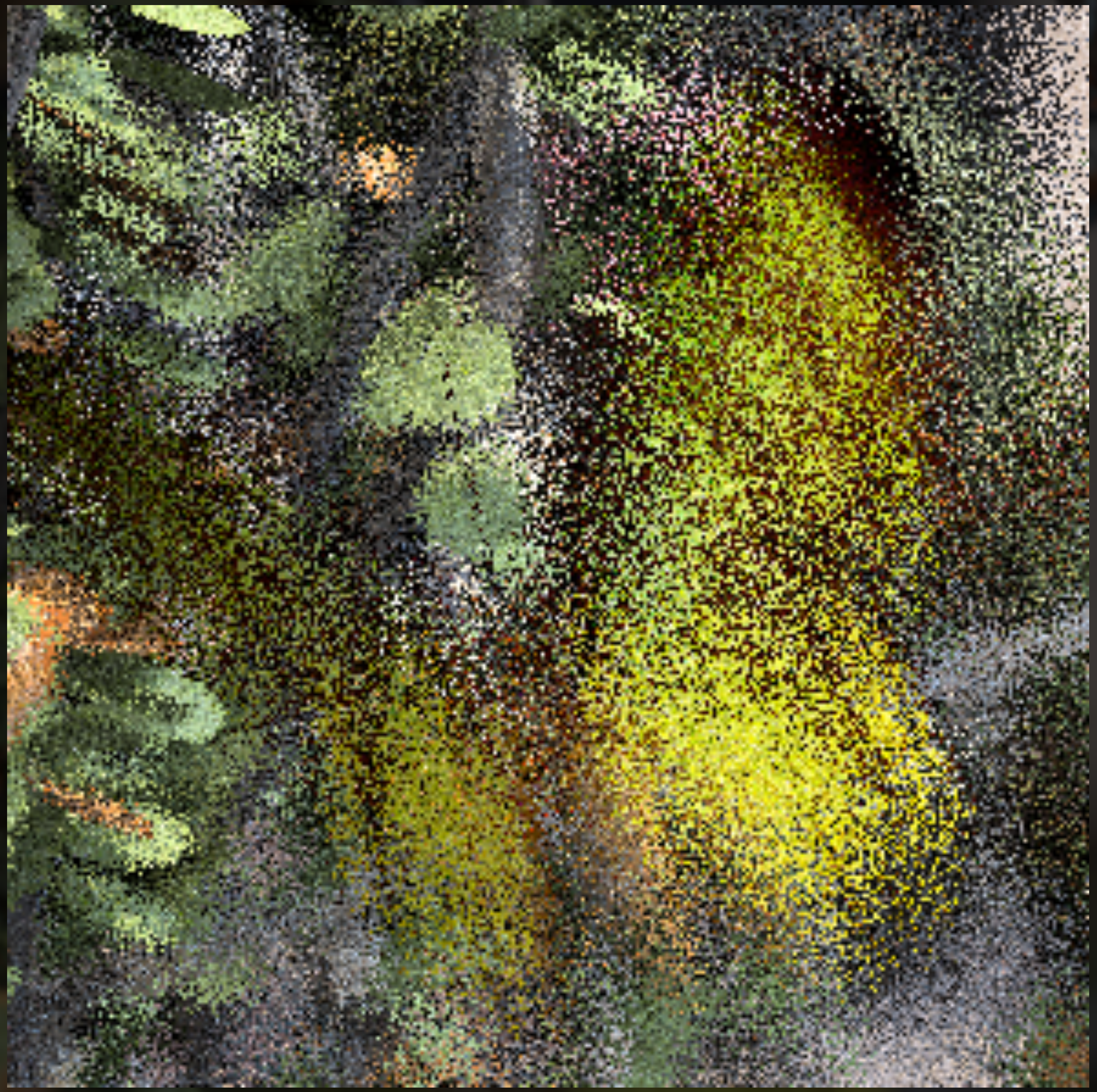
Reference: 256 spp
(16x time)

Comparison to reference




Motion blur and depth of field
1 sample per pixel


Our reconstruction



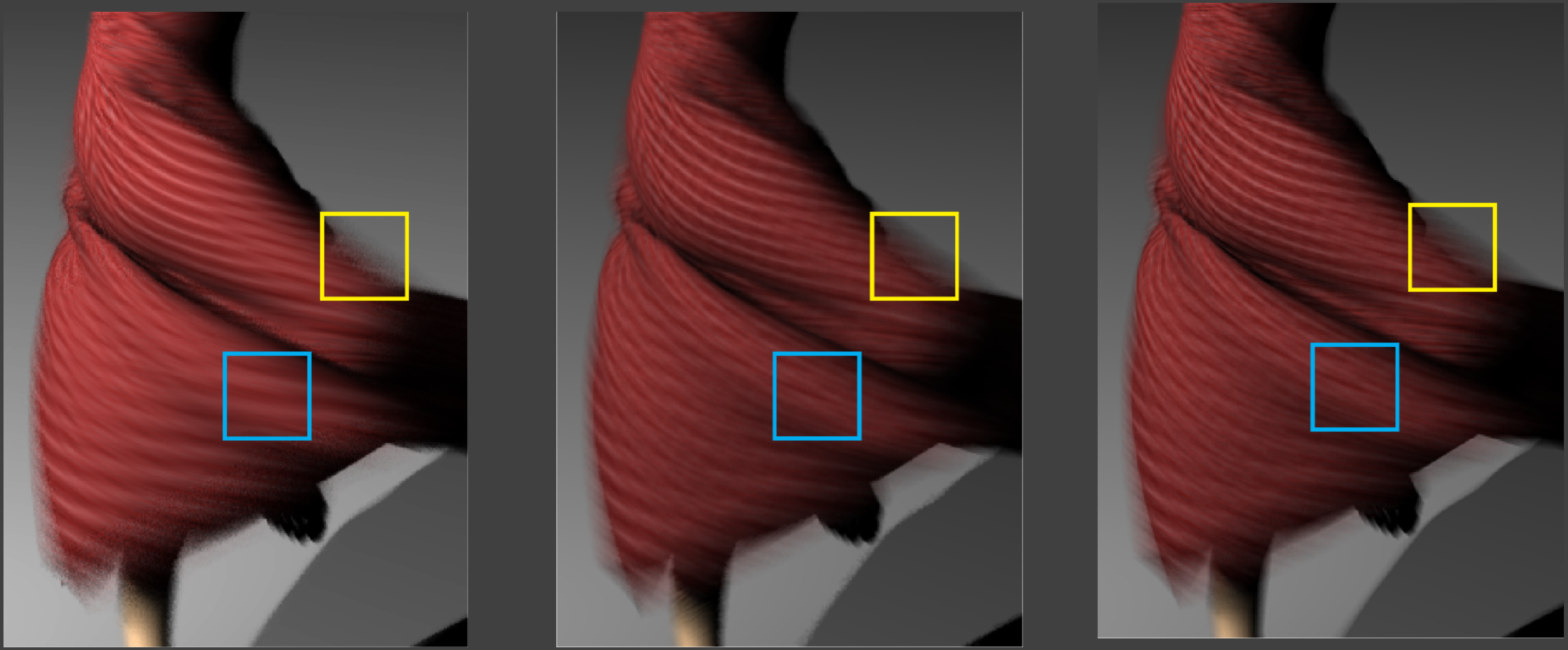
Input: 1 spp

Our result: 1 spp -> 128 spp

Reference 256 spp
(256x time)

Our reconstruction

Comparison to Egan et al. [2009]

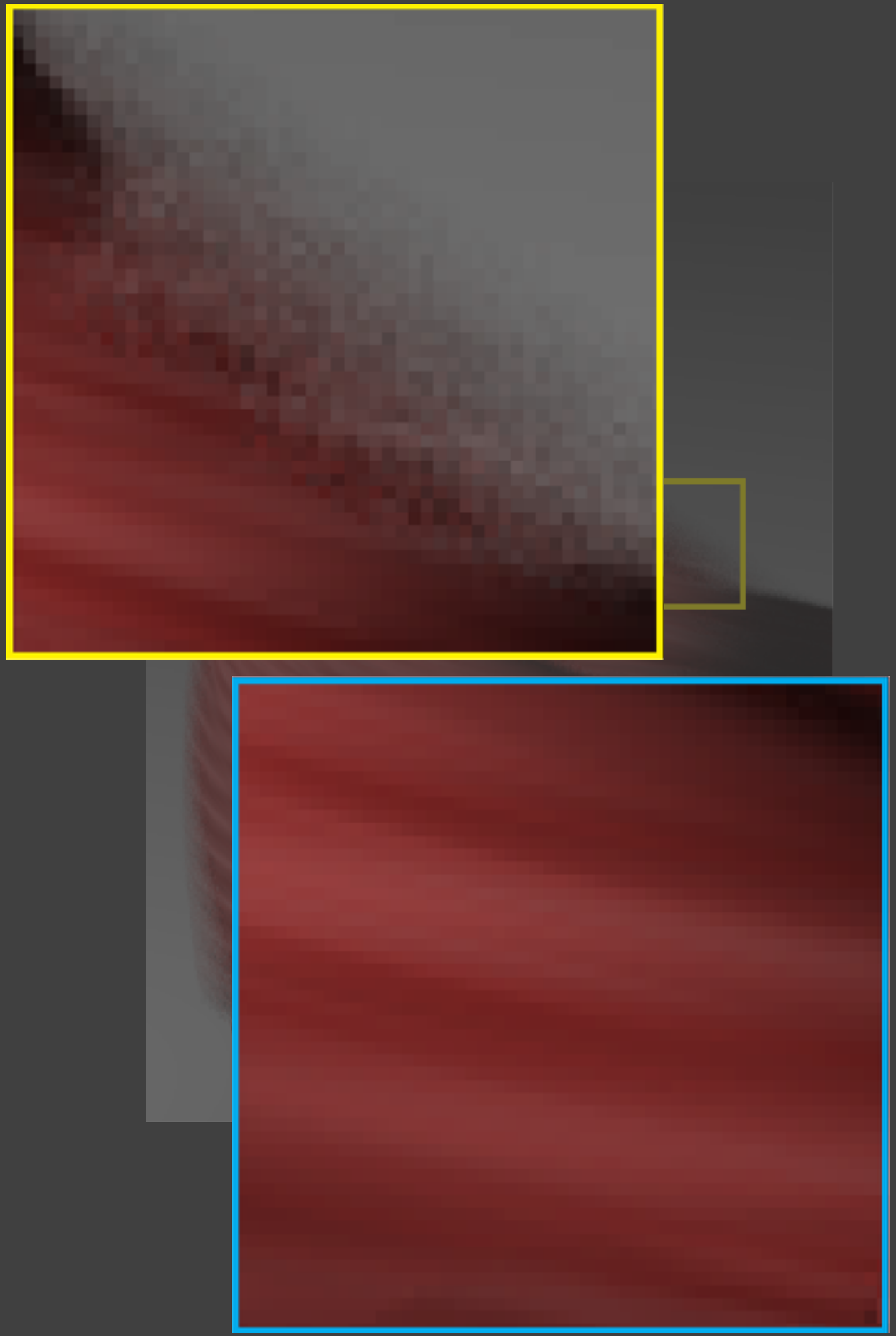


Egan et al. [2009]
8 samples / pixel

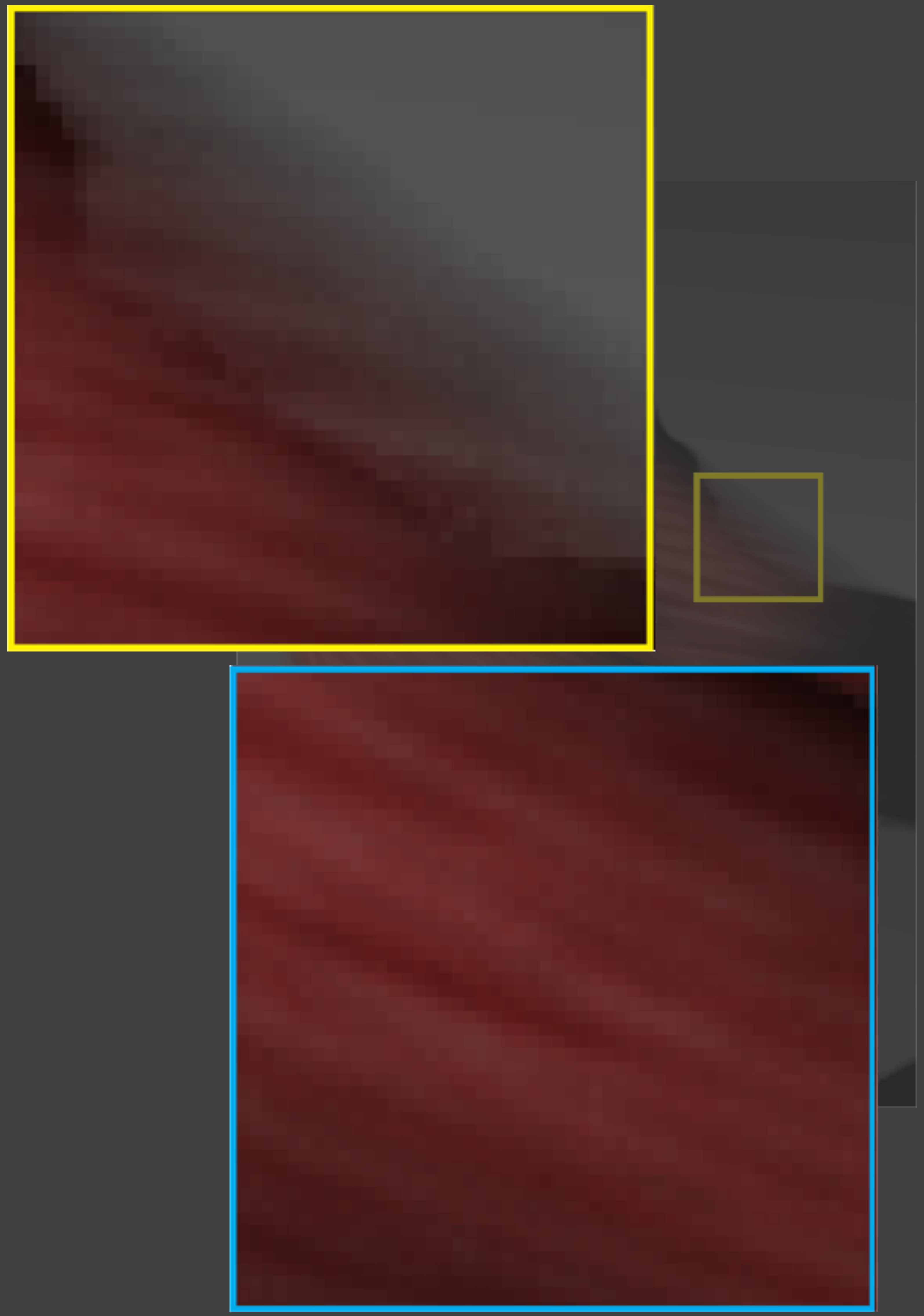
Our method
4 samples / pixel

Reference
256 samples / pixel

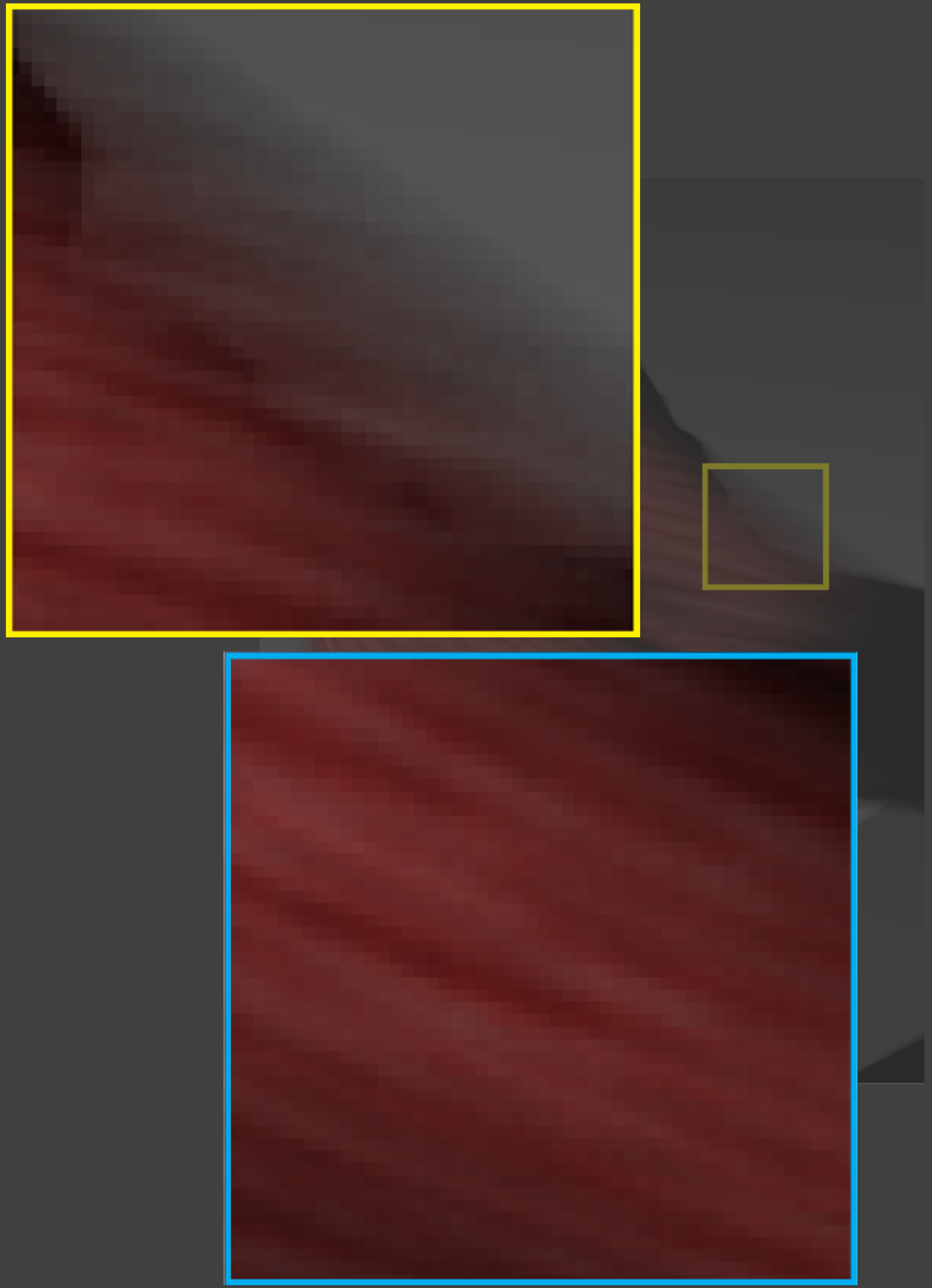
Comparison to Egan et al. [2009]



Egan et al. [2009]
8 samples / pixel



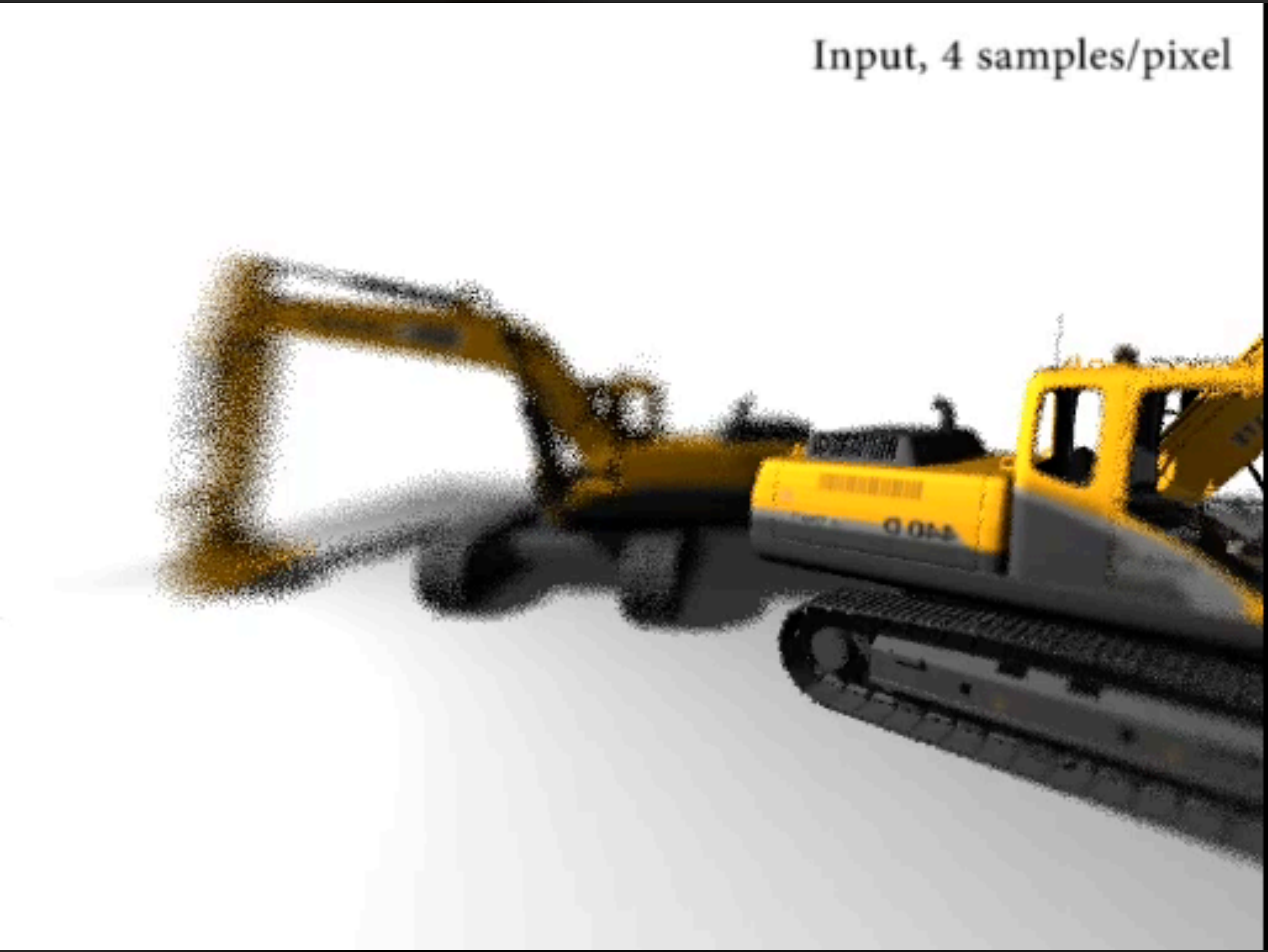
Our method
4 samples / pixel



Reference
256 samples / pixel

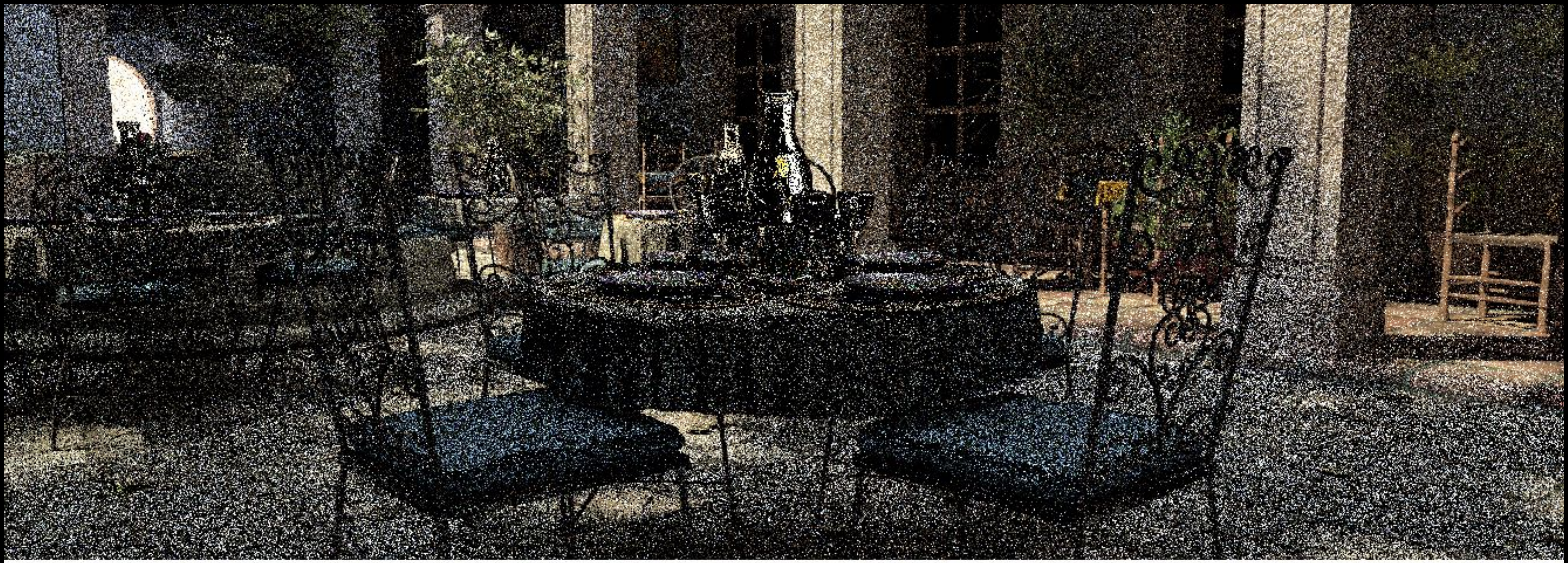
Soft shadows, 4 spp

7D soft shadows with motion and defocus, 4 spp



Filtering Monte Carlo Noise From Random Parameters

Sen and Darabi [2012]



input Monte Carlo (8 samples/pixel)

after RPF (8 samples/pixel)

High-dimensional Monte Carlo Integration

$$I(i,j) = \int_{i-\frac{1}{2}}^{i+\frac{1}{2}} \int_{j-\frac{1}{2}}^{j+\frac{1}{2}} \cdots \int_{-1}^{1} \int_{-1}^{1} \int_{t_0}^{t_1} f(x, y, \cdots, u, v, t)\, dt\, dv\, du \cdots dy\, dx$$

(a) Input MC (8 spp) **(b)** Dependency on (u, v) **(c)** Our approach (RPF)

Parameters in Monte Carlo estimator

Random parameters: $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$

Color: $\mathbf{c}_i \Leftarrow f(\underbrace{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}}_{\text{screen position}}; \underbrace{\mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,n}}_{\text{random parameters}})$

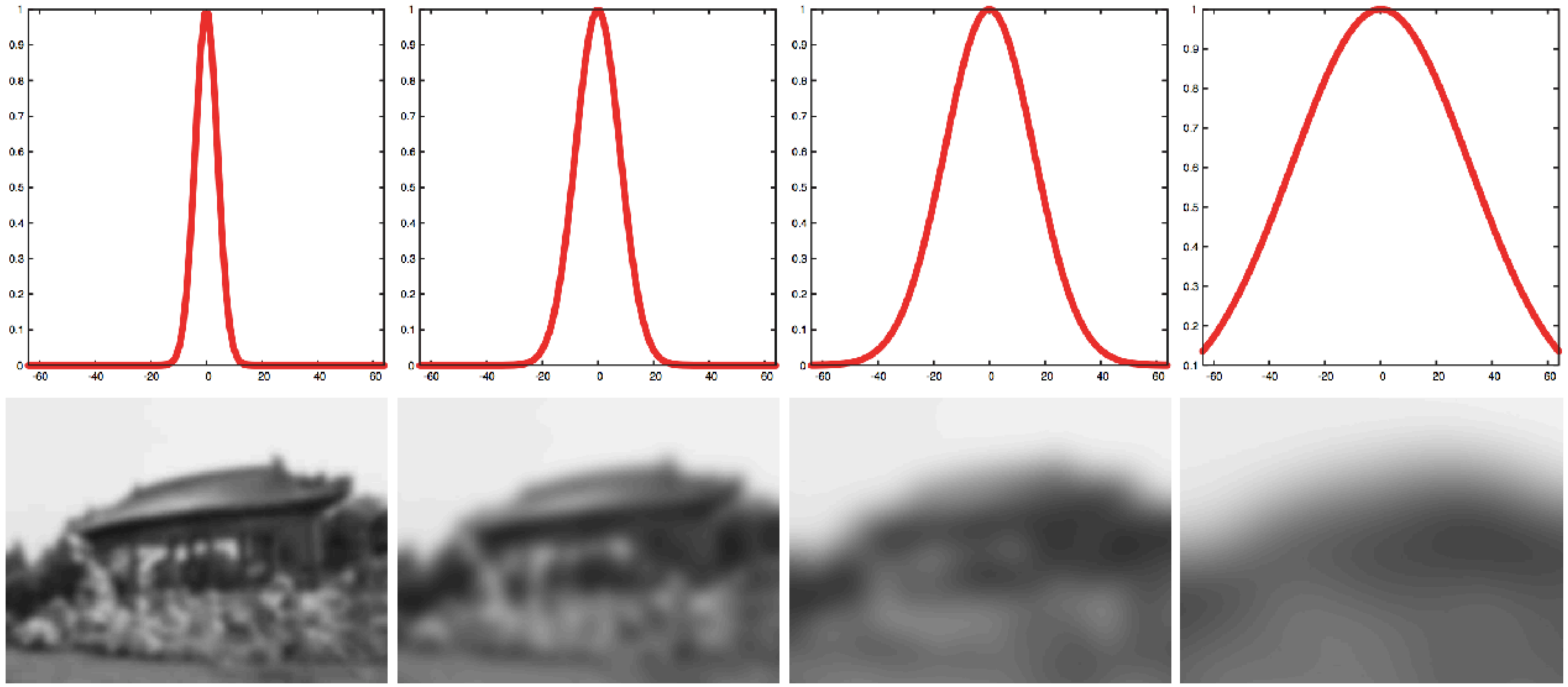
Random Parameters Classification

Random parameter for each pixel :

$$\mathbf{x}_i \Leftarrow f(\mathbf{p}_{i,1}, \mathbf{p}_{i,2}; \mathbf{r}_{i,1}, \mathbf{r}_{i,2}, \ldots, \mathbf{r}_{i,n})$$

$$\mathbf{x}_i = \{\underbrace{\boxed{\mathbf{p}_{i,1}, \mathbf{p}_{i,2}}}_{\text{screen position}}; \underbrace{\boxed{\mathbf{r}_{i,1}, \ldots, \mathbf{r}_{i,n}}}_{\text{random parameters}}; \underbrace{\boxed{\mathbf{f}_{i,1}, \ldots, \mathbf{f}_{i,m}}}_{\text{scene features}}; \underbrace{\boxed{\mathbf{c}_{i,1}, \mathbf{c}_{i,2}, \mathbf{c}_{i,3}}}_{\text{sample color}}\}$$

Gaussian Filtering



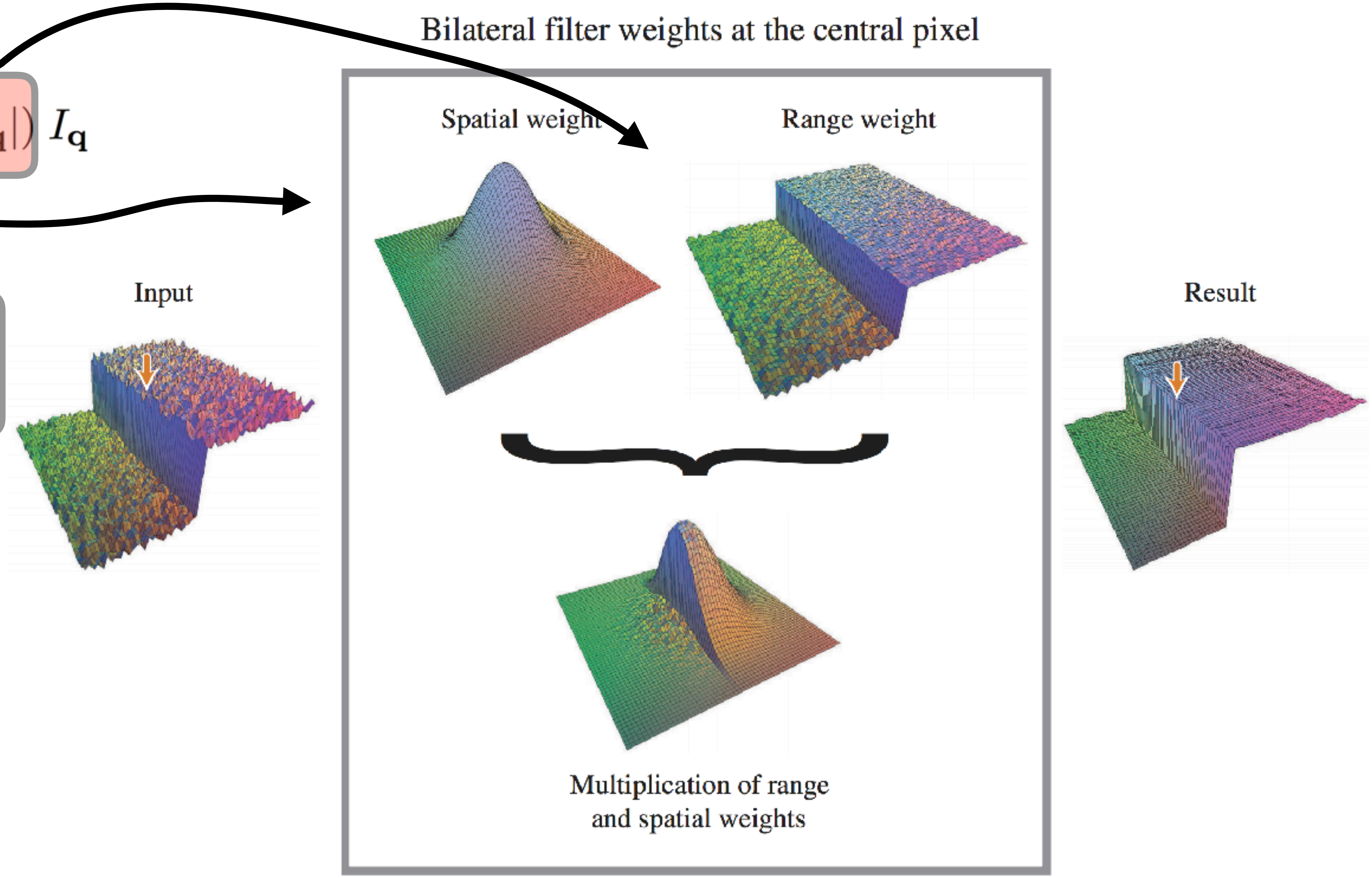
$\sigma = 4$ $\sigma = 8$ $\sigma = 16$ $\sigma = 32$

$$GC[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in \mathcal{S}} G_\sigma(\|\mathbf{p} - \mathbf{q}\|)\, I_{\mathbf{q}}, \qquad G_\sigma(x) = \frac{1}{2\pi\,\sigma^2} \exp\left(-\frac{x^2}{2\,\sigma^2}\right)$$

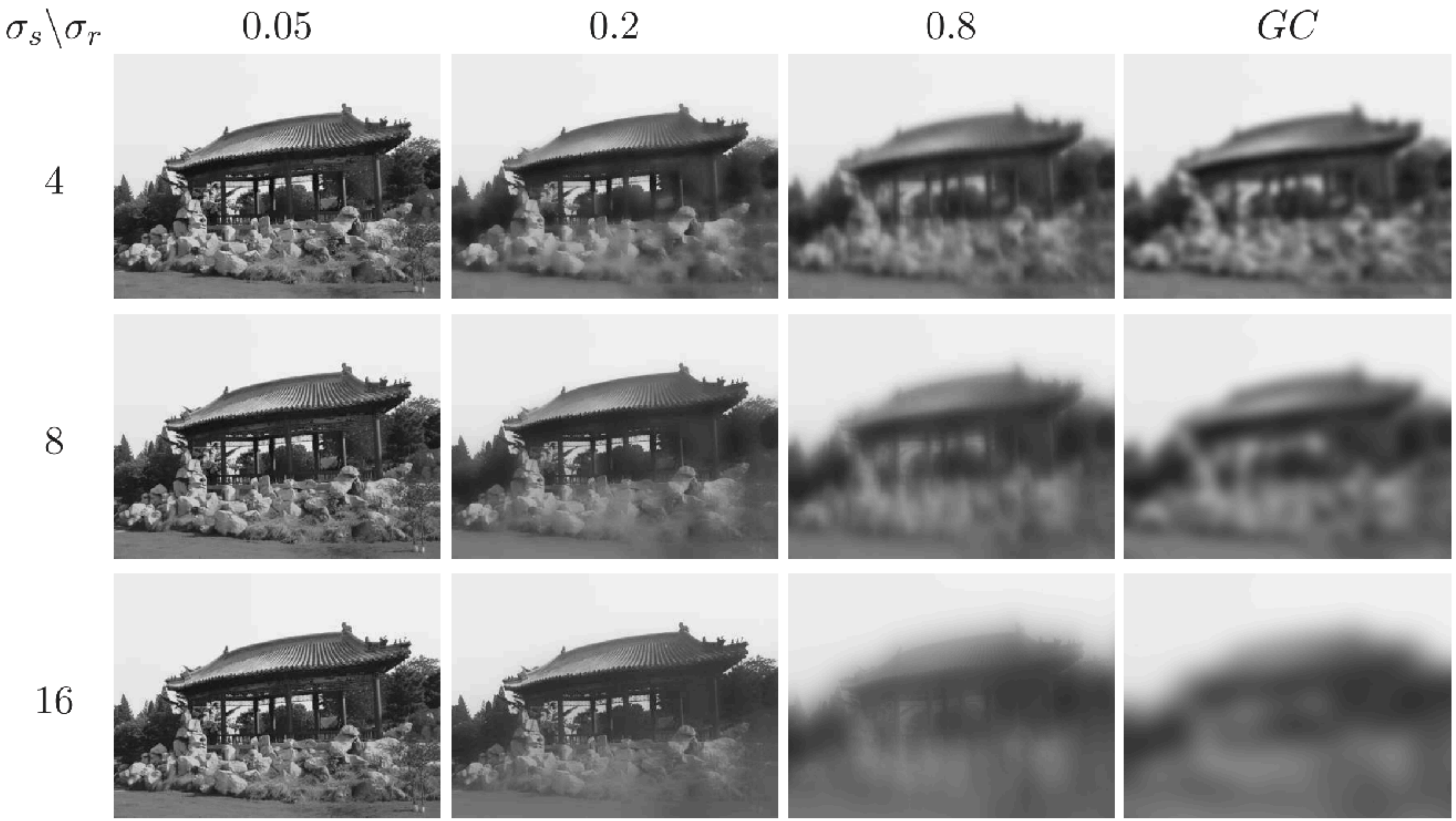
Bilateral Filtering

$$BF[I]_{\mathbf{p}} = \frac{1}{\boxed{W_{\mathbf{p}}}} \sum_{\mathbf{q}\in\mathcal{S}} \boxed{G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)}\, \boxed{G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}\, I_{\mathbf{q}}$$

$$\boxed{W_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}$$



Bilateral vs Gaussian Filtering



Bilateral Filtering of Features

$$w_{ij} = \exp[-\frac{1}{2\sigma_{\mathbf{p}}^2} \sum_{1 \le k \le 2} (\bar{\mathbf{p}}_{i,k} - \bar{\mathbf{p}}_{j,k})^2] \times$$

$$\exp[-\frac{1}{2\sigma_{\mathbf{c}}^2} \sum_{1 \le k \le 3} \boxed{\alpha_k} (\bar{\mathbf{c}}_{i,k} - \bar{\mathbf{c}}_{j,k})^2] \times$$

$$\exp[-\frac{1}{2\sigma_{\mathbf{f}}^2} \sum_{1 \le k \le m} \boxed{\beta_k} (\bar{\mathbf{f}}_{i,k} - \bar{\mathbf{f}}_{j,k})^2],$$

Bilateral Weights

$$W^{\mathbf{r}}_{\mathbf{f},k} = \frac{D^{\mathbf{r}}_{\mathbf{f},k}}{D^{\mathbf{r}}_{\mathbf{f},k} + D^{\mathbf{p}}_{\mathbf{f},k}}$$

$$\boxed{\beta_k} = 1 - W^{\mathbf{r}}_{\mathbf{f},k}$$

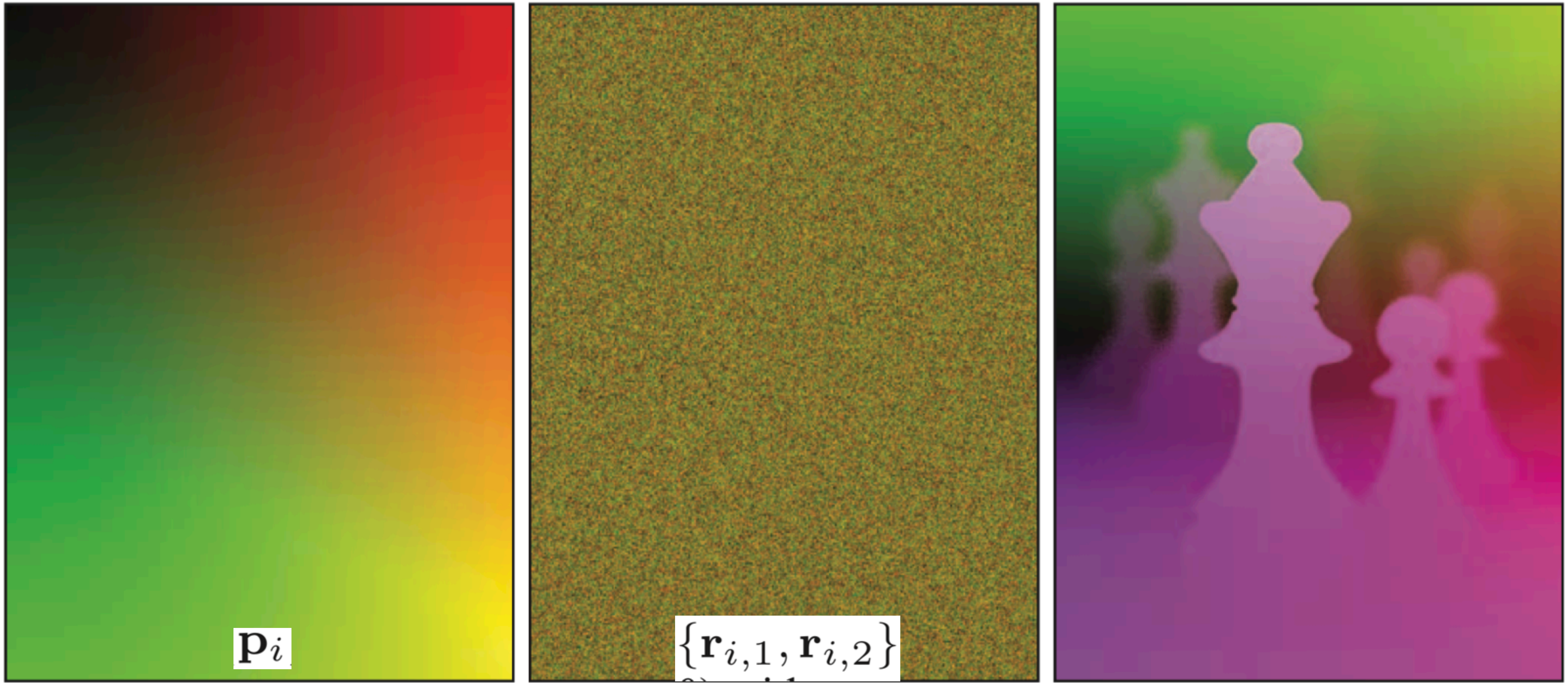
$$W^{\mathbf{r}}_{\mathbf{c},k} = \frac{D^{\mathbf{r}}_{\mathbf{c},k}}{D^{\mathbf{r}}_{\mathbf{c},k} + D^{\mathbf{p}}_{\mathbf{c},k}}$$

$$\boxed{\alpha_k} = 1 - W^{\mathbf{r}}_{\mathbf{c},k}.$$

Dependency on Random Parameters

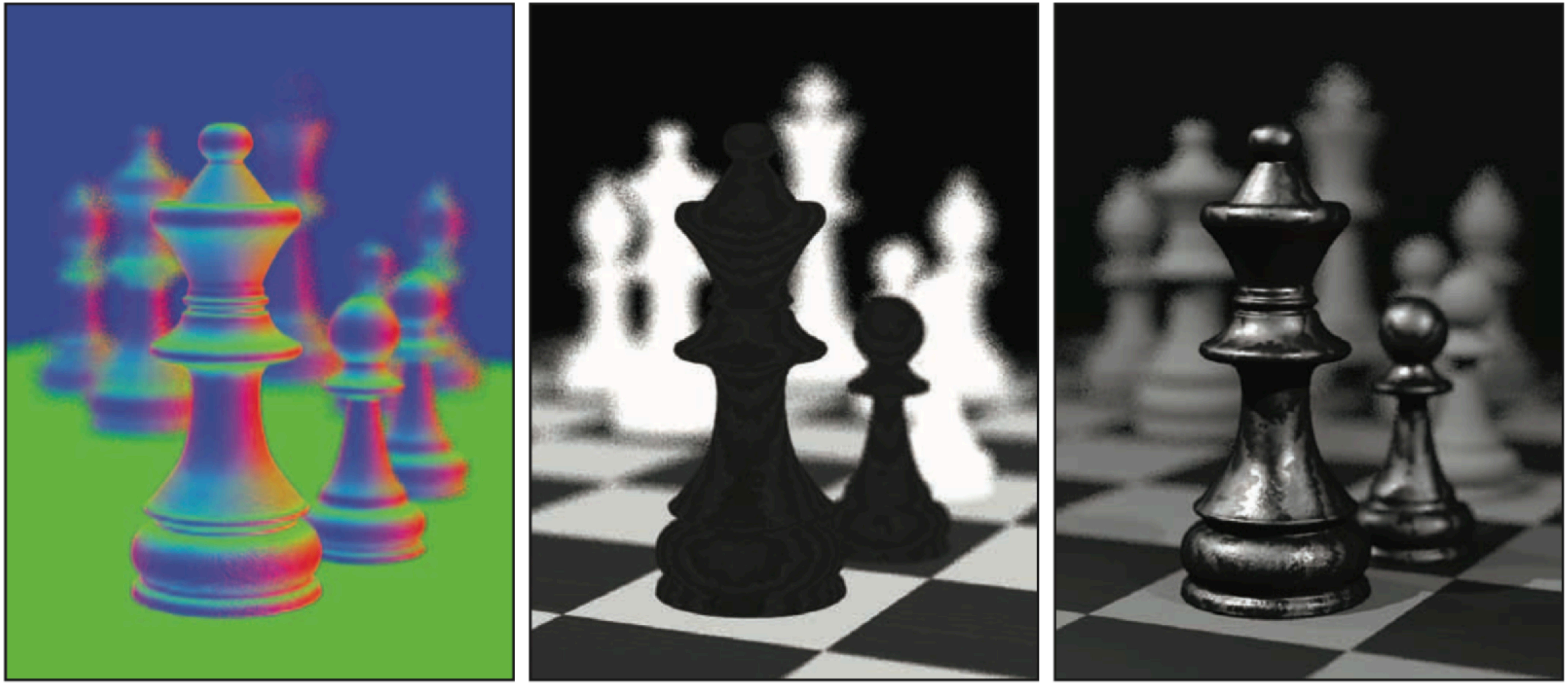
Input Monte Carlo (8 spp) | Dependency of color on random parameters ($D^{\mathbf{r}}_{\mathbf{c}}$) | Dependency of color on screen position ($D^{\mathbf{p}}_{\mathbf{c}}$) | Fractional dependency on random parameters ($W^{\mathbf{r}}_{\mathbf{c}}$) | Our approach (RPF) | Reference MC (512 spp)

Pixels,Random Params,Features



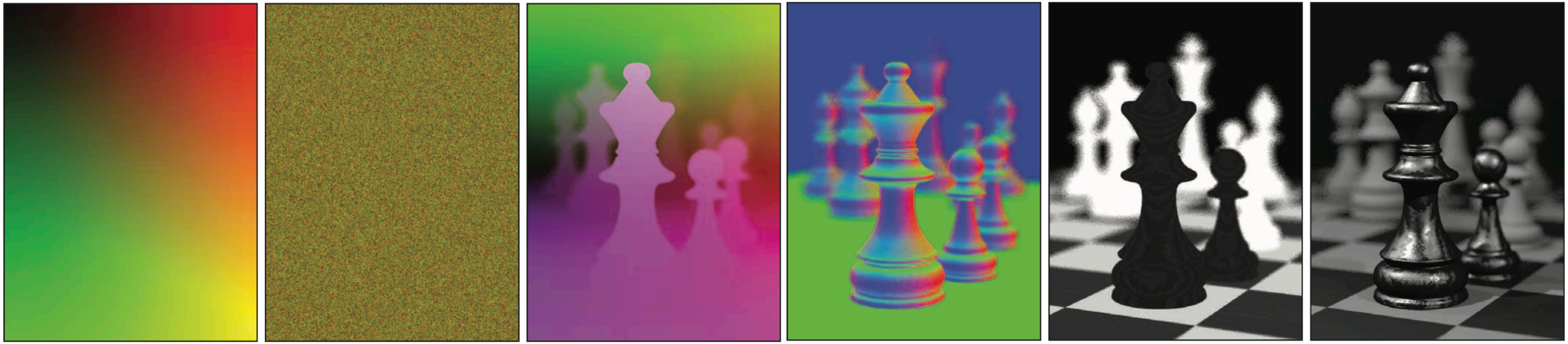
(a) Screen position (b) Random parameters (c) World space coords.

Pixels,Random Params,Features



(d) Surface normals (e) Texture value (f) Sample color

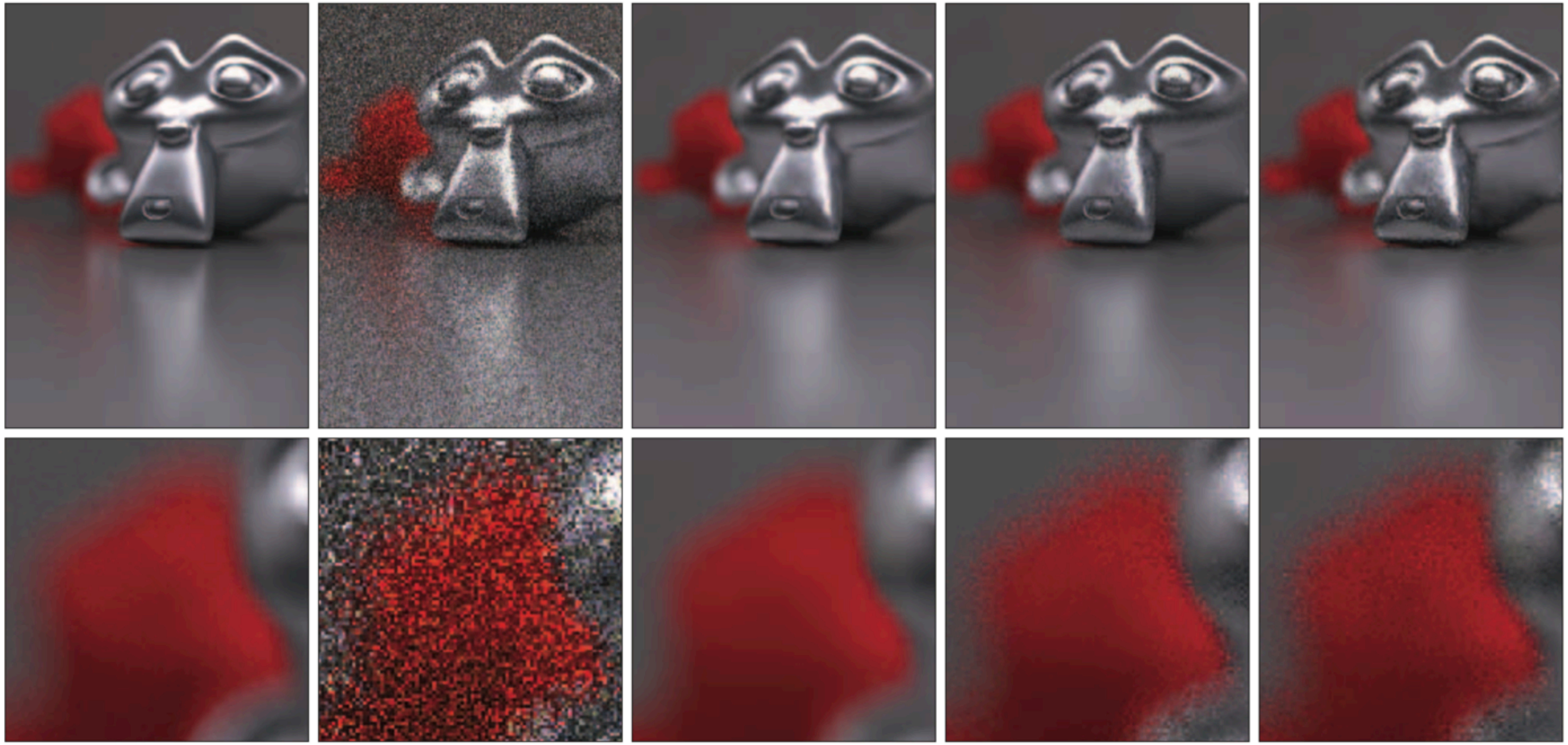
Pixels,Random Params,Features



(a) Screen position (b) Random parameters (c) World space coords. (d) Surface normals (e) Texture value (f) Sample color

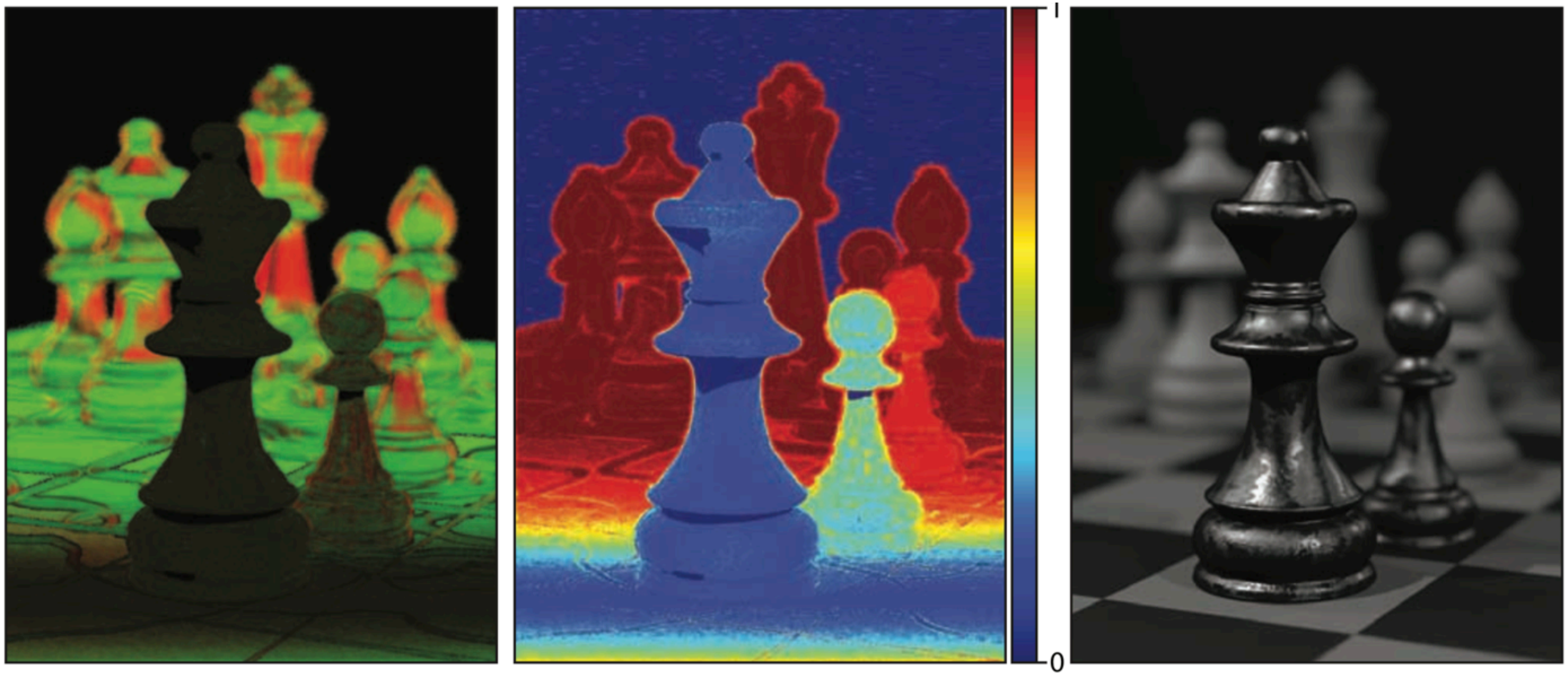
The algorithm computes the statistical dependency of **(c-f)** on the random parameters in (**b**)

Random Parameter Filtering



(a) Reference (b) MC Input (c) RPF (d) no clustering (e) no DoF params

Random Parameter Filtering



(a) $W^{\mathbf{r},1}_{\mathbf{c},k}$ and $W^{\mathbf{r},2}_{\mathbf{c},k}$ (b) $W^{\mathbf{r}}_{\mathbf{c},k}$ (c) Our output (RPF)

Statistical Dependency

Mutual information between two random variables:

$$\mu(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

where, these probabilities are computed over
the neighborhood of samples around a given pixel

Statistical Dependency

Functional dependency of the k-th scene parameter:

$$D^{\mathbf{r}}_{\mathbf{f},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{f},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l})$$

$$D^{\mathbf{p}}_{\mathbf{f},k} = \sum_{1 \le l \le 2} D^{\mathbf{p},l}_{\mathbf{f},k} = \sum_{1 \le l \le 2} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{p}}_{\mathcal{N},l}),$$

$$D^{\mathbf{r}}_{\mathbf{c},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{c},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{c}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l}),$$

$$D^{\mathbf{p}}_{\mathbf{c},k} = \sum_{1 \le l \le 2} D^{\mathbf{p},l}_{\mathbf{c},k} = \sum_{1 \le l \le 2} \mu(\bar{\mathbf{c}}_{\mathcal{N},k}; \bar{\mathbf{p}}_{\mathcal{N},l}).$$

Statistical Dependency

$$D^{\mathbf{r}}_{\mathbf{f},k} = \sum_{1 \le l \le n} D^{\mathbf{r},l}_{\mathbf{f},k} = \sum_{1 \le l \le n} \mu(\bar{\mathbf{f}}_{\mathcal{N},k}; \bar{\mathbf{r}}_{\mathcal{N},l})$$

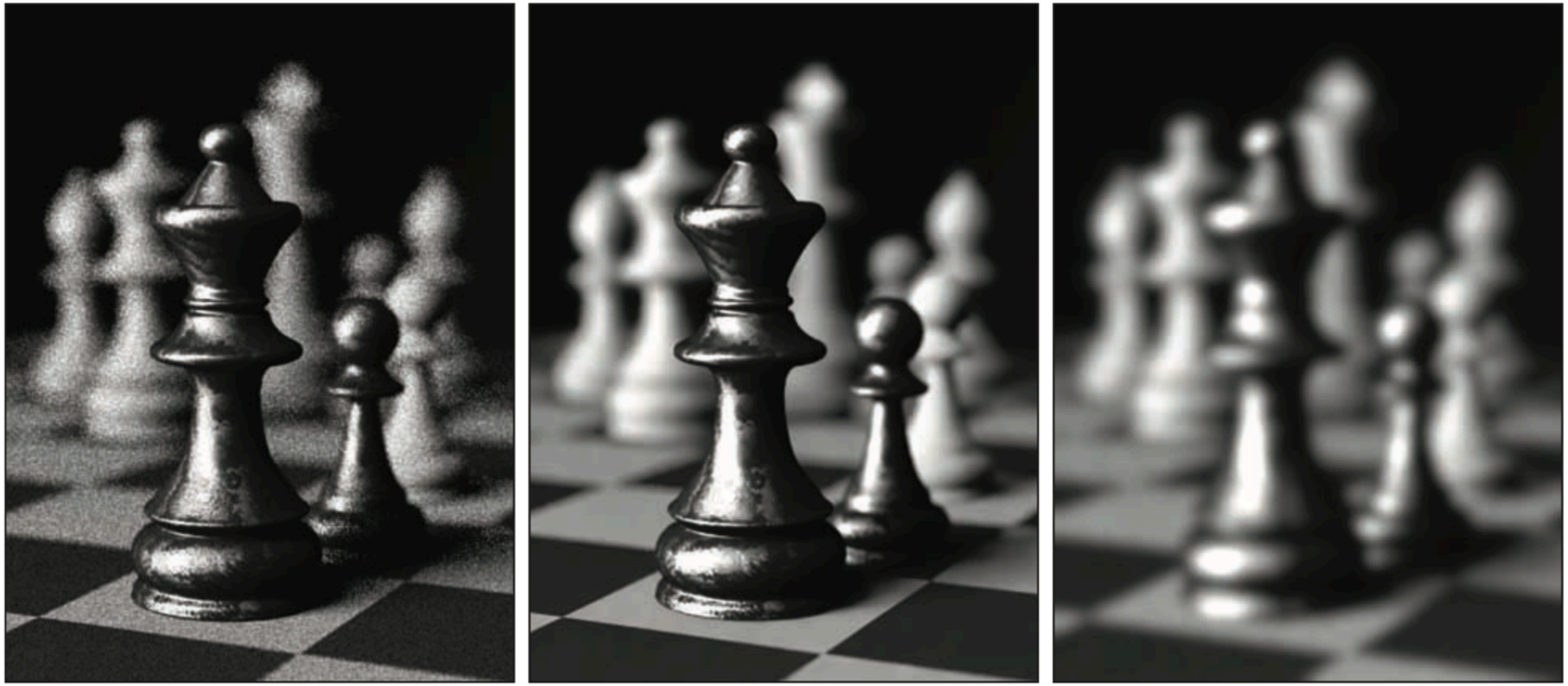
$$W^{\mathbf{f},k}_{\mathbf{c}} = \frac{D^{\mathbf{f},k}_{\mathbf{c}}}{D^{\mathbf{r}}_{\mathbf{c}} + D^{\mathbf{p}}_{\mathbf{c}} + D^{\mathbf{f}}_{\mathbf{c}}}$$

$$D^{\mathbf{r}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{r}}_{\mathbf{c},k}, \quad D^{\mathbf{p}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{p}}_{\mathbf{c},k}, \quad D^{\mathbf{f}}_{\mathbf{c}} = \sum_{1 \le k \le 3} D^{\mathbf{f}}_{\mathbf{c},k}$$

Weighted Average Bilateral Filtering

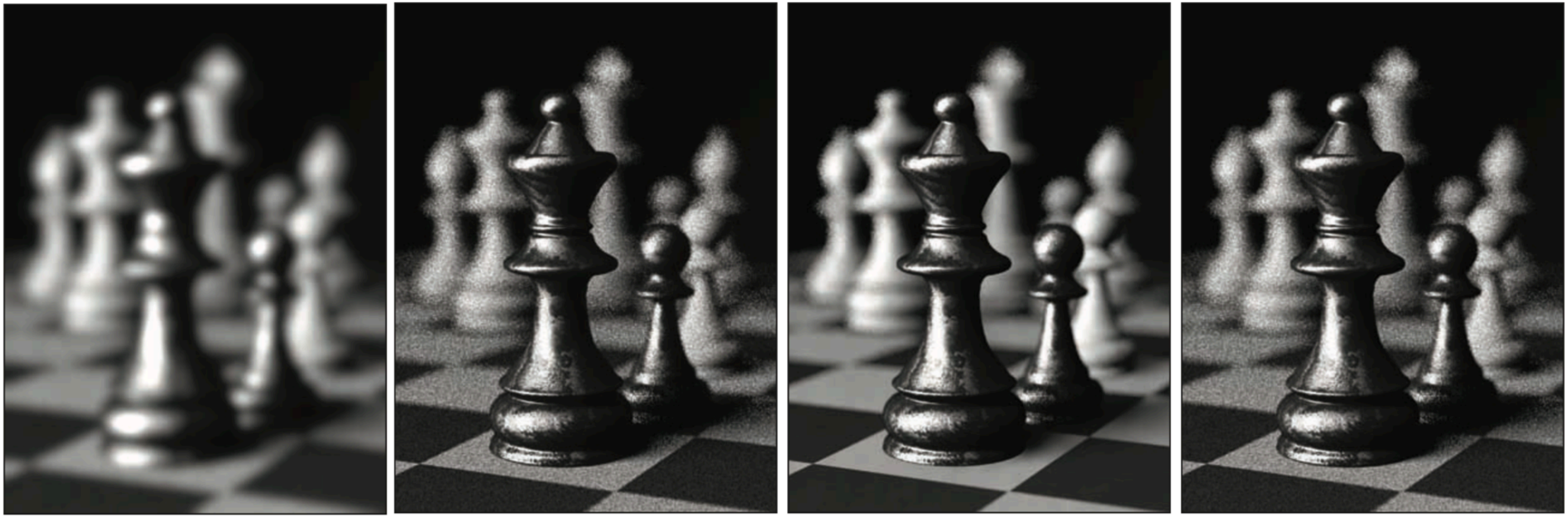
$$\mathbf{c}'_{i,k} = \frac{\sum_{j \in \mathcal{N}} w_{ij} \mathbf{c}_{j,k}}{\sum_{j \in \mathcal{N}} w_{ij}}$$

Results



(a) MC Input (8 spp) **(b)** Our approach (RPF) **(c)** $\alpha_k = 0, \beta_k = 0$

Results



) **(c)** $\alpha_k = 0, \beta_k = 0$ **(d)** $\alpha_k = 1, \beta_k = 0$ **(e)** $\alpha_k = 0, \beta_k = 1$ **(f)** $\alpha_k = 1, \beta_k = 1$

Results


Reference (8,192 spp)
Input Monte Carlo (8 spp)
MDAS
AWR
À-Trous
Our approach (RPF)