Monte Carlo

Applications

Francesco Banterle, Ph.D. - July 2021

Applications
Introduction

- Monte-Carlo methods and integration can be applied in several fields:
  - Deep Learning
  - Imaging
  - Computer Graphics
  - Finance
  - Chemistry
  - Physics

A 2D Problem: Image Filtering

The Bilateral Filter

Introduction

- The bilateral filter is a non-linear filter for images and videos.
- It works in spatial domain and intensity/range domain of an image/video.

- Basically, it is an adaptive linear filter:
  - It behaves as a linear filter in flat regions;
  - At strong edges (step-edge), filtering is “limited”.

The Bilateral Filter

Introduction

Spatial Function

Range Function

$$BF[I](\mathbf{x}, f_s, g_r) = \frac{1}{K(\mathbf{x}, f_s, g_r)} \sum_{\mathbf{y} \in \Omega(\mathbf{x})} I(\mathbf{y}) f_s(\|\mathbf{x} - \mathbf{y}\|) g_r(\|I(\mathbf{y}) - I(\mathbf{x})\|),$$

$$K[I](\mathbf{x}, f_s, g_r) = \sum_{\mathbf{y} \in \Omega(\mathbf{x})} f_s(\|\mathbf{x} - \mathbf{y}\|) g_r(\|I(\mathbf{y}) - I(\mathbf{x})\|),$$

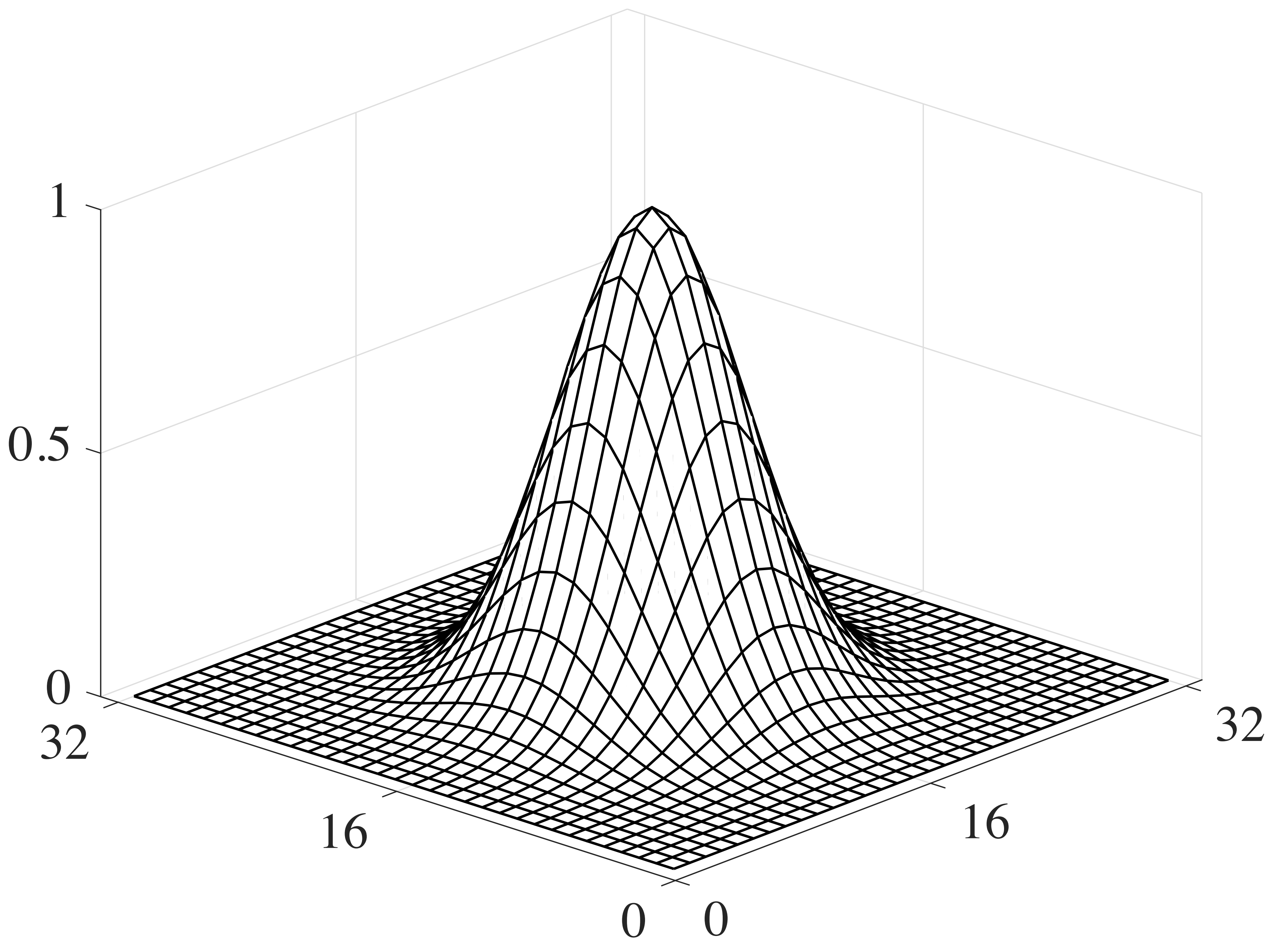
The Bilateral Filter
Introduction

- f_s (Spatial function): a Gaussian function
- g_r (Range function): a Gaussian function
- How large is the kernel?
  - If the spatial function is a Gaussian:

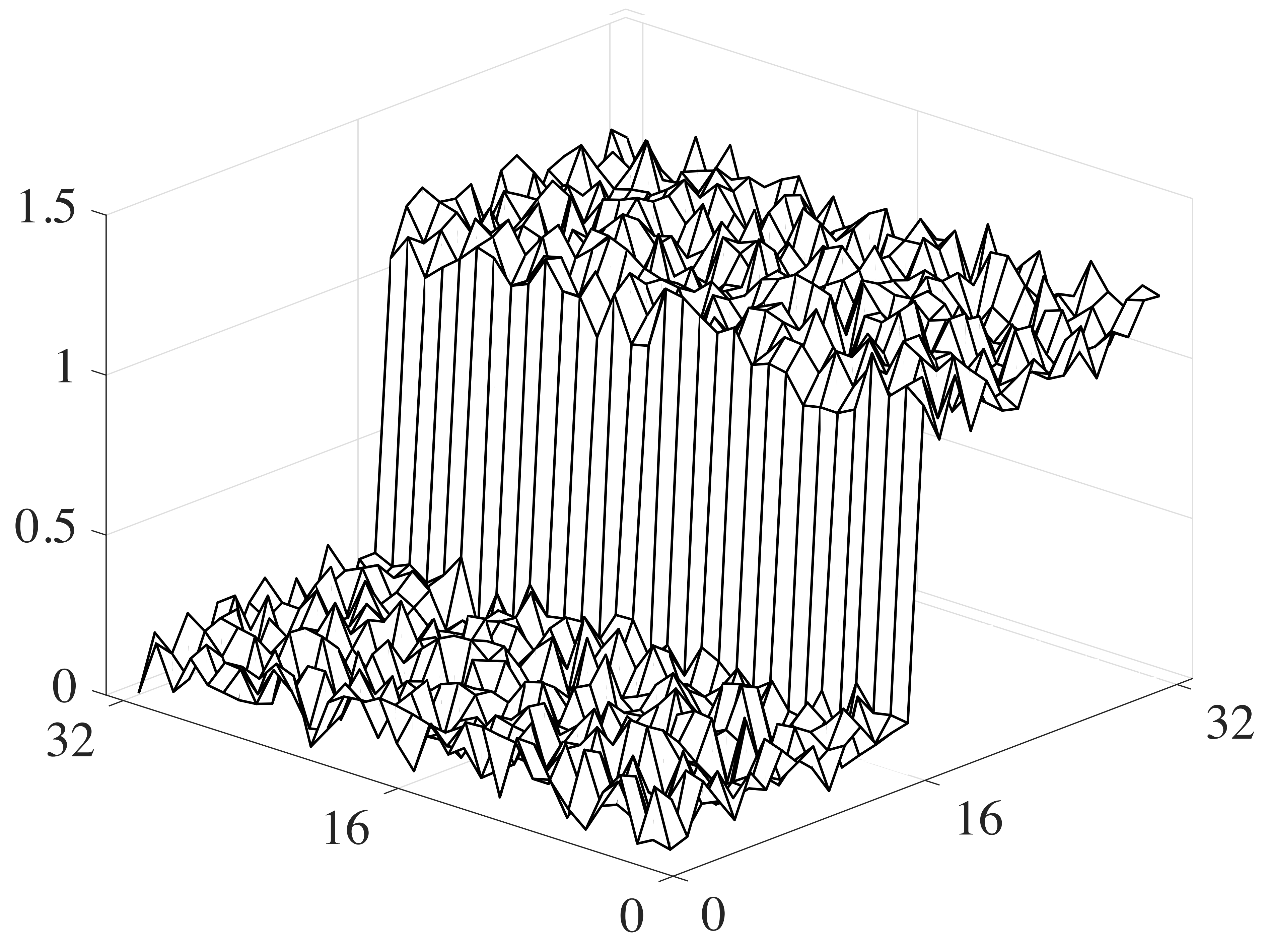
$$N = M = \frac{5}{2}\sigma_s \, .$$

The Bilateral Filter

Example

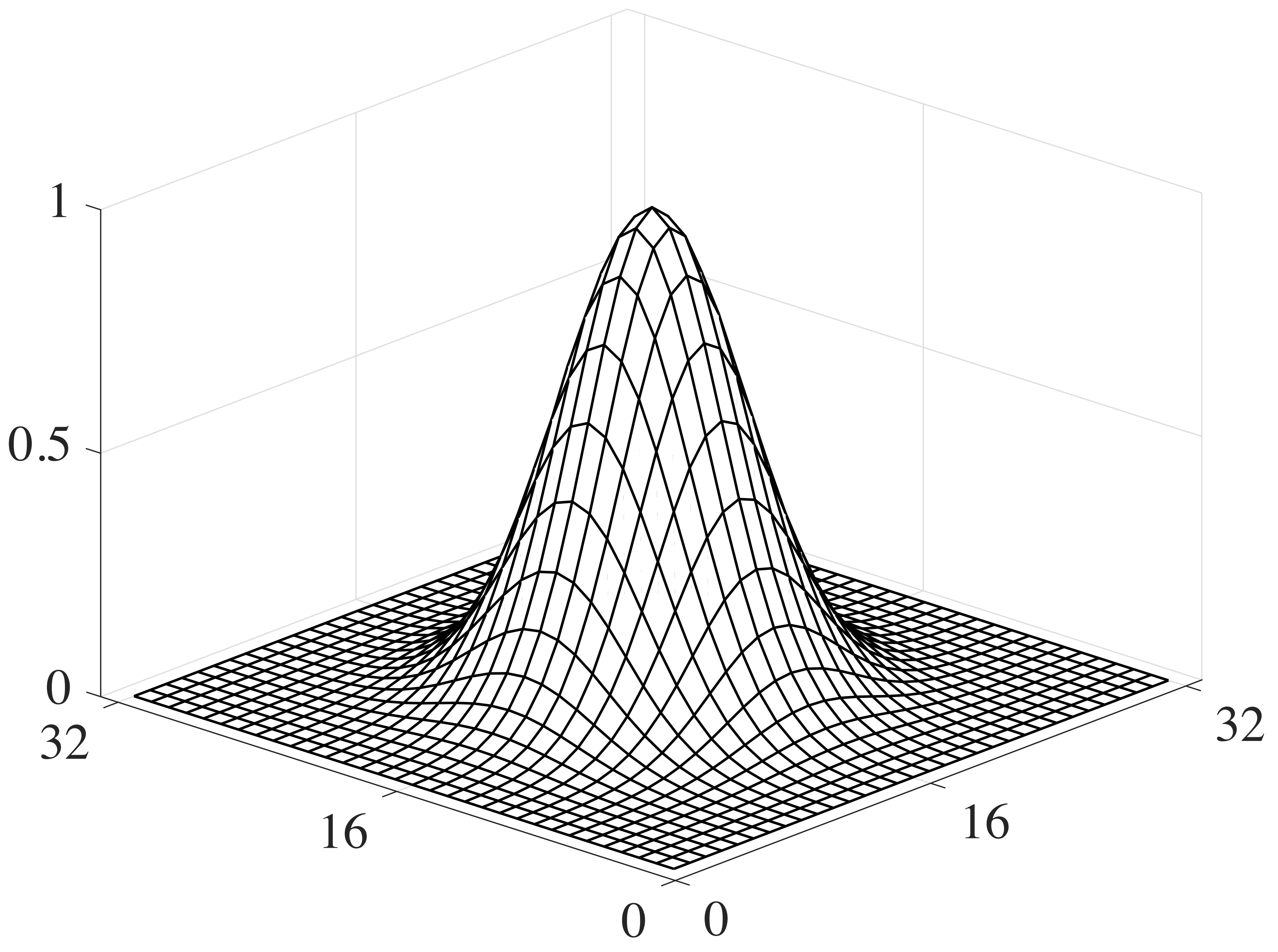


Kernel

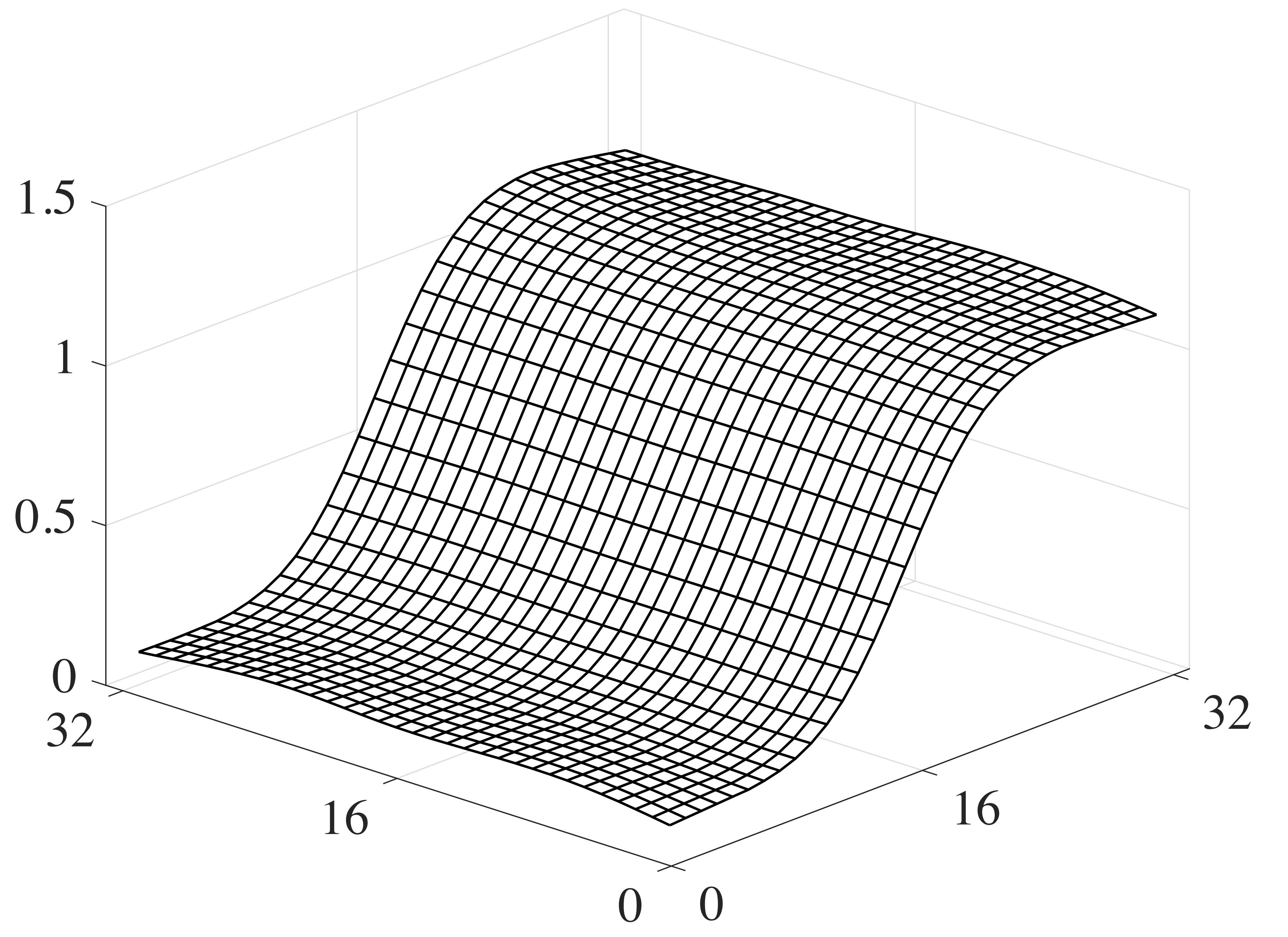


Image

The Bilateral Filter
Example



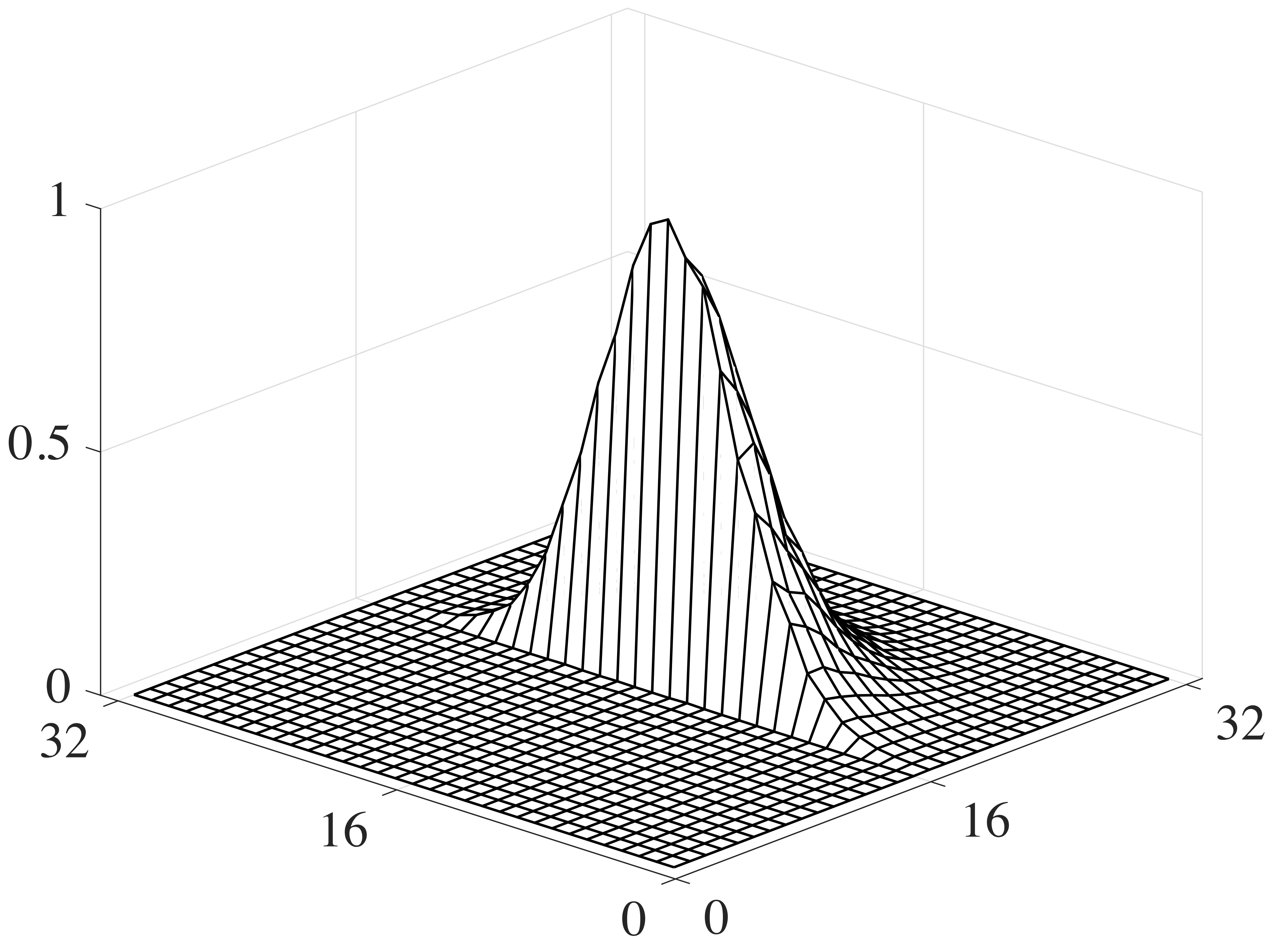
Kernel



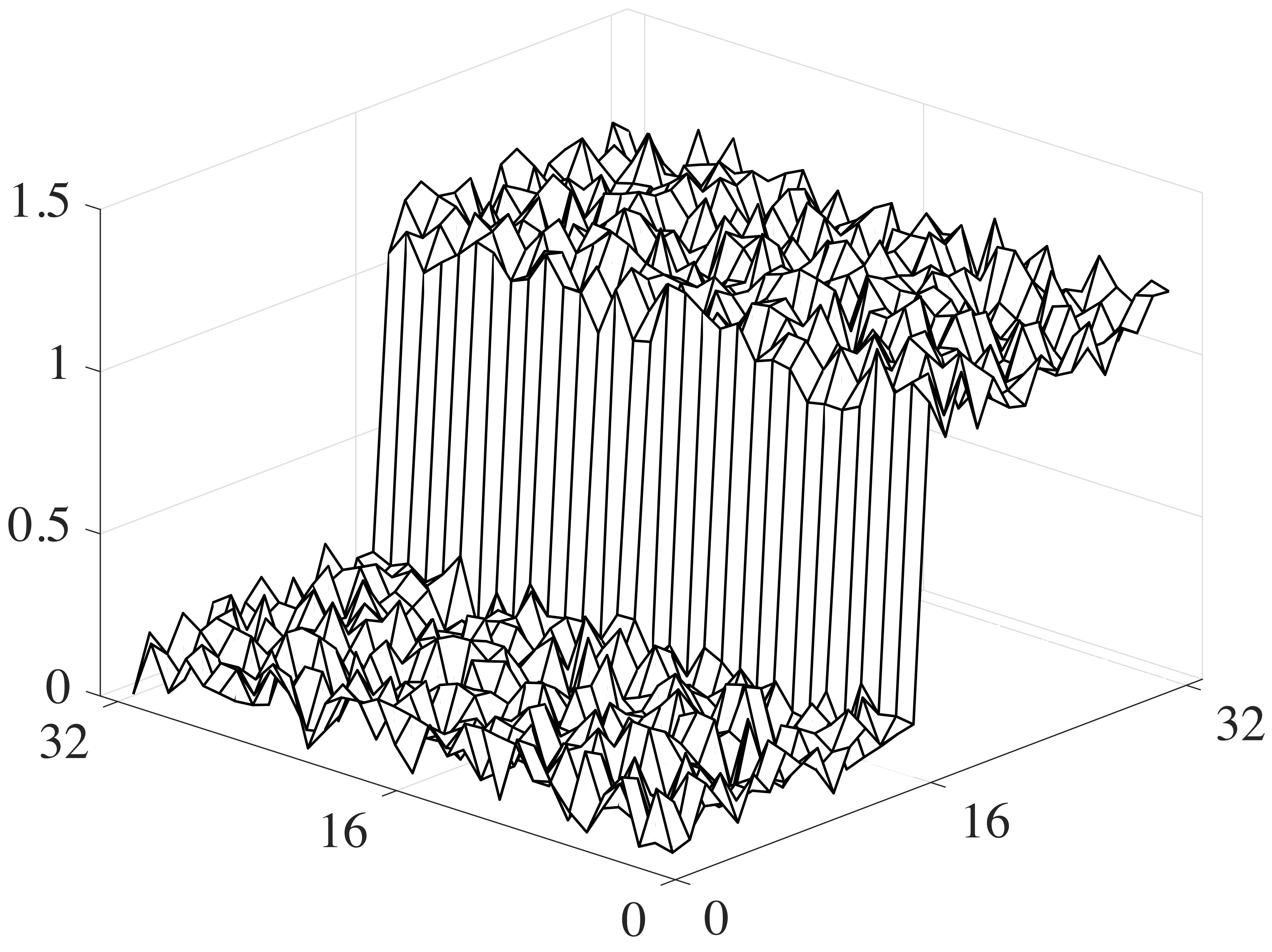
Image

The Bilateral Filter

Example



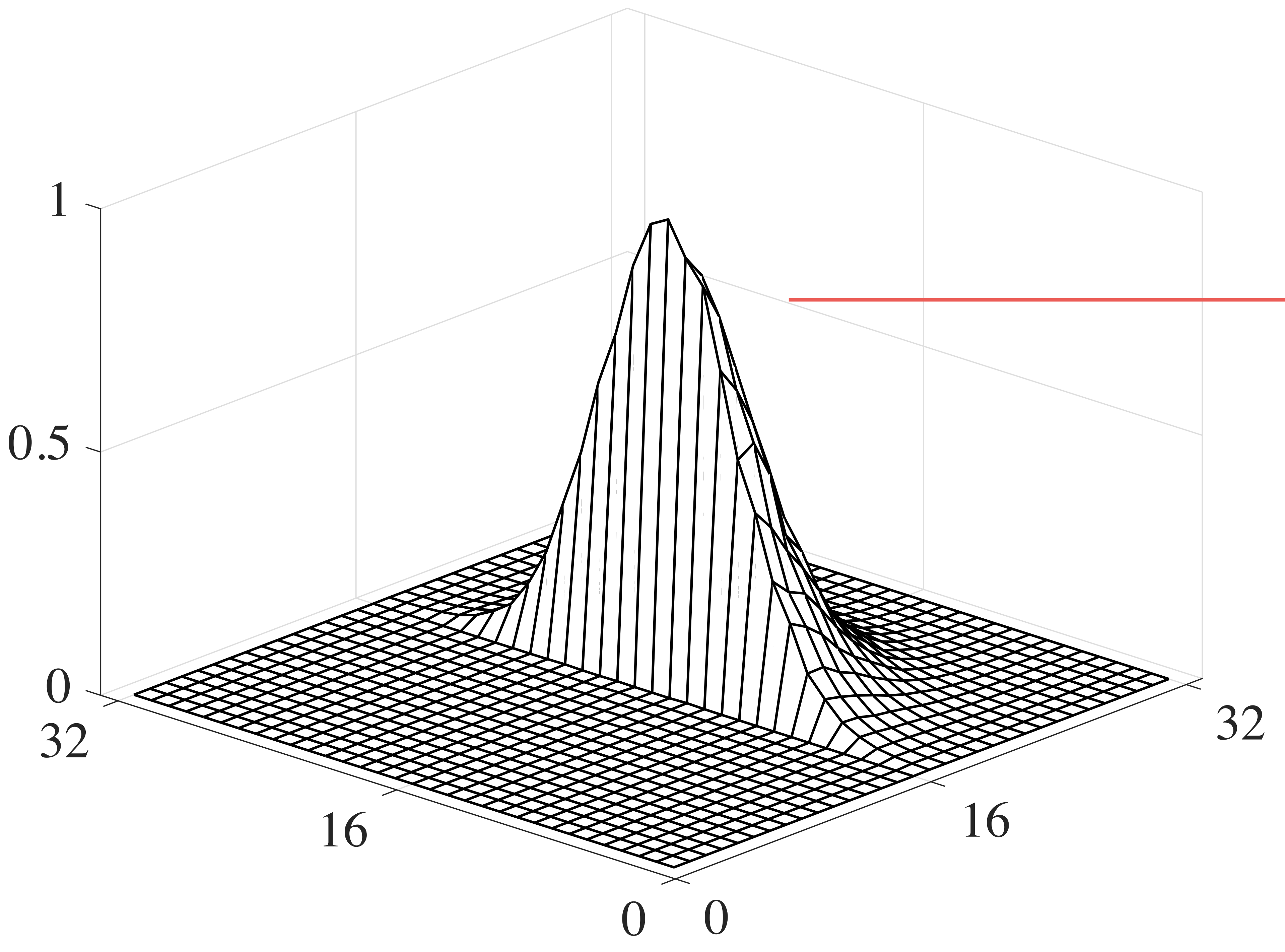
Kernel
(change for each pixel!!)

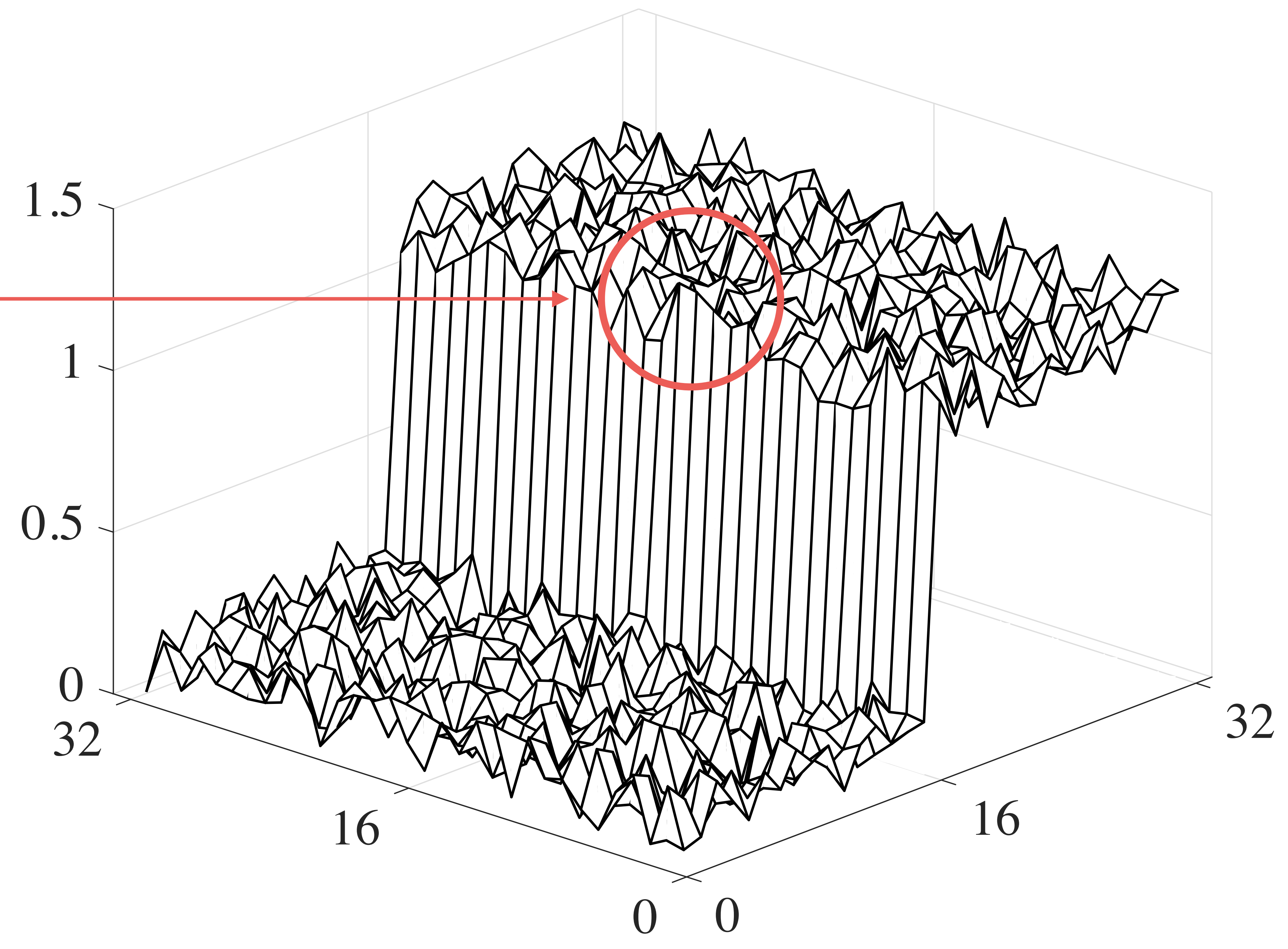


Image

The Bilateral Filter

Example



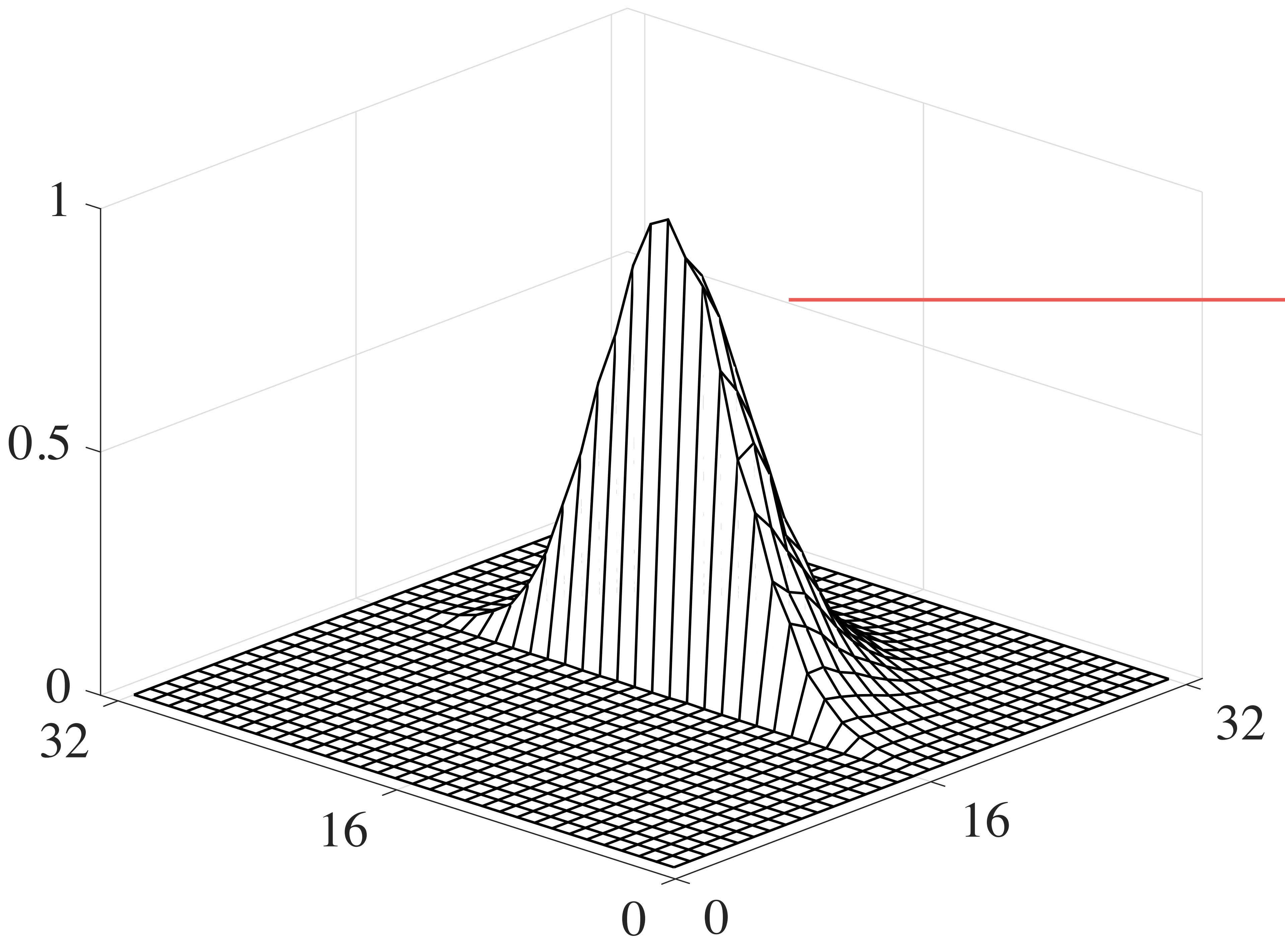


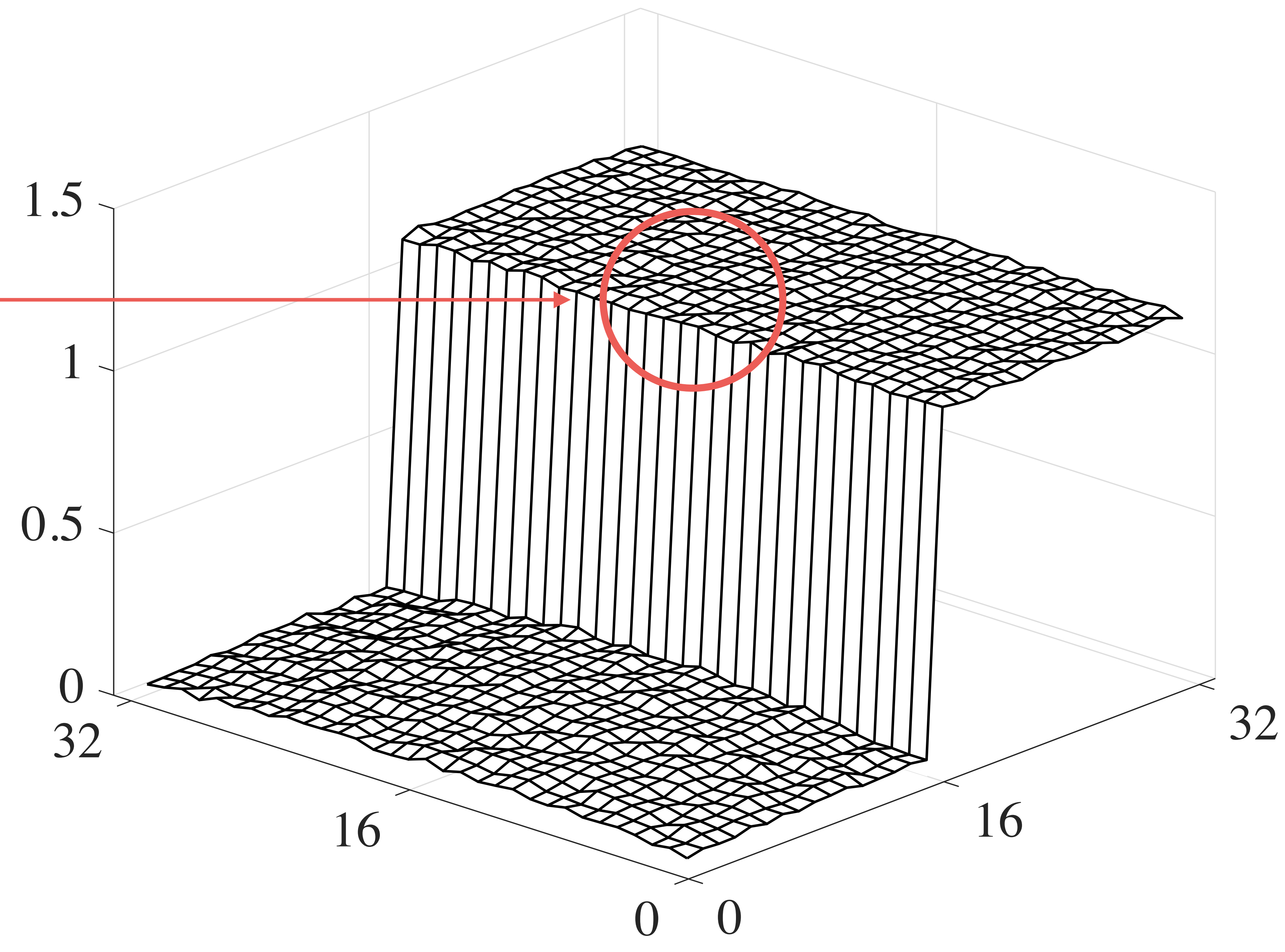
Kernel
(change for each pixel!!)

Image

The Bilateral Filter

Example





Kernel
(change for each pixel!!)

Image

The Bilateral Filter
Example

The Bilateral Filter

Example

The Bilateral Filter

Example

The Bilateral Filter

Example

The Bilateral Filter

Computational Complexity

- The main problem of the filter is its high computational complexity for real-time applications:

$$\mathcal{O}(nk^2),$$

  where n is the number of pixels of an image/video, and k is the size.

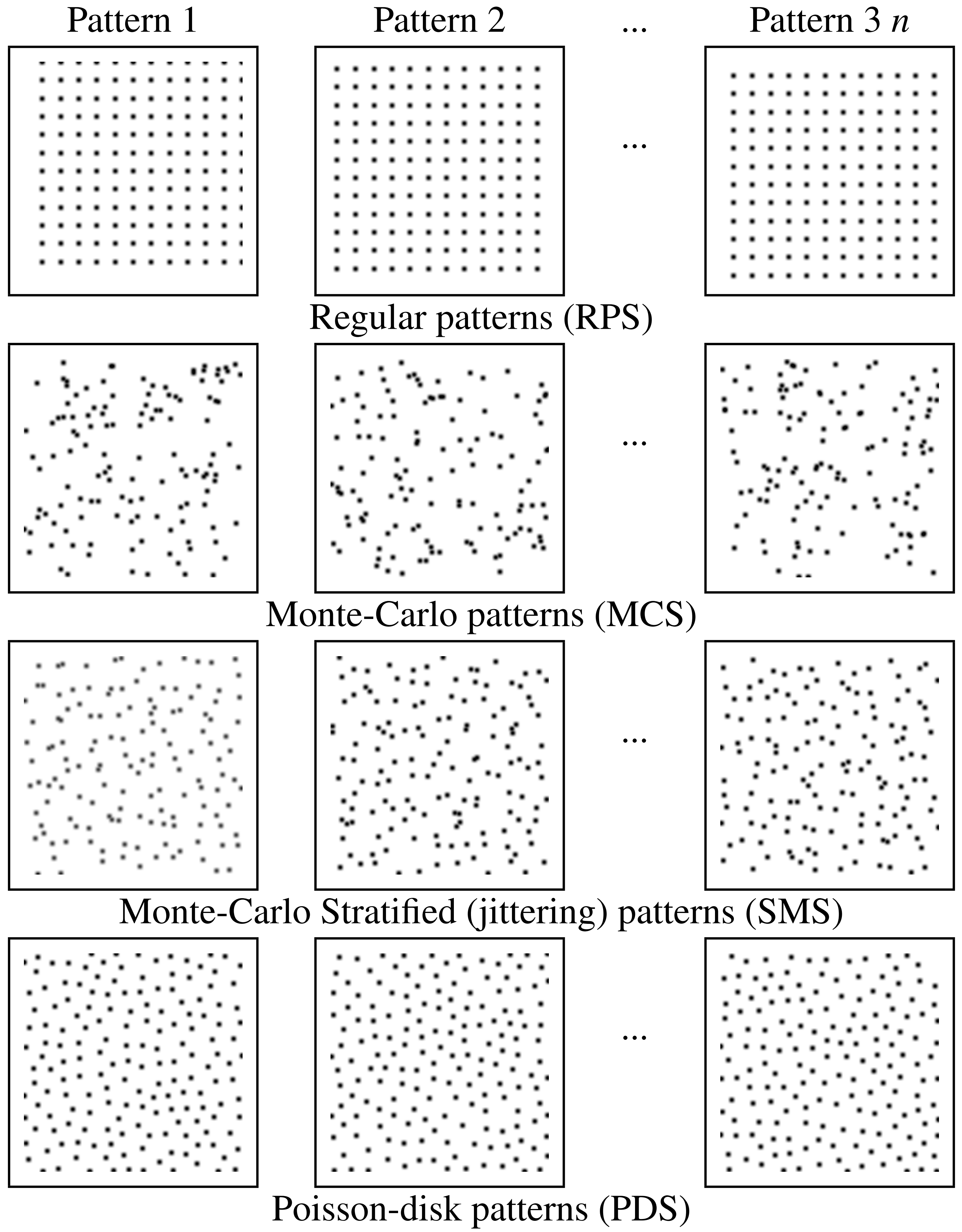
- Compared to a Gaussian filter:
  - Not separable;
  - No Fourier domain.

The Bilateral Filter
Monte-Carlo

- In this case, we can solve with Monte-Carlo!
- Basic idea:
  - We draw sample according the spatial Gaussian:
    - Box-Muller method.
  - We limit the number of samples to k or ck; with $c < k$ a constant.

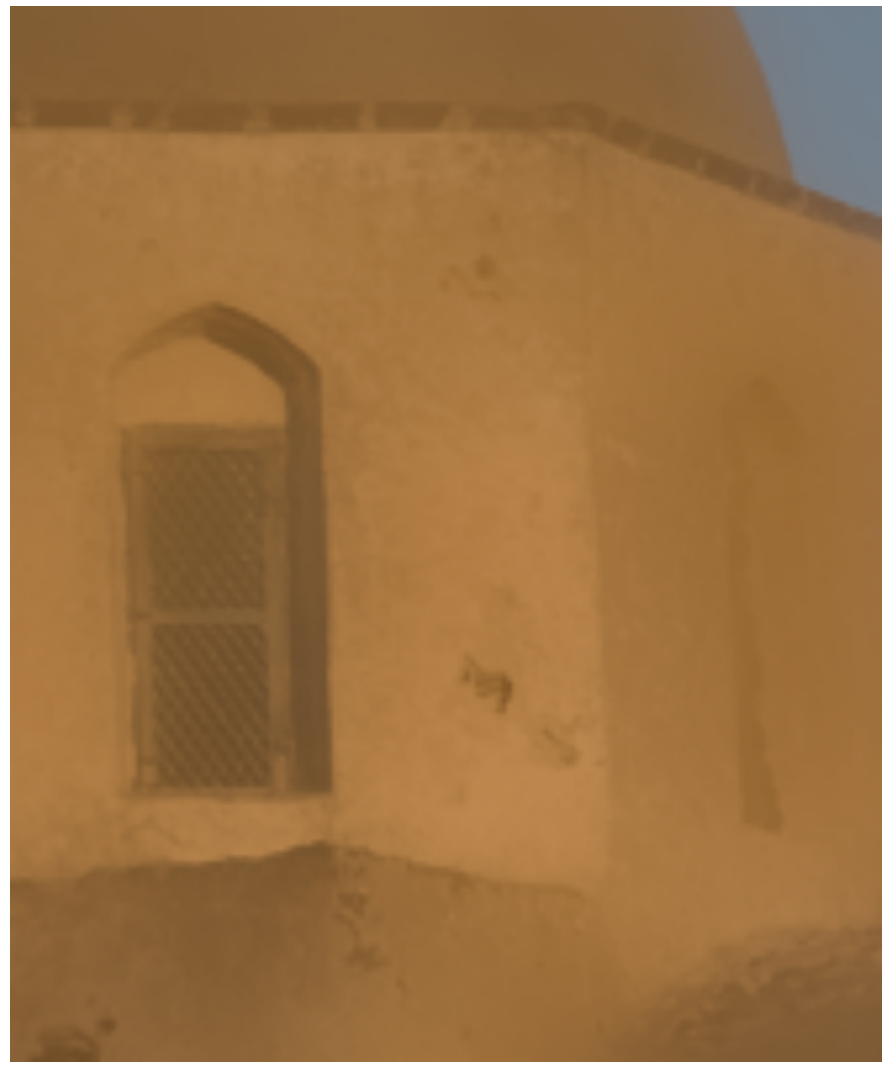
The Bilateral Filter

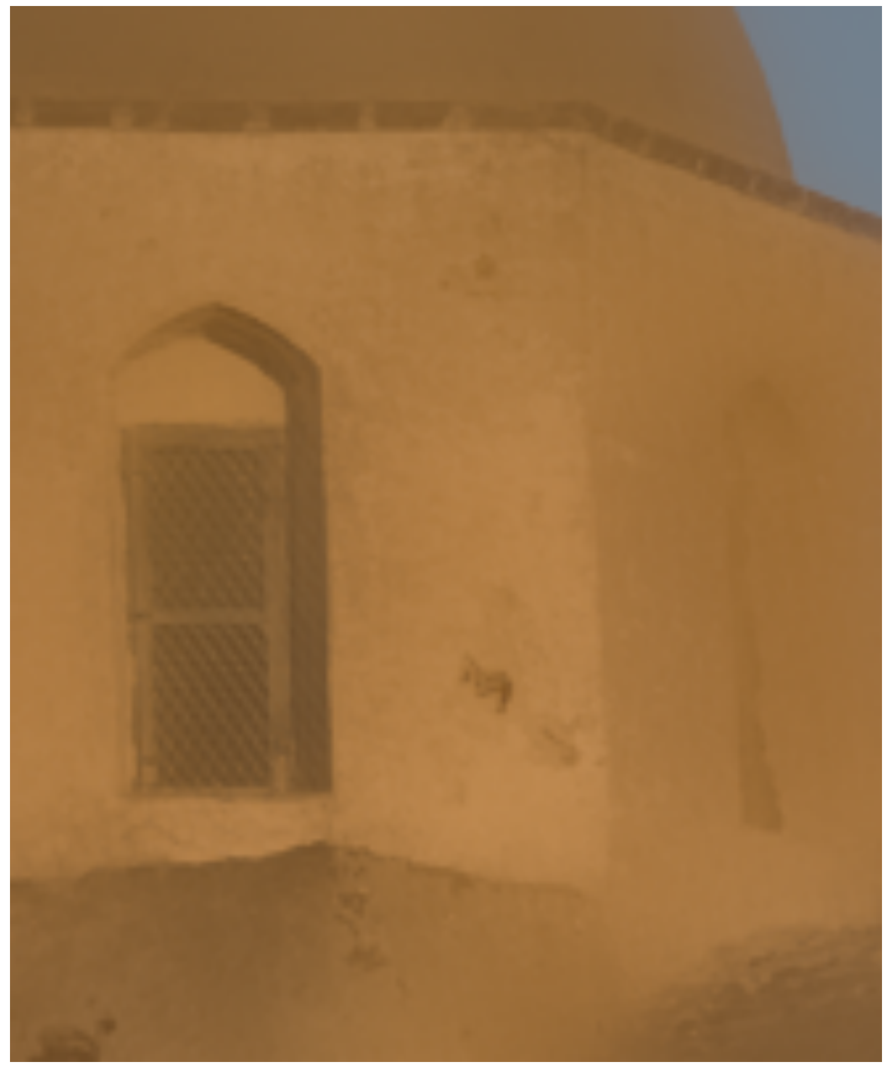
Sampling Strategies



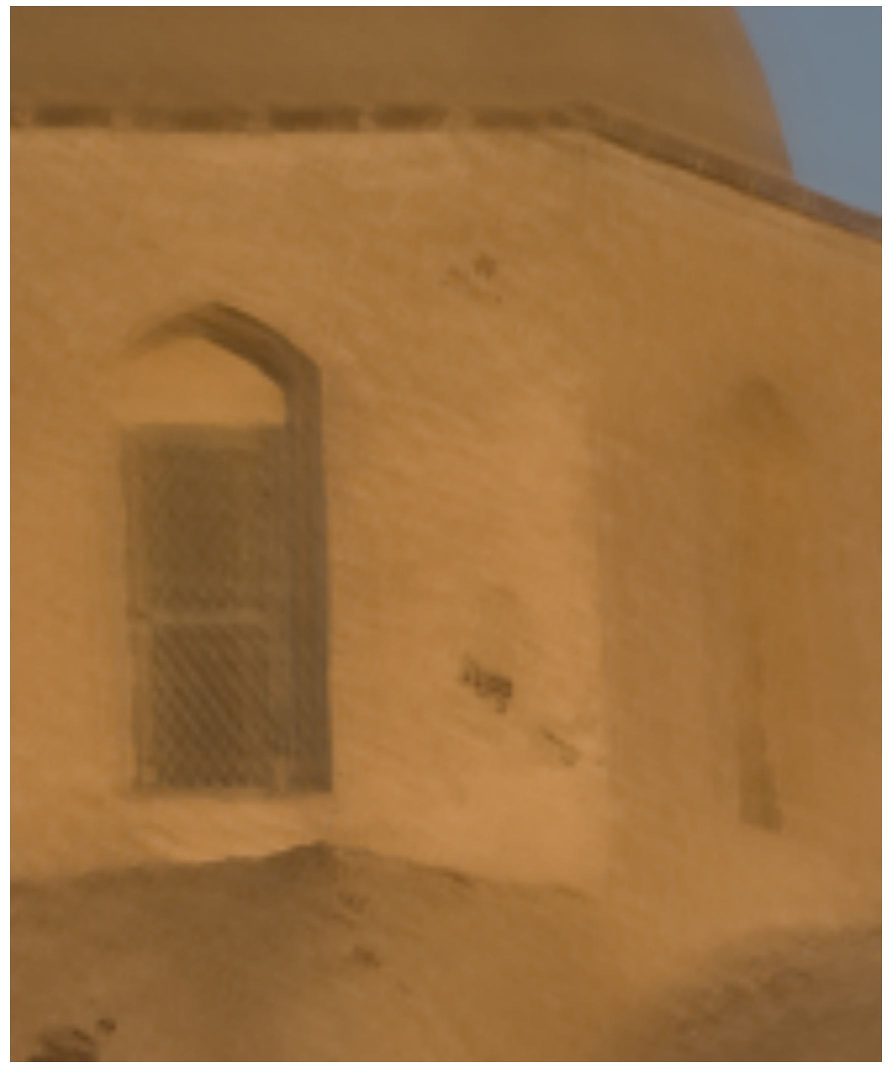
The Bilateral Filter

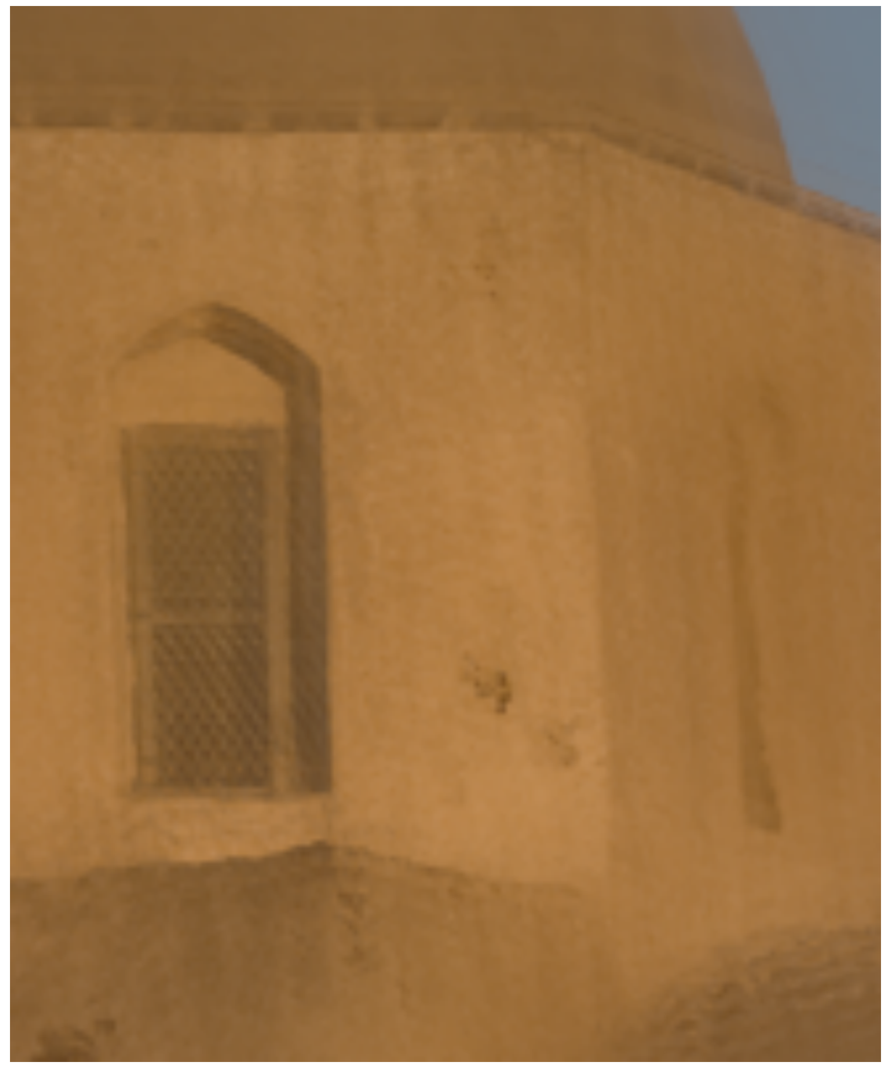
Sampling Strategies


Original


Poisson-Disk


Regular Grid


Monte-Carlo


Stratified
Monte-Carlo

The Bilateral Filter

Sampling Strategies

A Recursive Problem: Rendering

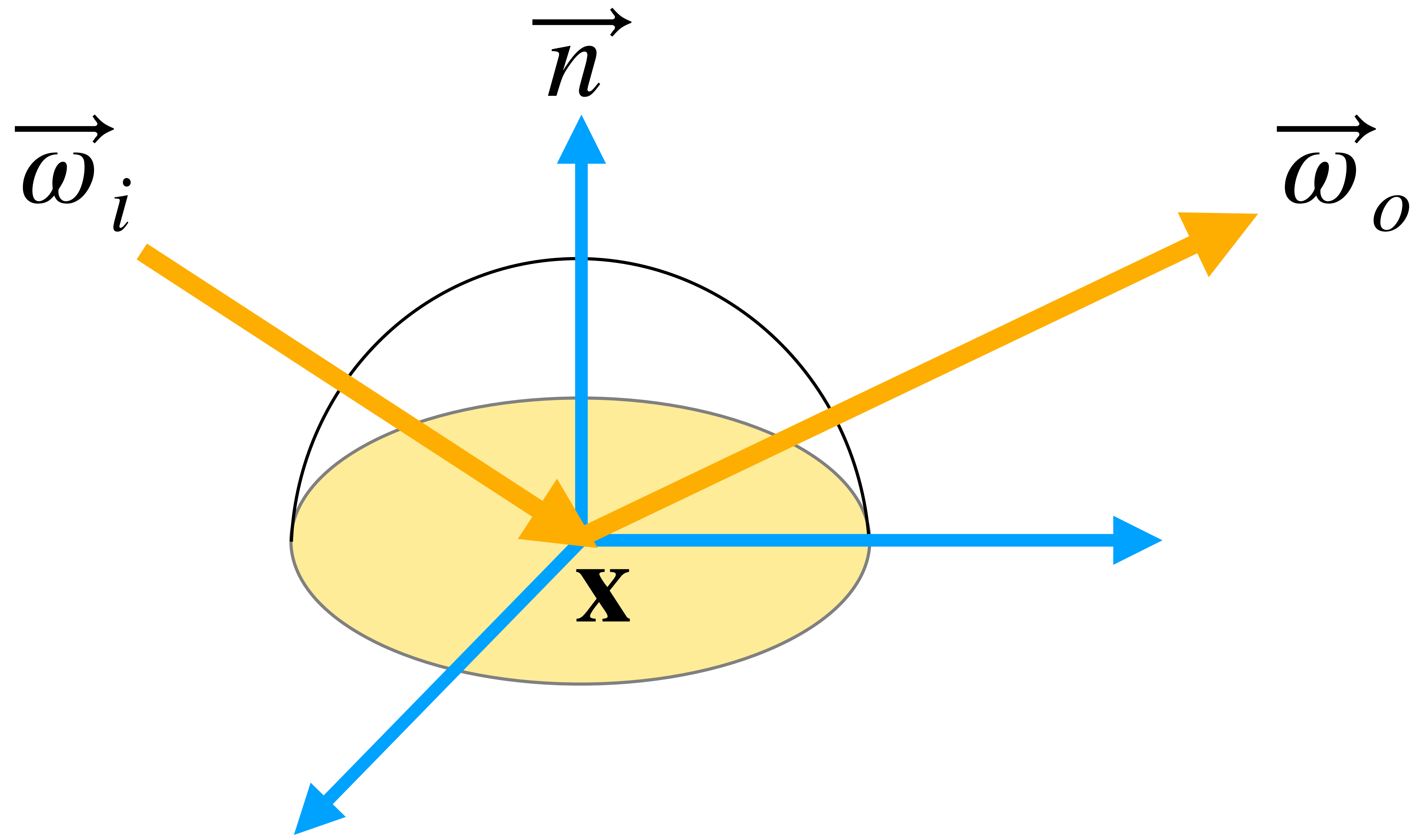
Path-Tracing
Introduction

- A classic problem in Computer Graphics is given:
    - Camera;
    - 3D Geometry;
    - Light sources' description;
    - Materials' description.
- To compute the color of each pixel in the image plane of our plane by simulating the light transport in a physically based manner.

Path-Tracing

The Rendering Equation

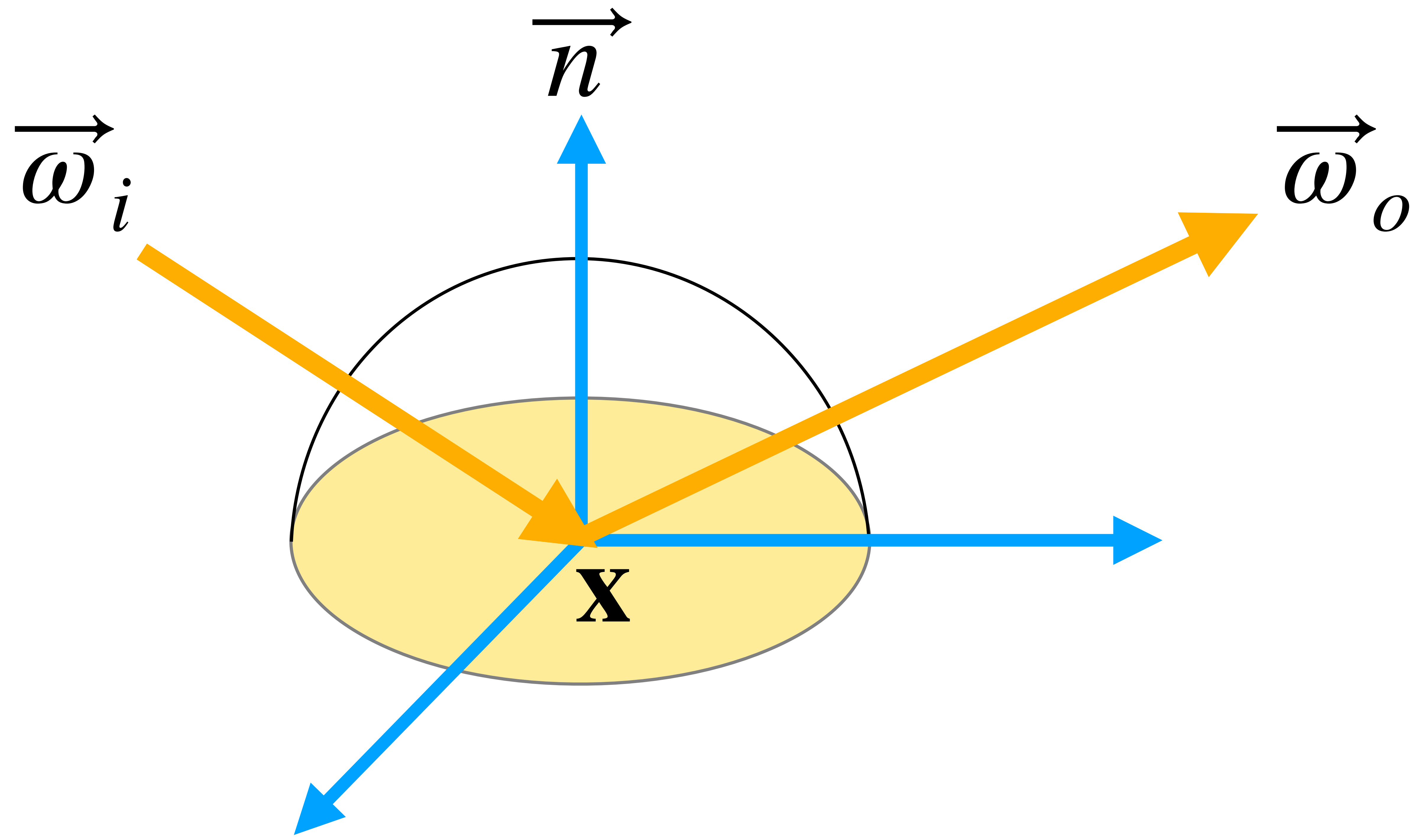
$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o, \lambda) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \, | \overrightarrow{n} \cdot \overrightarrow{\omega}_i | \, d\omega_i$$



Path-Tracing

The Rendering Equation

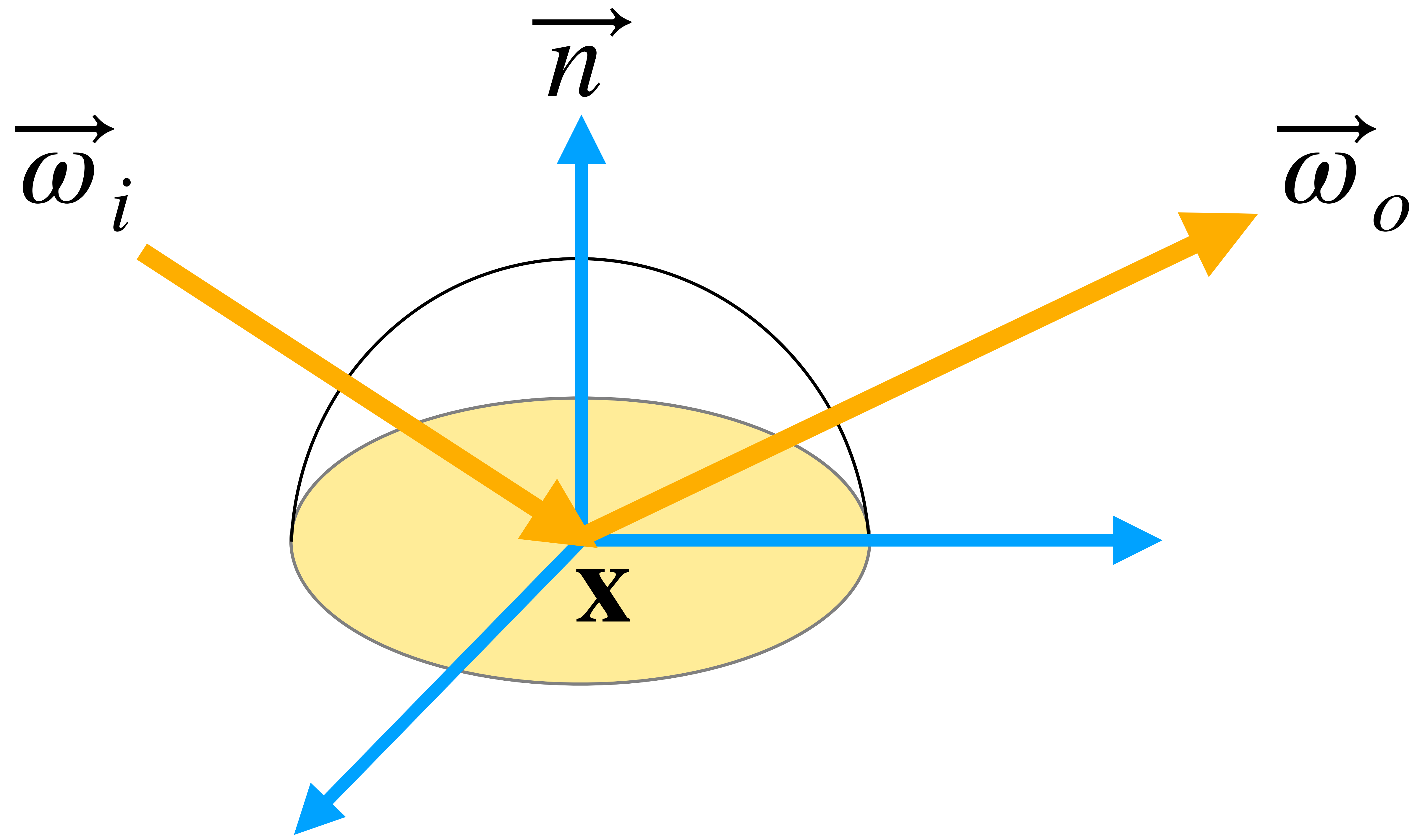
$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o, \lambda) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \,|\, \overrightarrow{n} \cdot \overrightarrow{\omega}_i \,|\, d\omega_i$$



Path-Tracing

The Rendering Equation

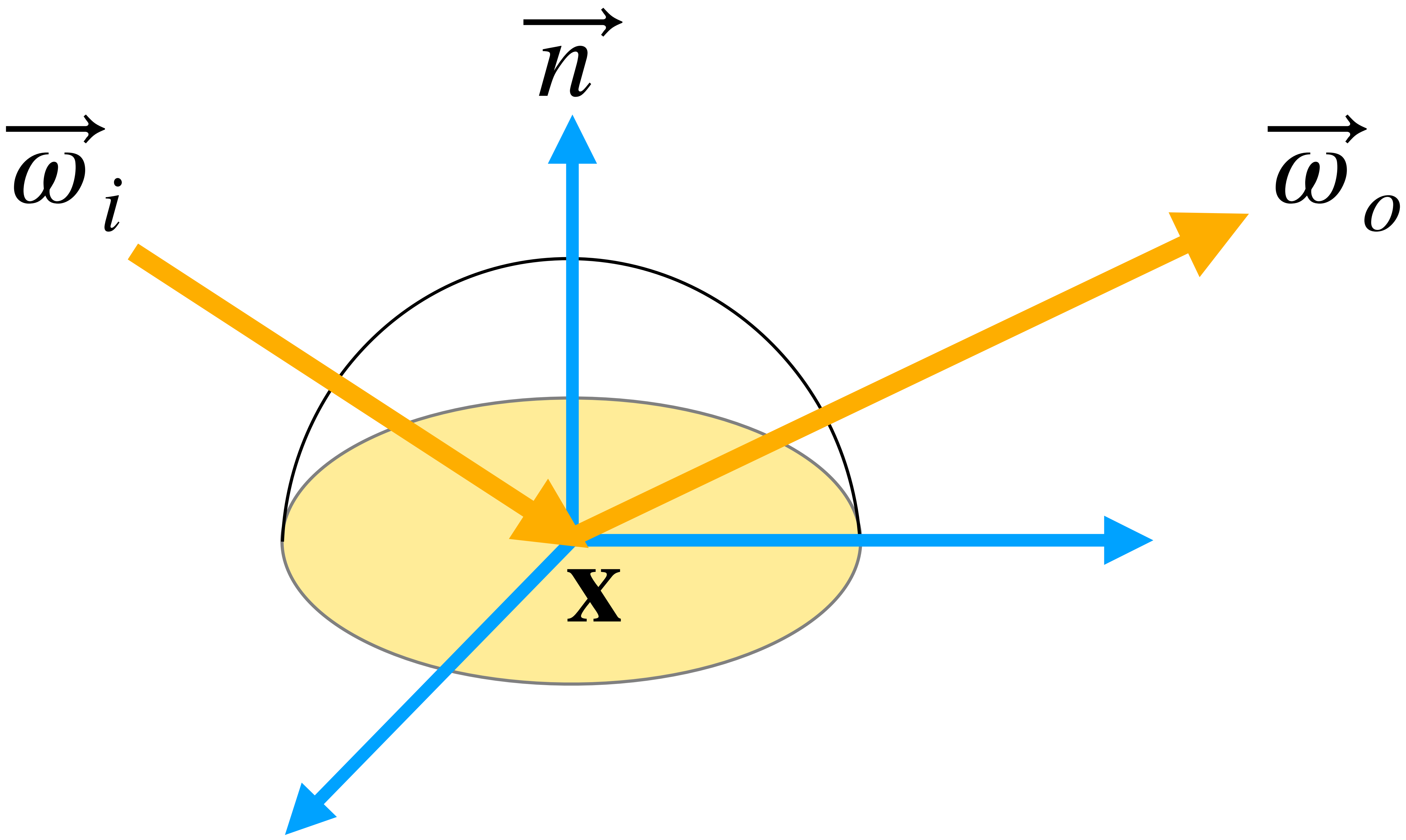
$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o, \lambda) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \,|\, \overrightarrow{n} \cdot \overrightarrow{\omega}_i \,|\, d\omega_i$$



Path-Tracing

The Rendering Equation

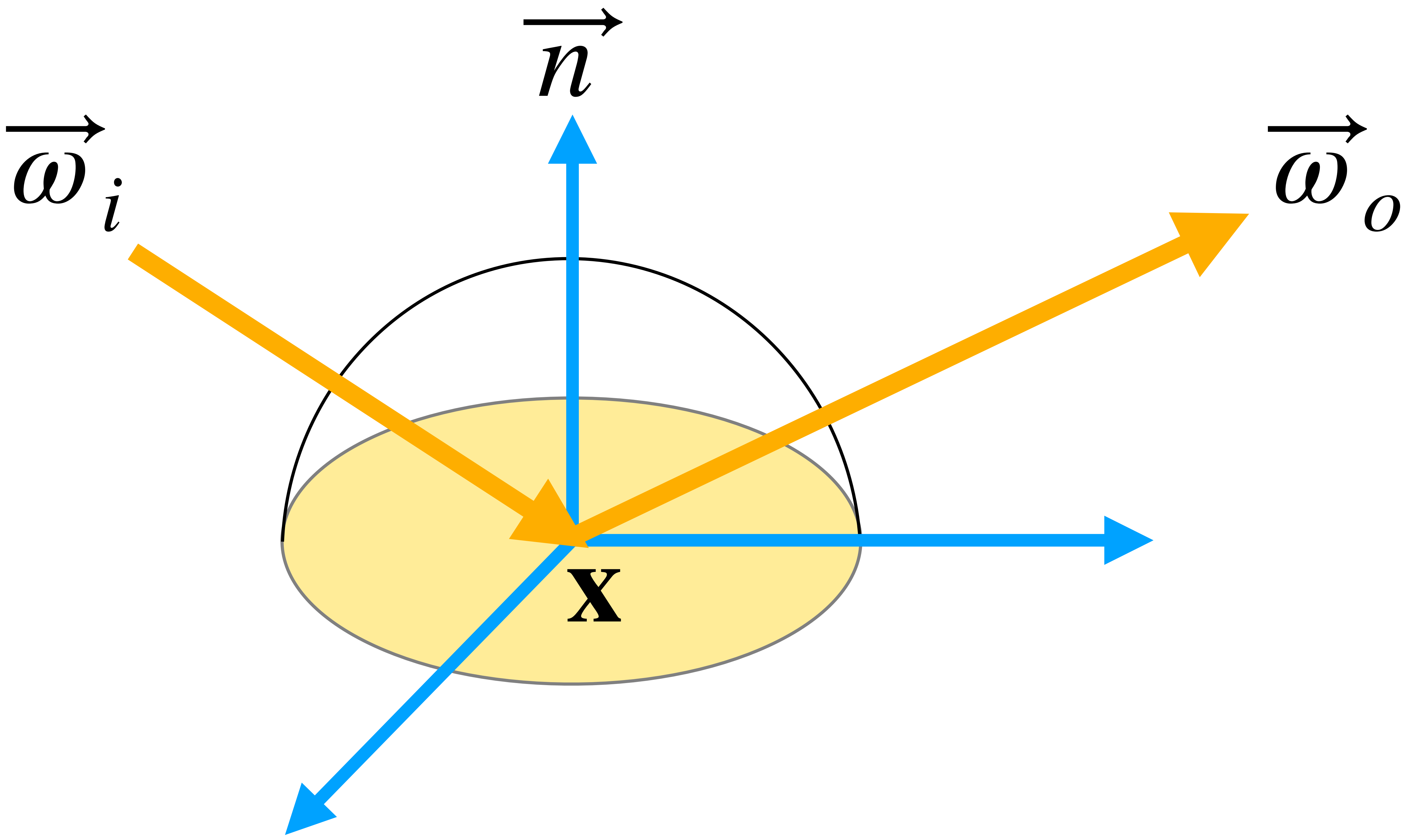
$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o, \lambda) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \, | \overrightarrow{n} \cdot \overrightarrow{\omega}_i | \, d\omega_i$$



Path-Tracing

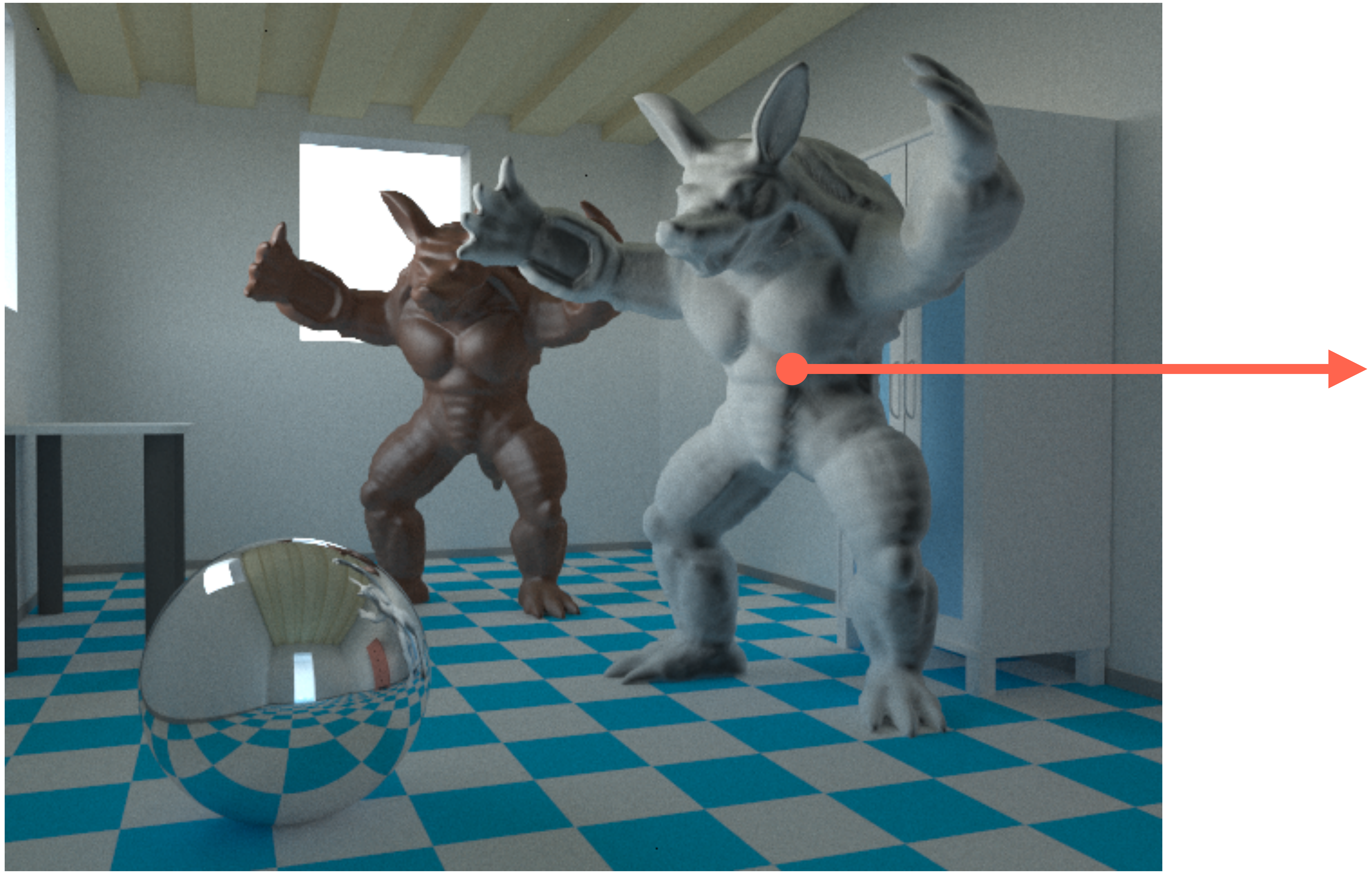
The Rendering Equation

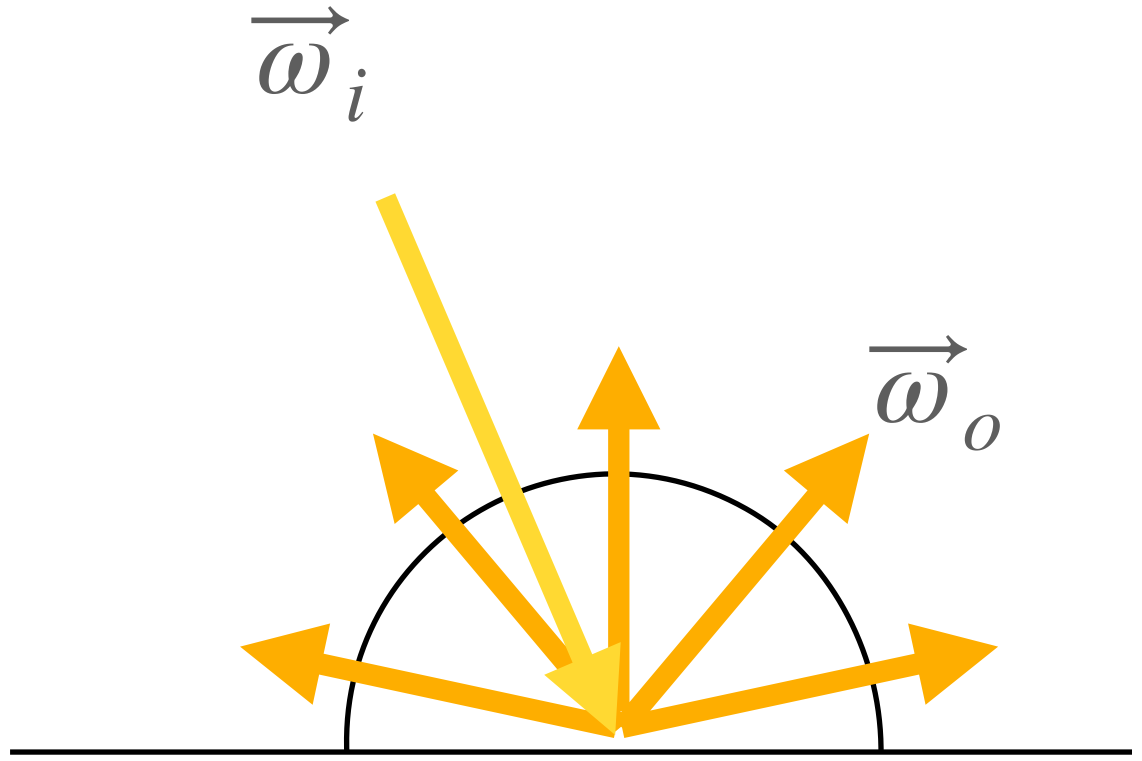
$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o, \lambda) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \, | \overrightarrow{n} \cdot \overrightarrow{\omega}_i \, | \, d\omega_i$$



Path-Tracing

The Rendering Equation: BRDF

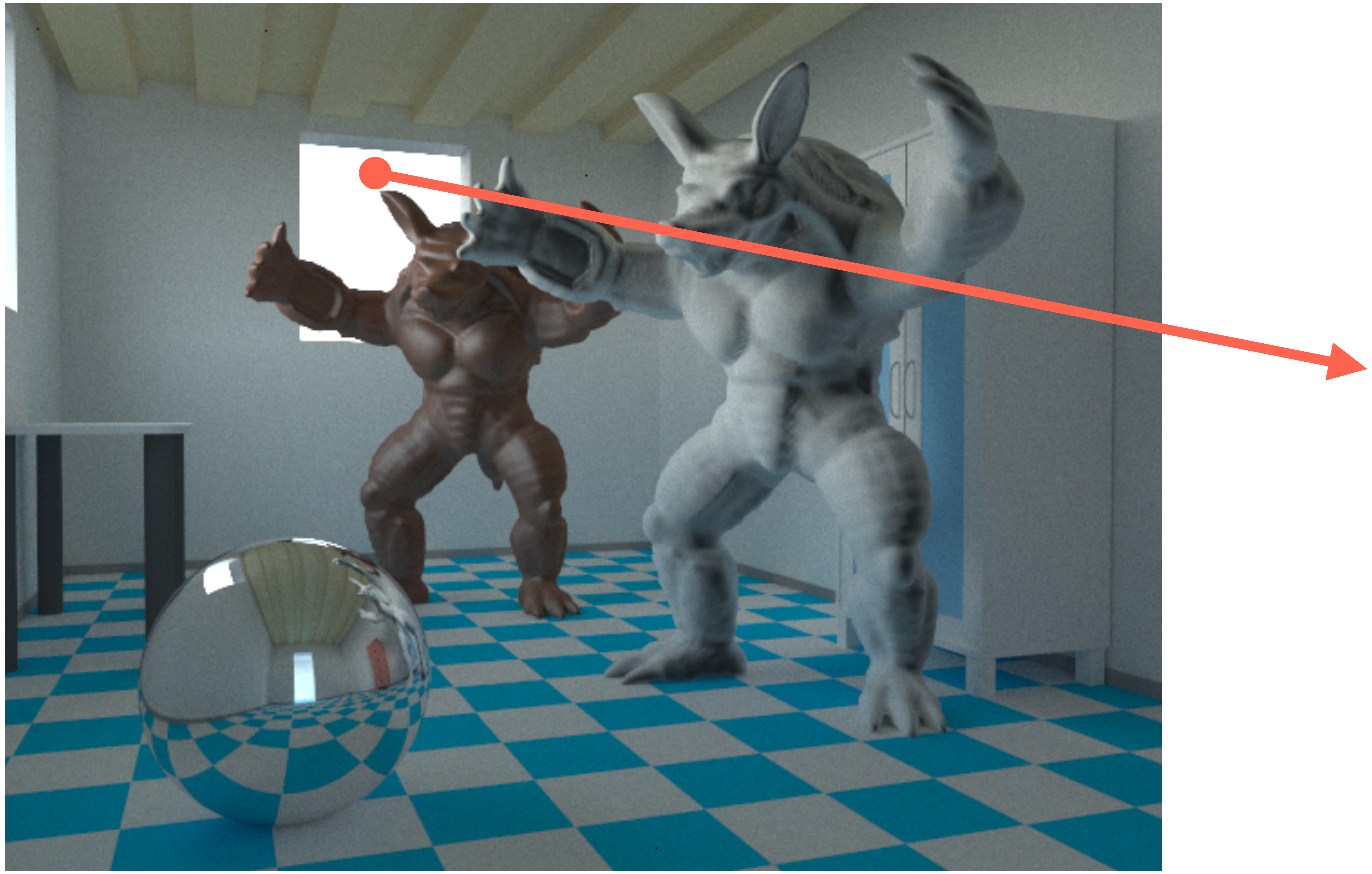


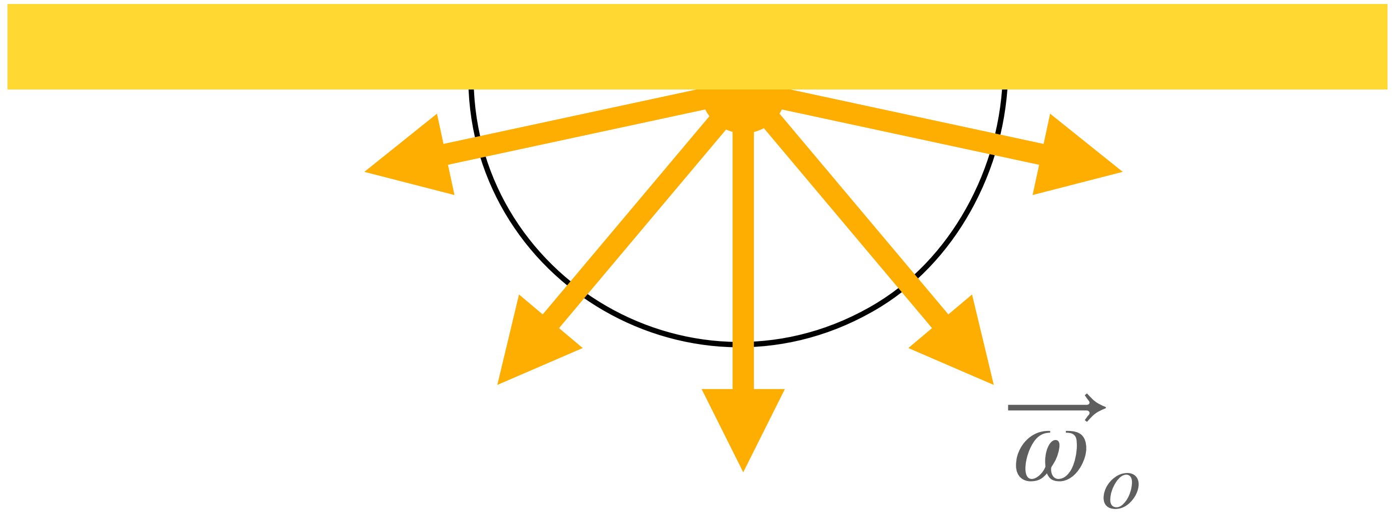


$$f_r(\mathbf{x}), \vec{\omega}_i, \vec{\omega}_o, \lambda) = \frac{\rho_\lambda}{\pi}$$

Path-Tracing

The Rendering Equation: Light Sources

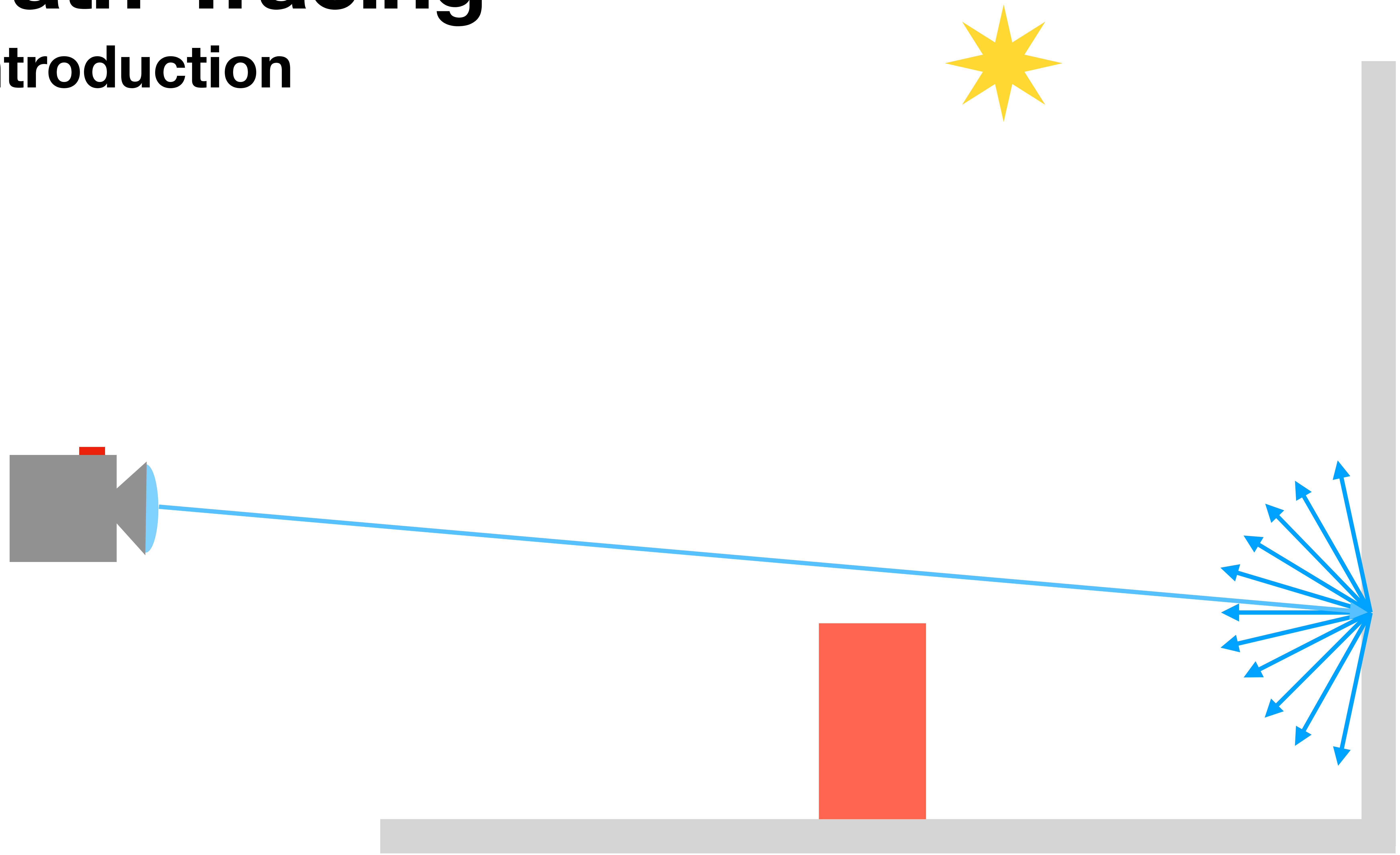




$$I_e(\mathbf{x}, \vec{\omega}_o) = \frac{\Phi_\lambda}{\pi A}$$

Path-Tracing

Introduction



$$L_o(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) = L_e(\mathbf{x}, \overrightarrow{\omega}_o, \lambda) + \int_\Omega f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o) L_i(\mathbf{x}, \overrightarrow{\omega}_i, \lambda) \, | \overrightarrow{n} \cdot \overrightarrow{\omega}_i | \, d\omega_i$$

Path-Tracing
Introduction

- In a deterministic way, we should shoot n rays at each bounce for each location:

$$\sum_k n^k,$$

and this highly impractical.

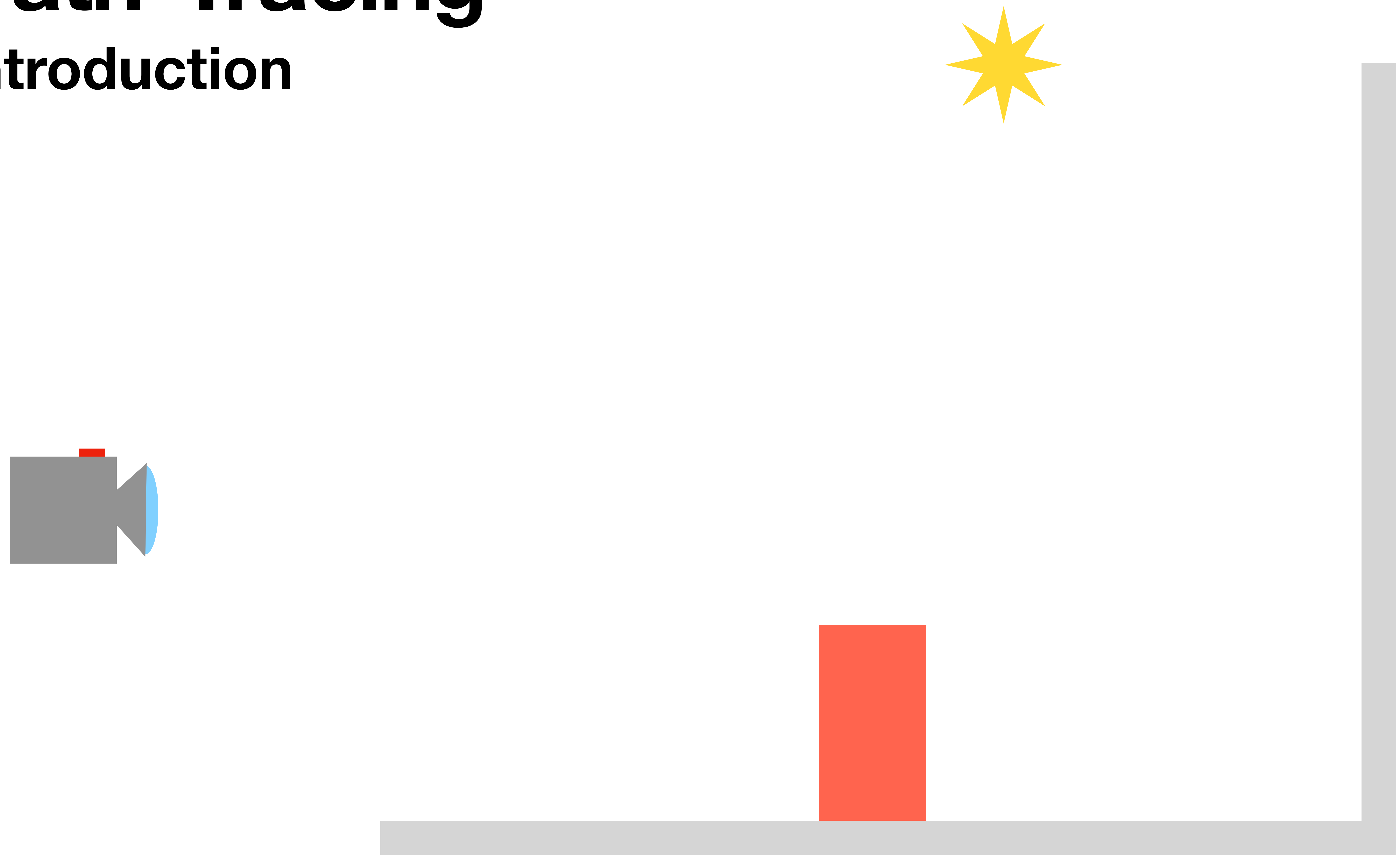
- Our estimator is the classic estimator seen so far:

$$\hat{\mu} = \frac{1}{N}\sum_{i=1}^{n} Y_i \, .$$

So We Generate Different Paths and We Sum Them Up

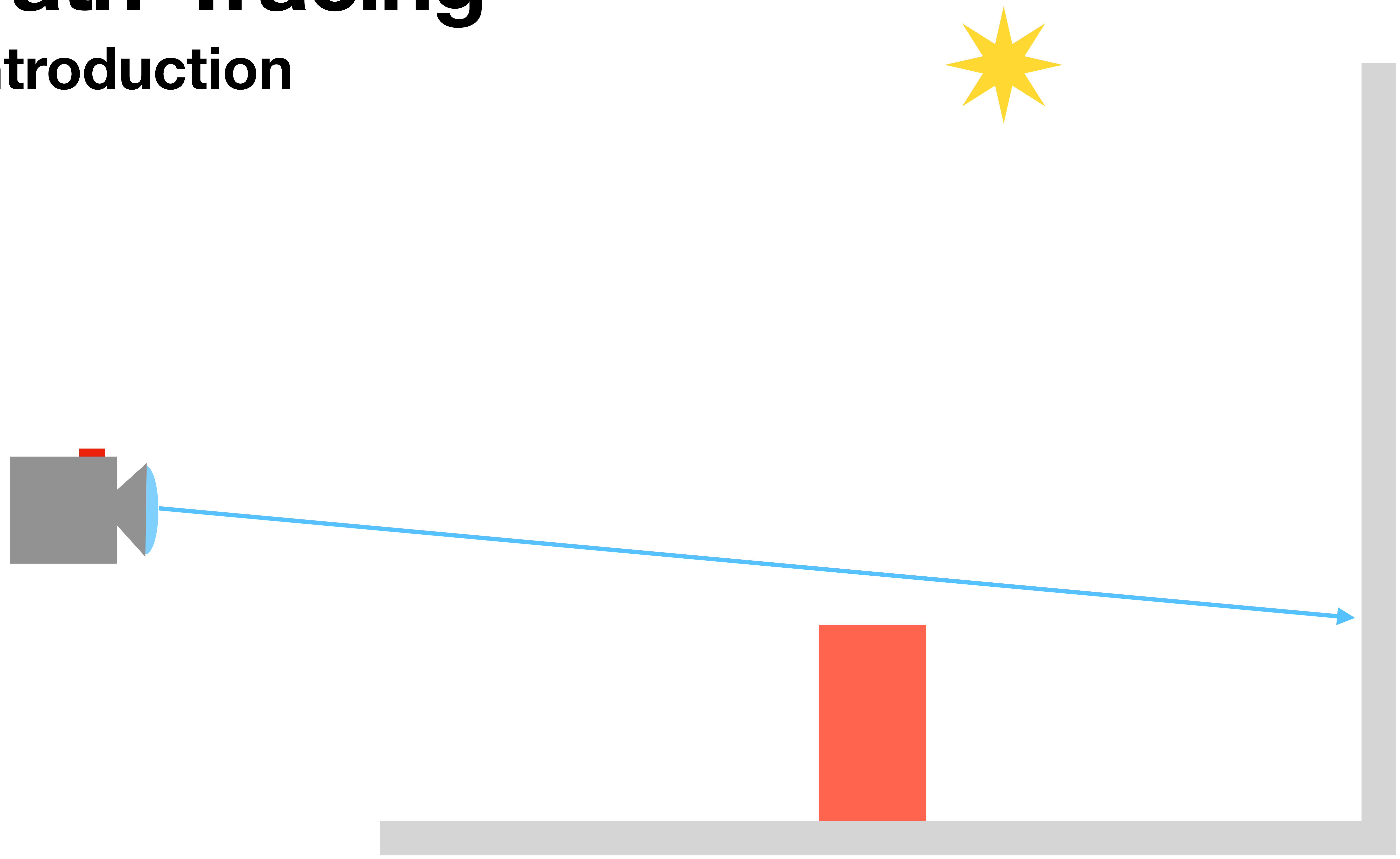
Path-Tracing

Introduction



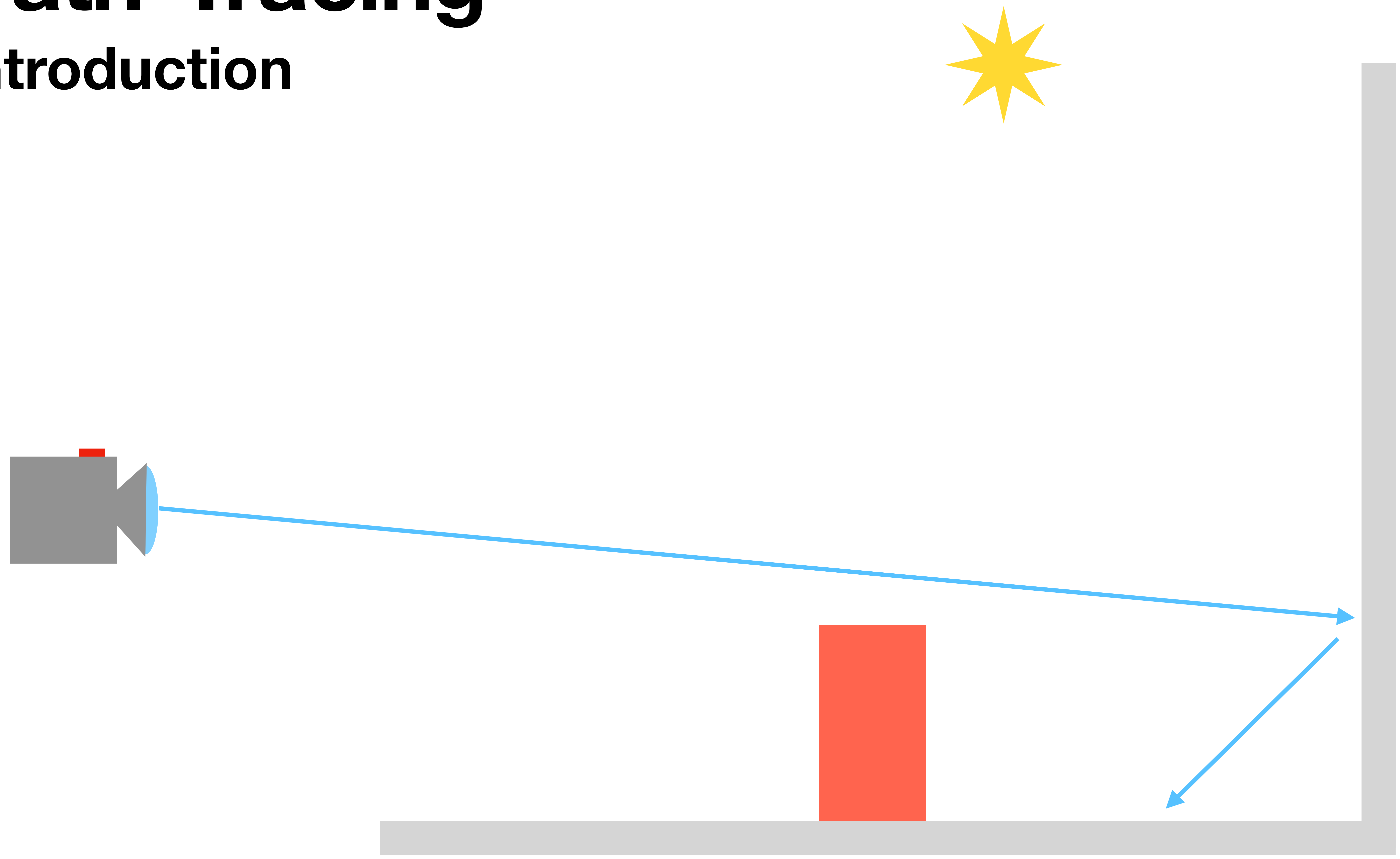
Path-Tracing

Introduction



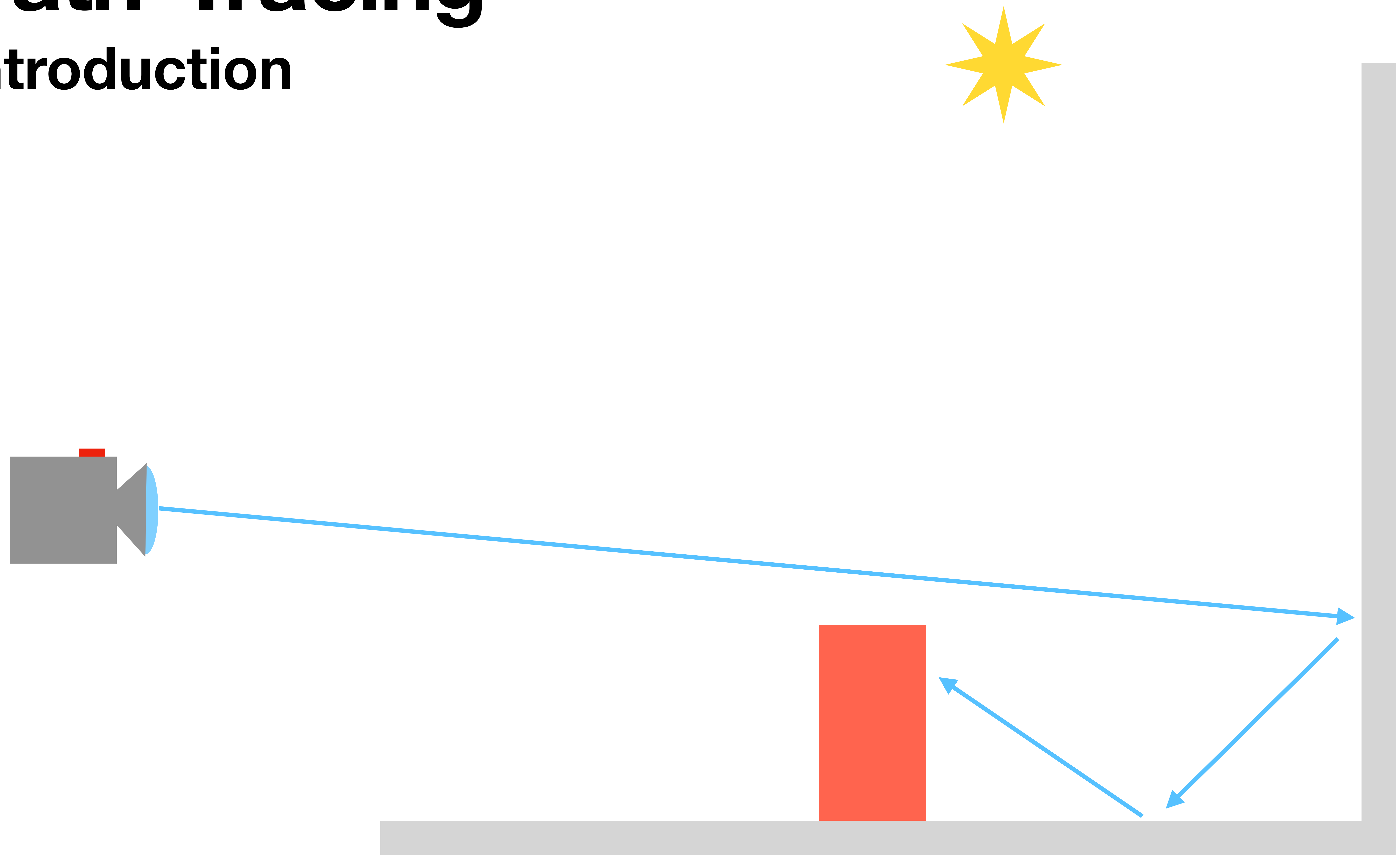
Path-Tracing

Introduction



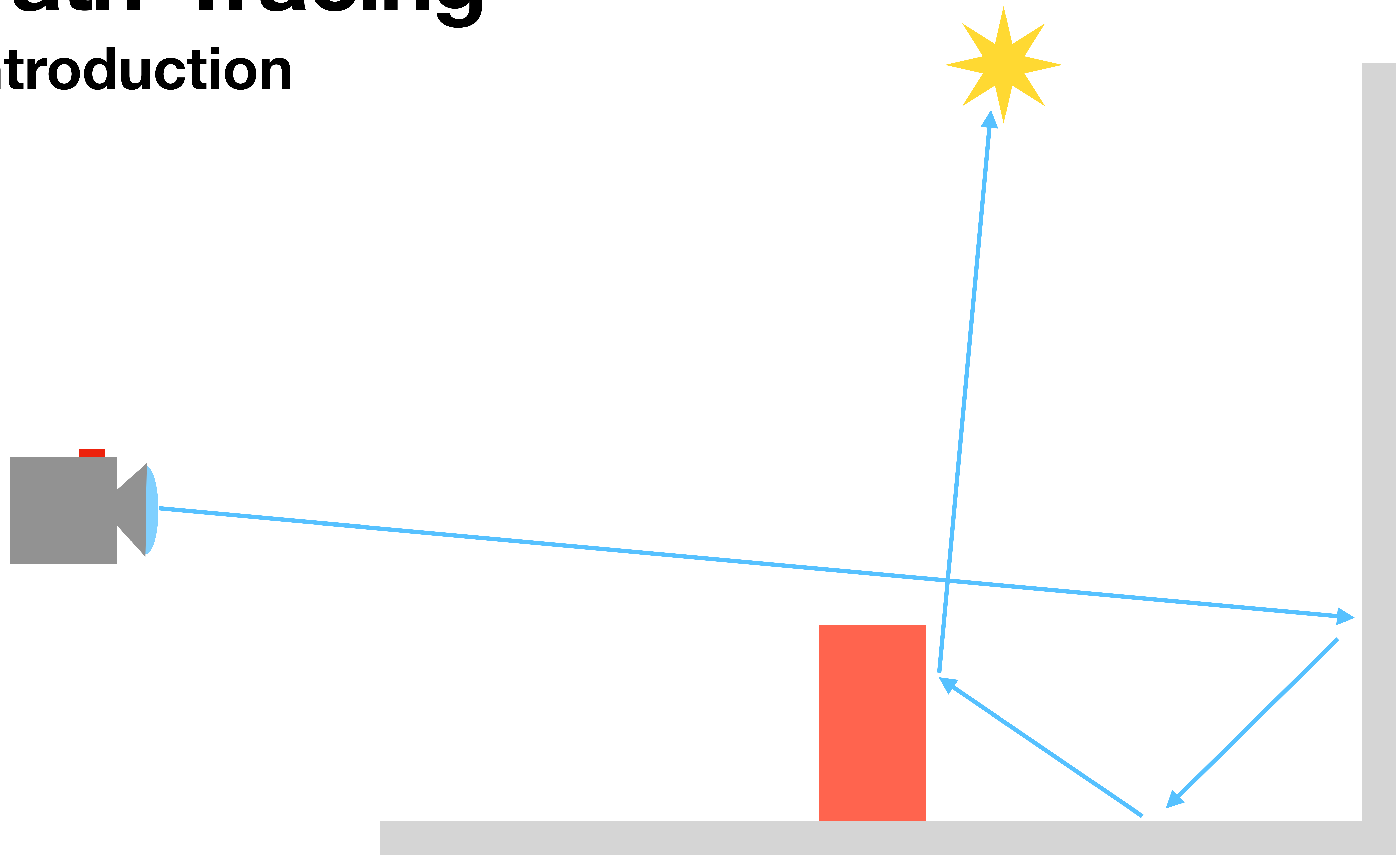
Path-Tracing

Introduction



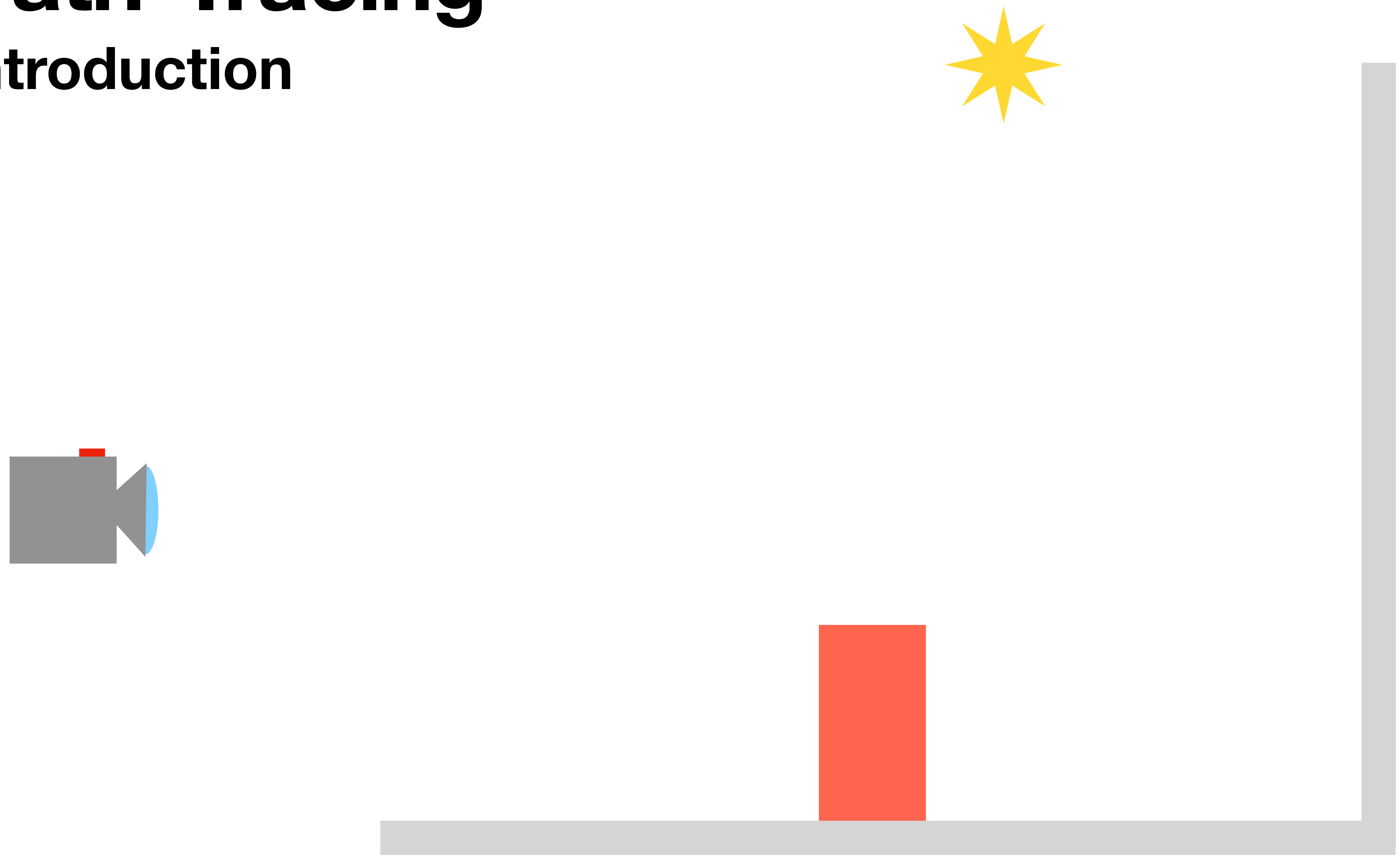
Path-Tracing

Introduction



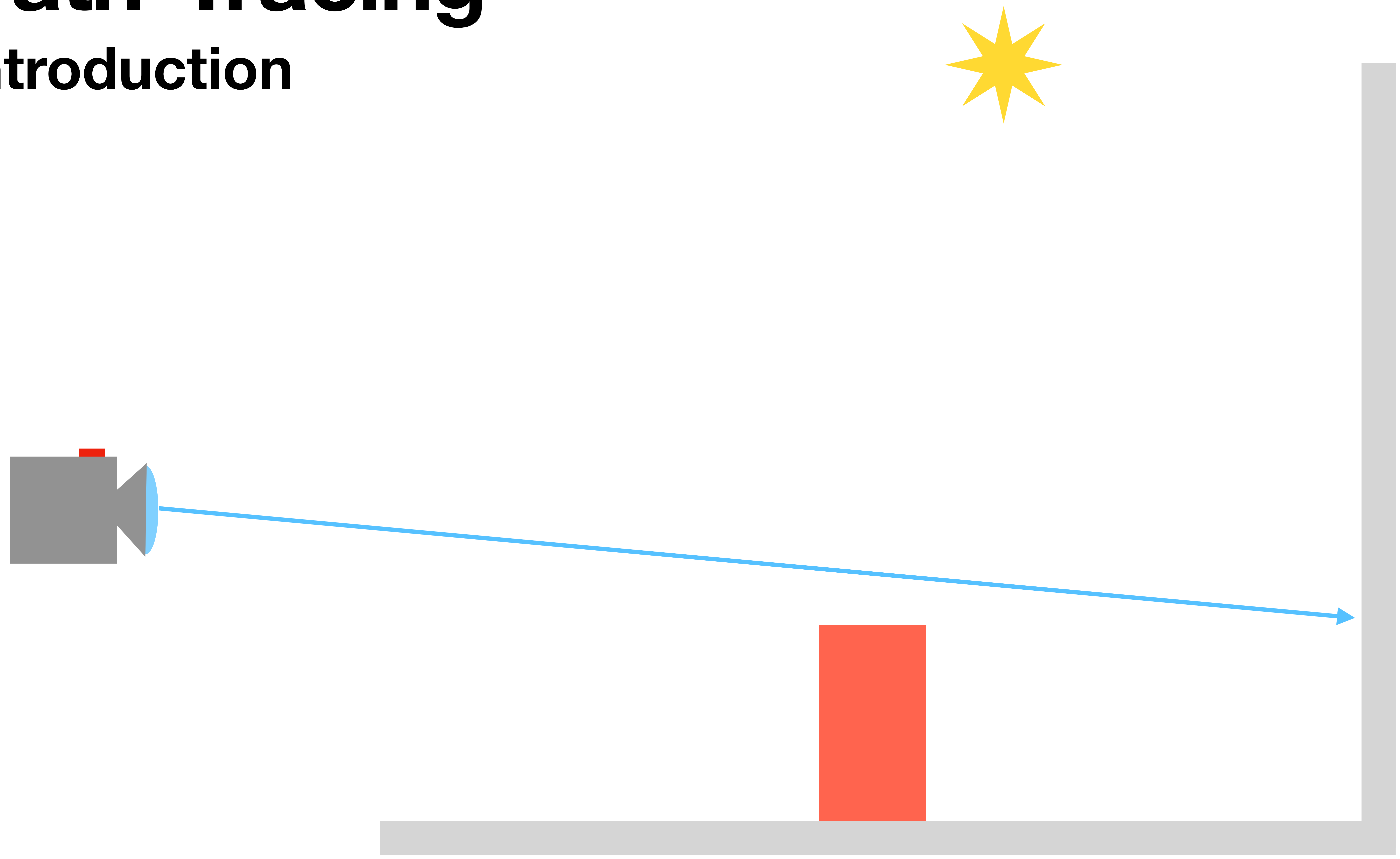
Path-Tracing

Introduction



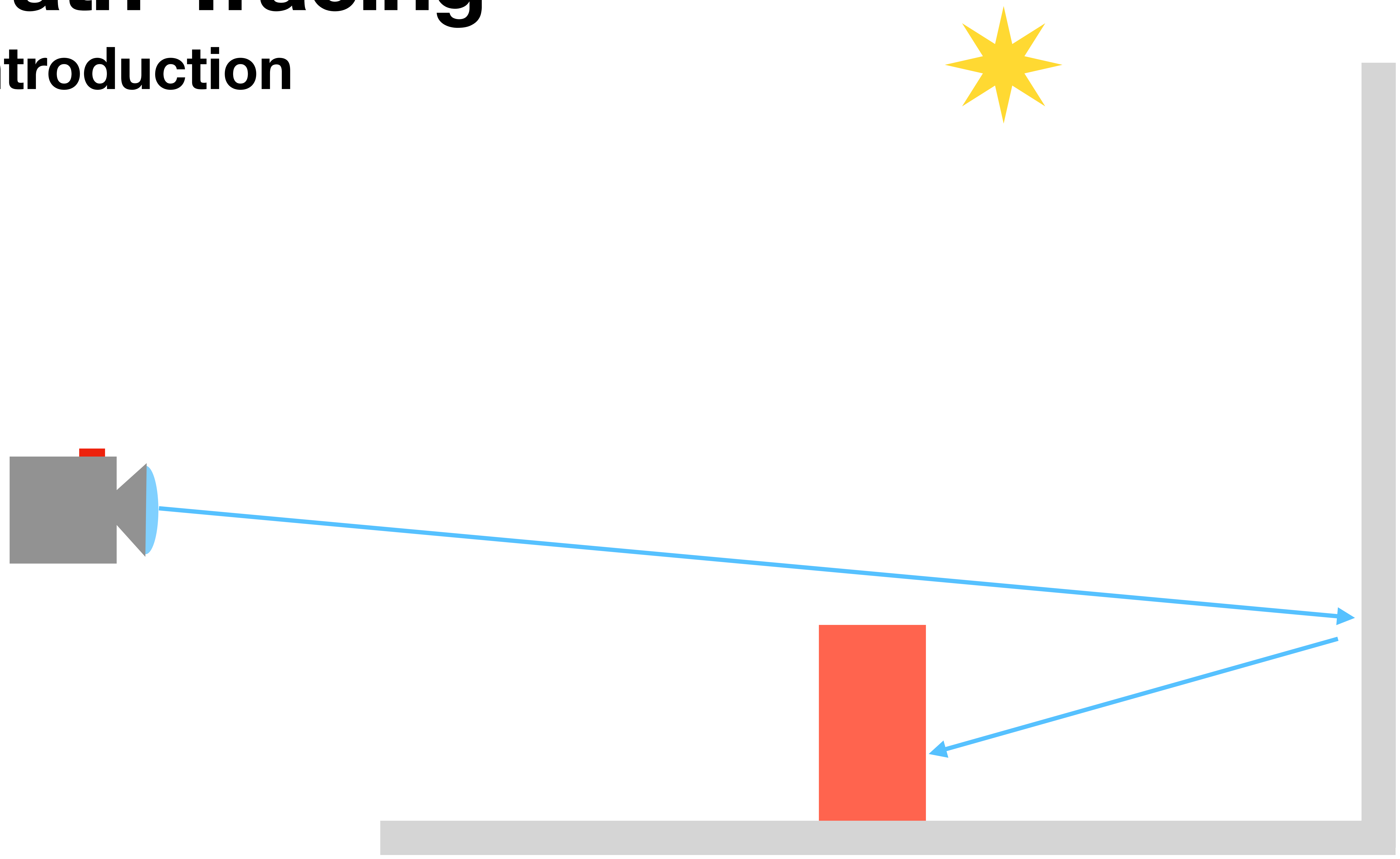
Path-Tracing

Introduction



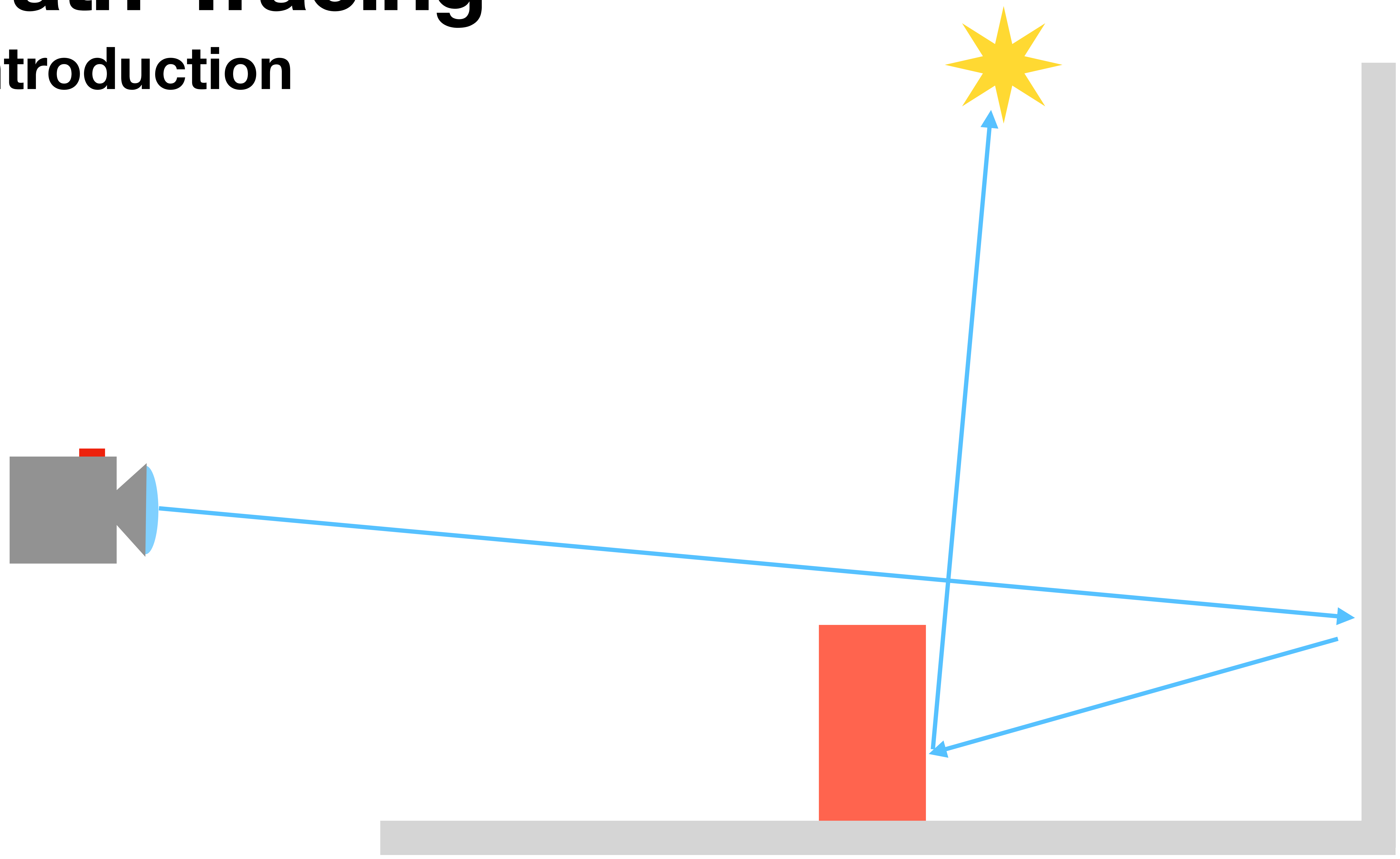
Path-Tracing

Introduction



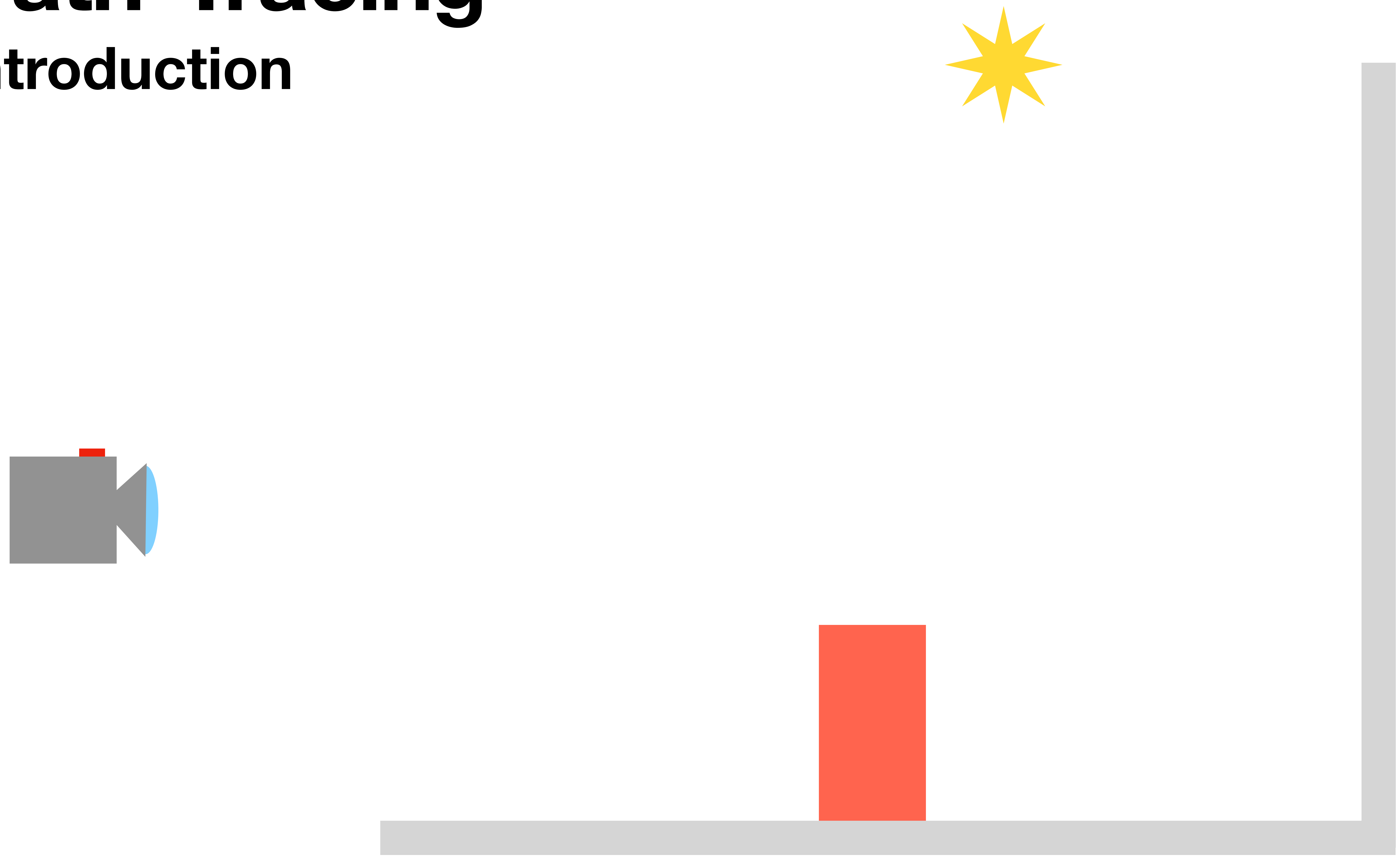
Path-Tracing

Introduction



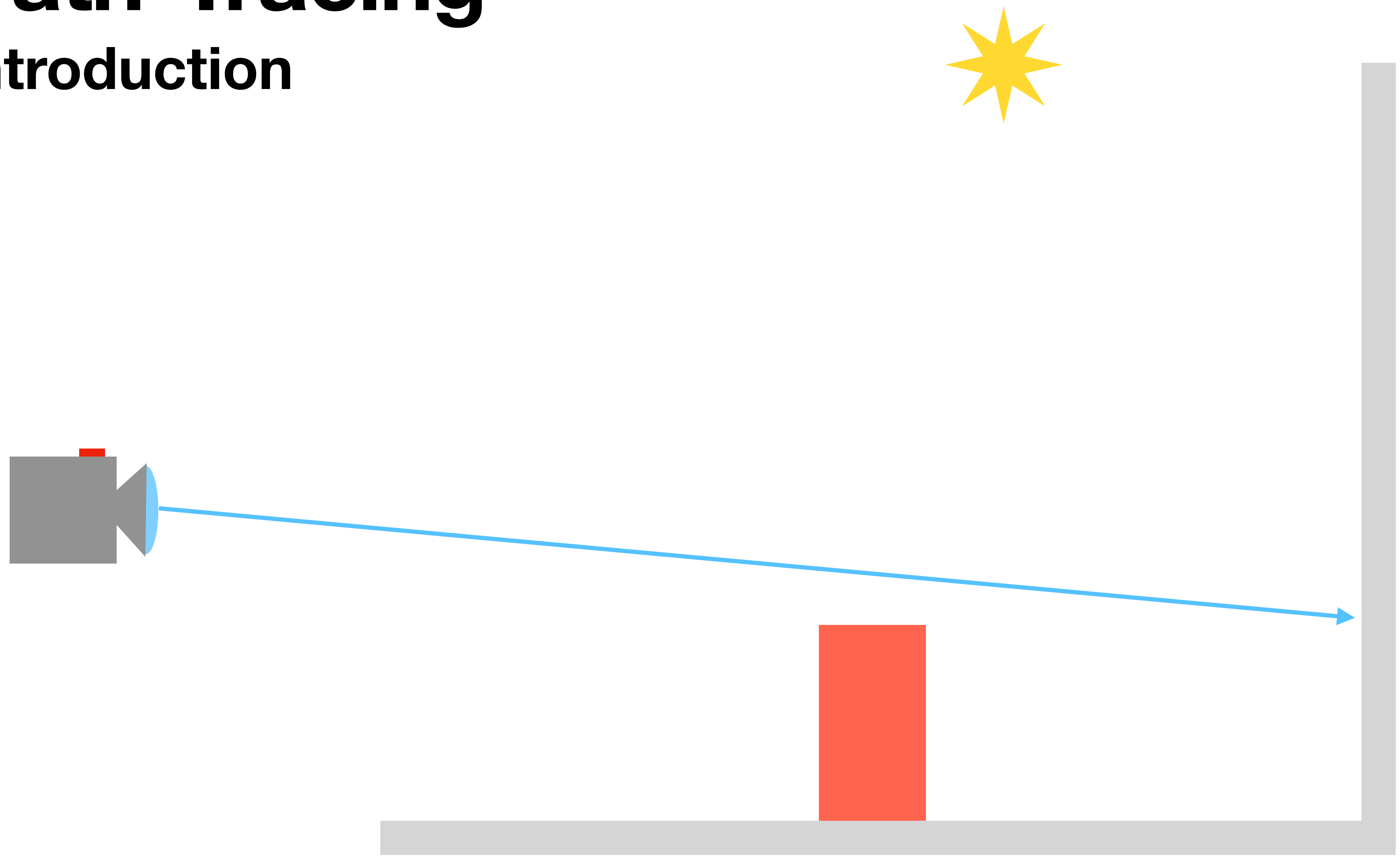
Path-Tracing

Introduction



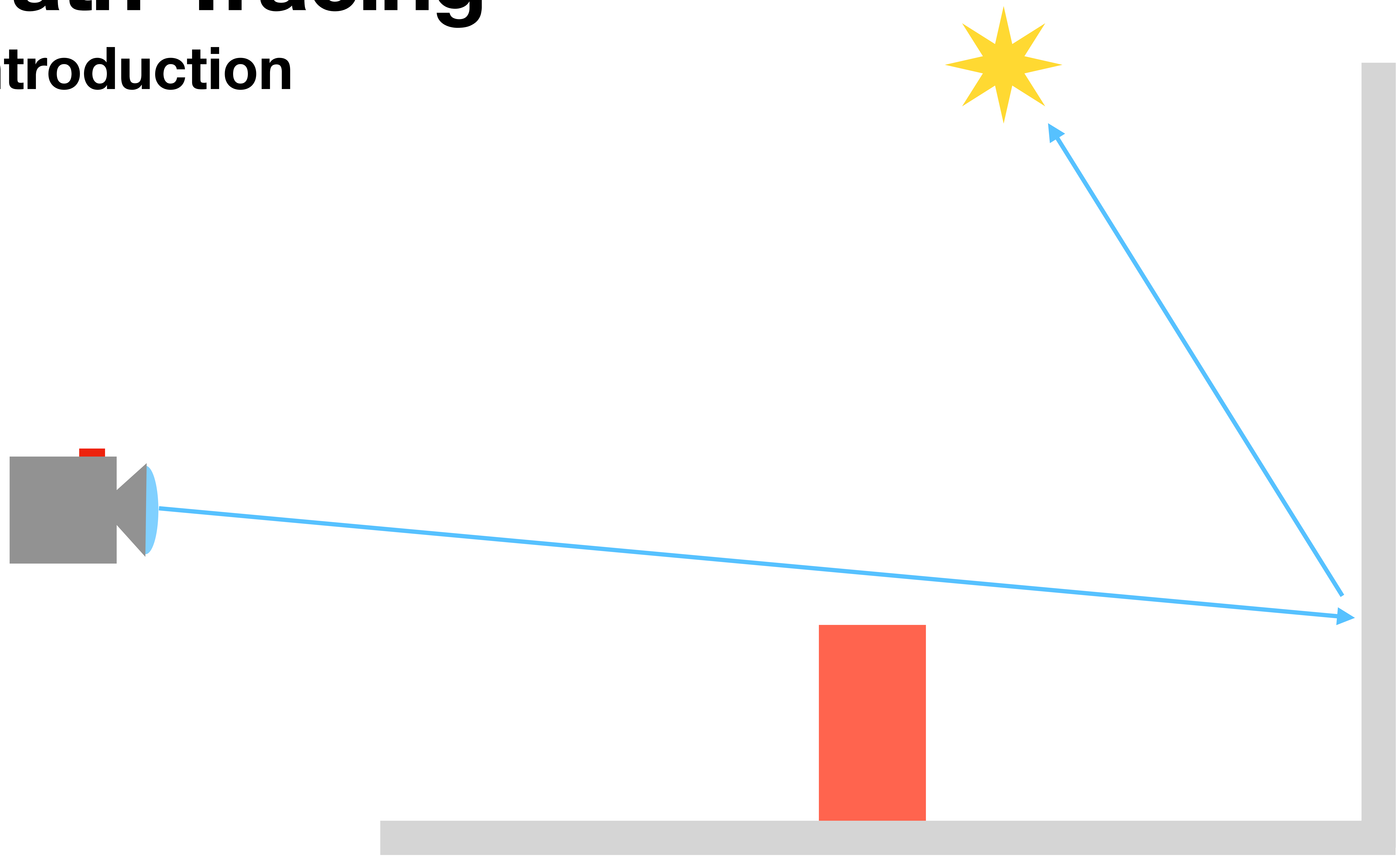
Path-Tracing

Introduction



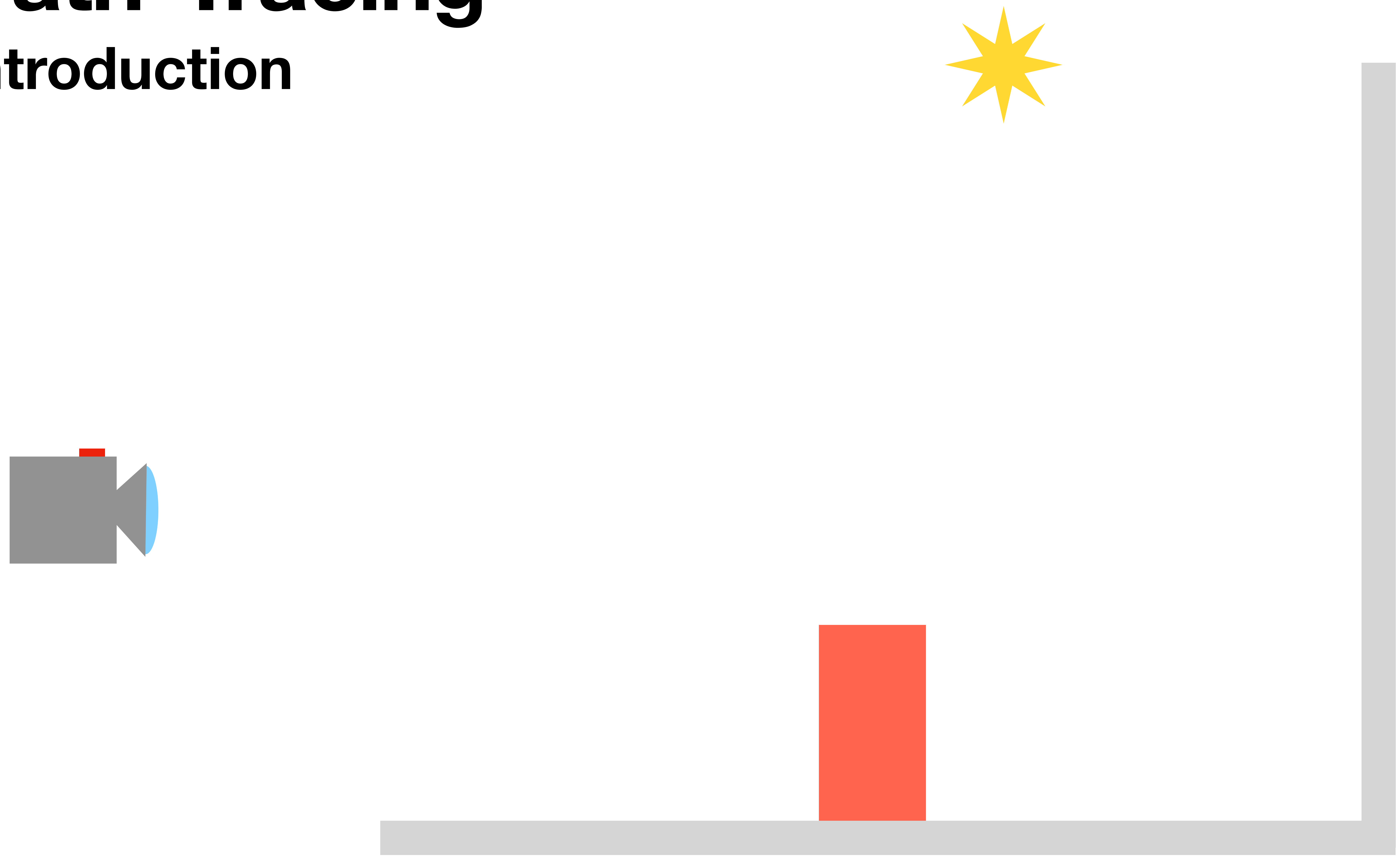
Path-Tracing

Introduction



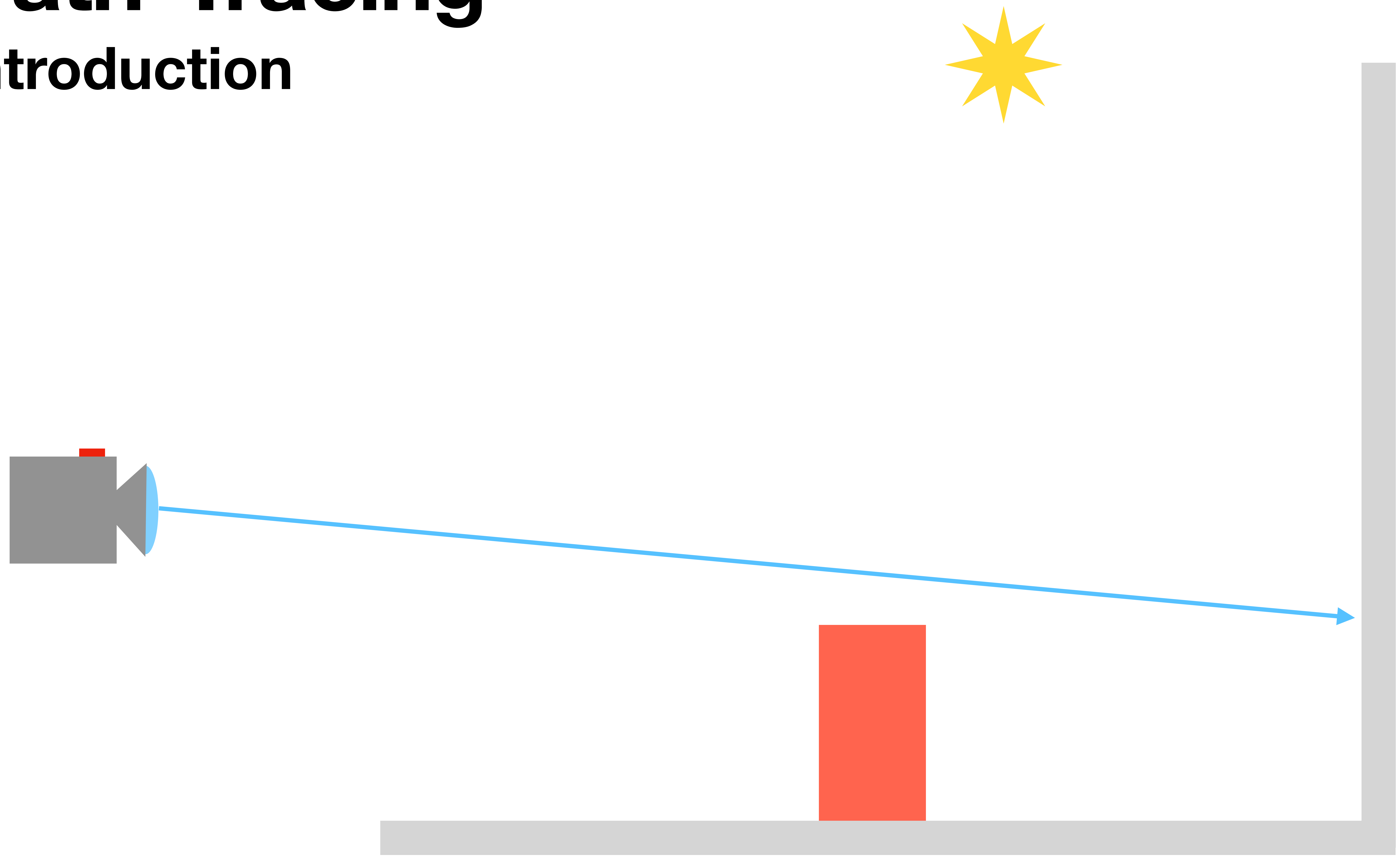
Path-Tracing

Introduction



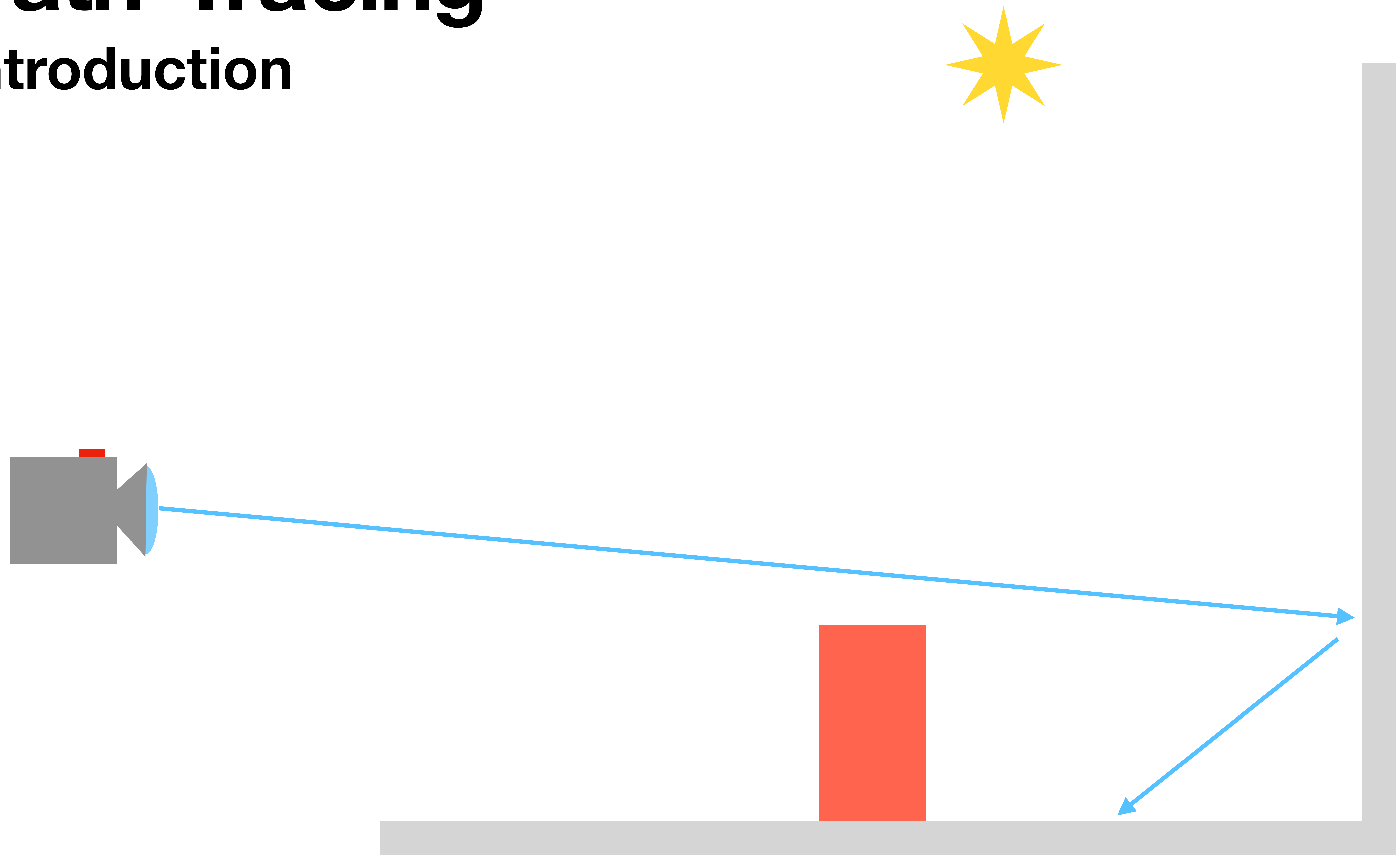
Path-Tracing

Introduction



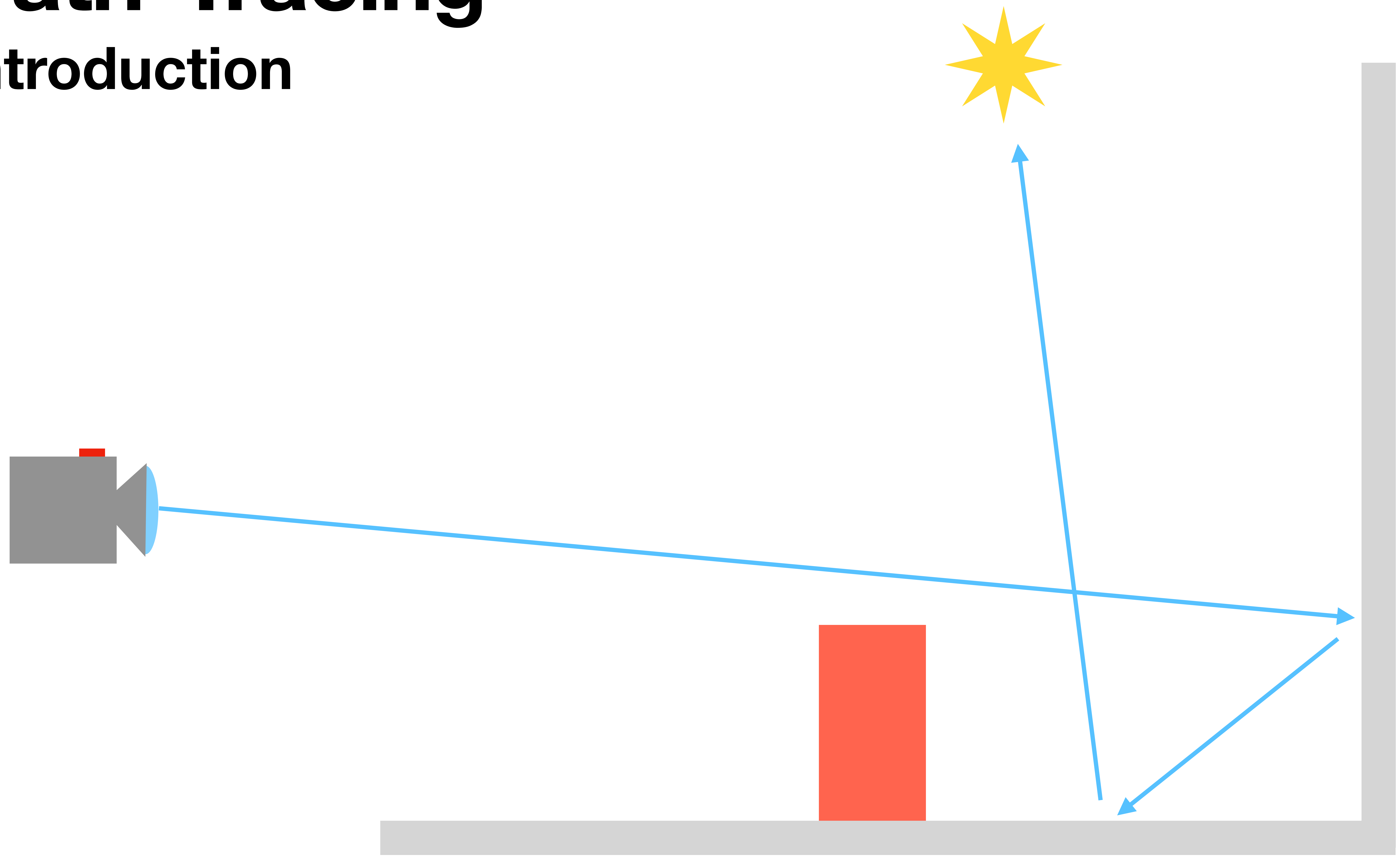
Path-Tracing

Introduction



Path-Tracing

Introduction



Path-Tracing

Monte-Carlo Techniques

- Techniques used:
  - Russian roulette —> to limit the length of paths.
  - Stratification.
  - Importance sampling:
    - 1D/2D distribution of light sources;
    - BRDF.
  - Metropolis.

Path-Tracing

Sampling the BRDF

- To sample the BRDF, we generate $\overrightarrow{\omega}_i$ directions randomly chosen according to its PDF:

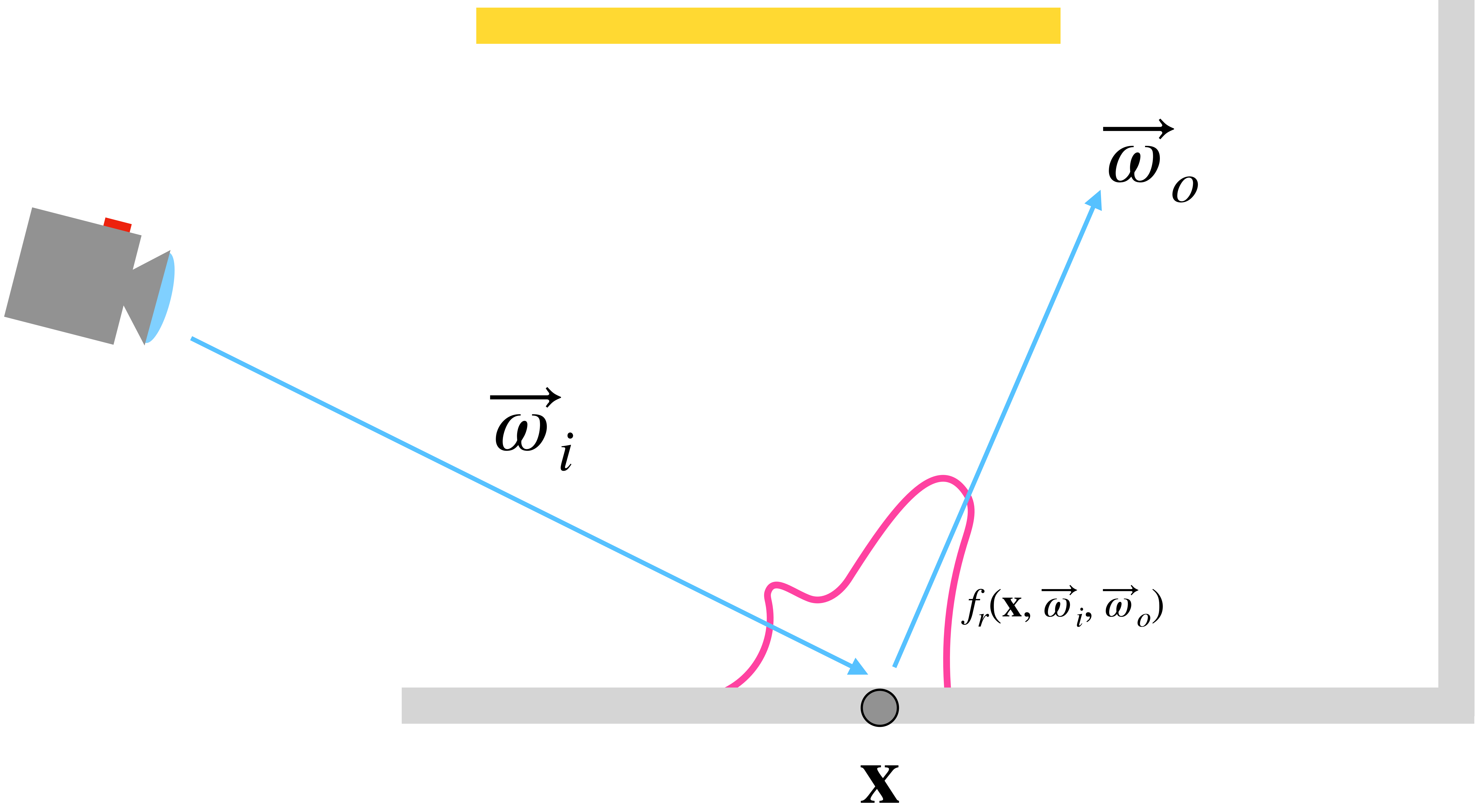
$$p(\overrightarrow{\omega}_i) \propto f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o).$$

- So, we compute our estimate as:

$$L_o(\mathbf{x}, \overrightarrow{\omega}_o) \approx \frac{f_r(\mathbf{x}, \overrightarrow{\omega}_i, \overrightarrow{\omega}_o) L_i(\mathbf{x}, \overrightarrow{\omega}_i)}{p(\overrightarrow{\omega}_i)}.$$

Path-Tracing

Sampling the BRDF



Path-Tracing

Sampling the BRDF

Image by Eric Veach

Path-Tracing

Sampling the Light Source

- To sample the light source, we generate random points, $\mathbf{x}_l$, on the light source according to its PDF:

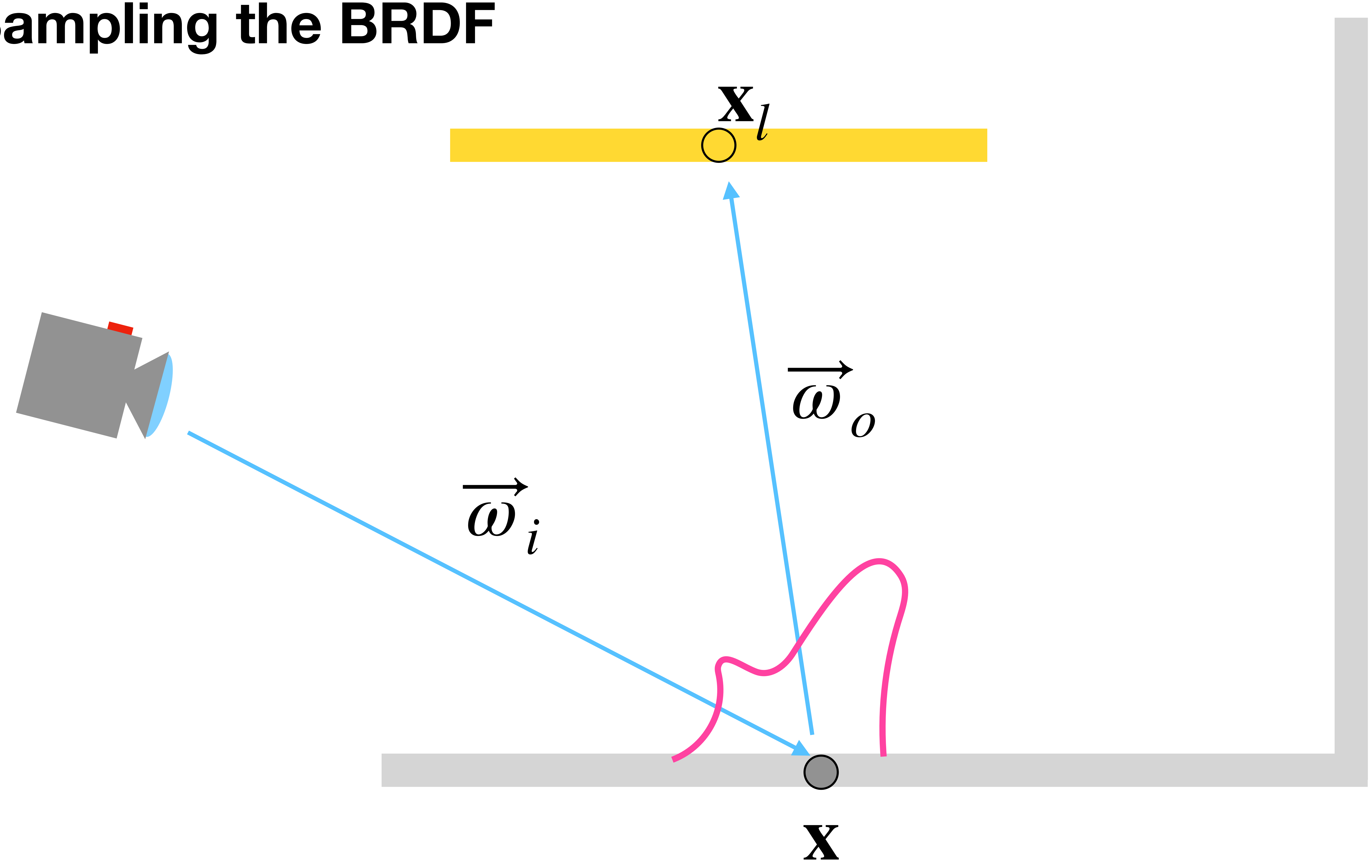
$$p(\mathbf{x}_l).$$

- So, we compute our estimate as:

$$L_o(\mathbf{x}, \overrightarrow{\omega}_o) \approx \frac{f_r(\mathbf{x}, \overrightarrow{\omega}'_i, \overrightarrow{\omega}_o) L_i(\mathbf{x}, \overrightarrow{\omega}'_i)}{p(\mathbf{x}_l)} \qquad \overrightarrow{\omega}'_i = \frac{\mathbf{x}_l - \mathbf{x}}{\|\mathbf{x}_l - \mathbf{x}\|}.$$

Path-Tracing

Sampling the BRDF



Path-Tracing

Sampling the Light Source

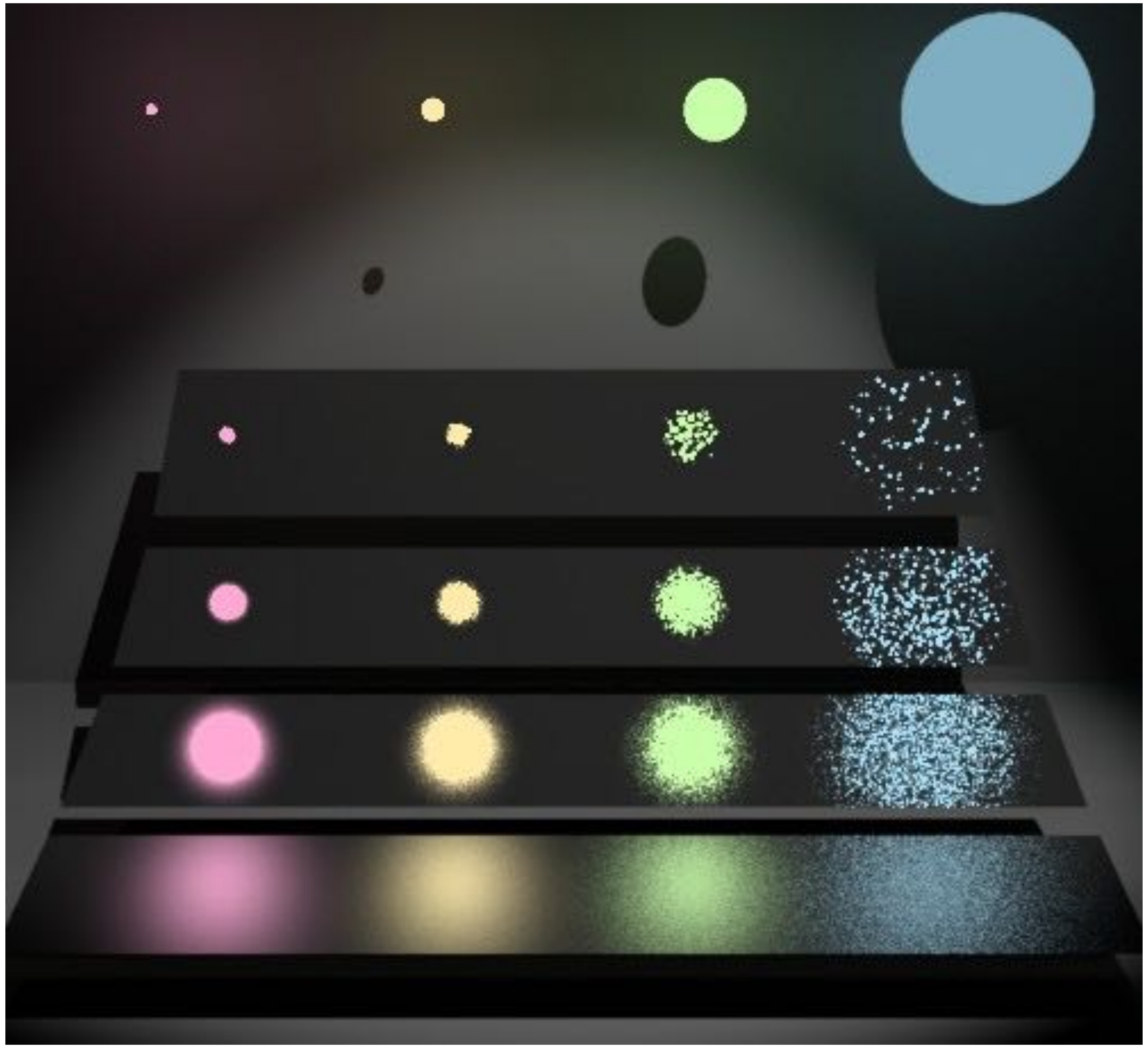


Image by Eric Veach

Path-Tracing

Multiple Importance Sampling (MIS)

- The naive solution would be to average the two estimations:
  - However, variance is additive, so we do not decrease it!
- The main idea of Multiple Importance Sampling (MIS) is to:
  - Draw samples from different distributions;
  - Mix all these samples using weights:
    - These weights should remove large peaks of variance when we have differences between our estimation and the distribution.

.

Path-Tracing

MIS

- In general, we may have K distributions, q_i, and we generate n_j samples $\mathbf{x}_{i,j} \sim q_j$ for each distribution.
- In this case, our estimation is:

$$\hat{\mu} = \sum_{j=1}^{K} \frac{1}{n_j} \sum_{i=1}^{n_j} \omega_j(\mathbf{x}_{i,j}) \frac{f(\mathbf{x}_{i,j})p(\mathbf{x}_{i,j})}{q_j(\mathbf{x}_{i,j})}.$$

- The weighting function, $\omega(\mathbf{x}) \geq 0$, is normalized:

$$\sum_{j=1}^{K} \omega(\mathbf{x}) = 1.$$

- Balance heuristic $\omega_j(\mathbf{x}) \propto n_j q_j(\mathbf{x})$:

$$\omega_j(\mathbf{x}) = \frac{n_j q_j(\mathbf{x})}{\sum_{i=1}^{K} n_i q_i(\mathbf{x})}.$$

Path-Tracing
MIS

- What's about its variance?

$$\text{Var}(\hat{\mu}_{BH}) = \text{Var}(\hat{\mu}) + \left(\frac{1}{\min_j n_j} - \frac{1}{\sum_j n_j}\right)\mu^2.$$

- Other heuristics? Yes
  - Power Heuristic: $\omega_j(\mathbf{x}) \propto \left(n_j q_j(\mathbf{x})\right)^\beta \quad \beta \geq 1.$

Path-Tracing

MIS Example

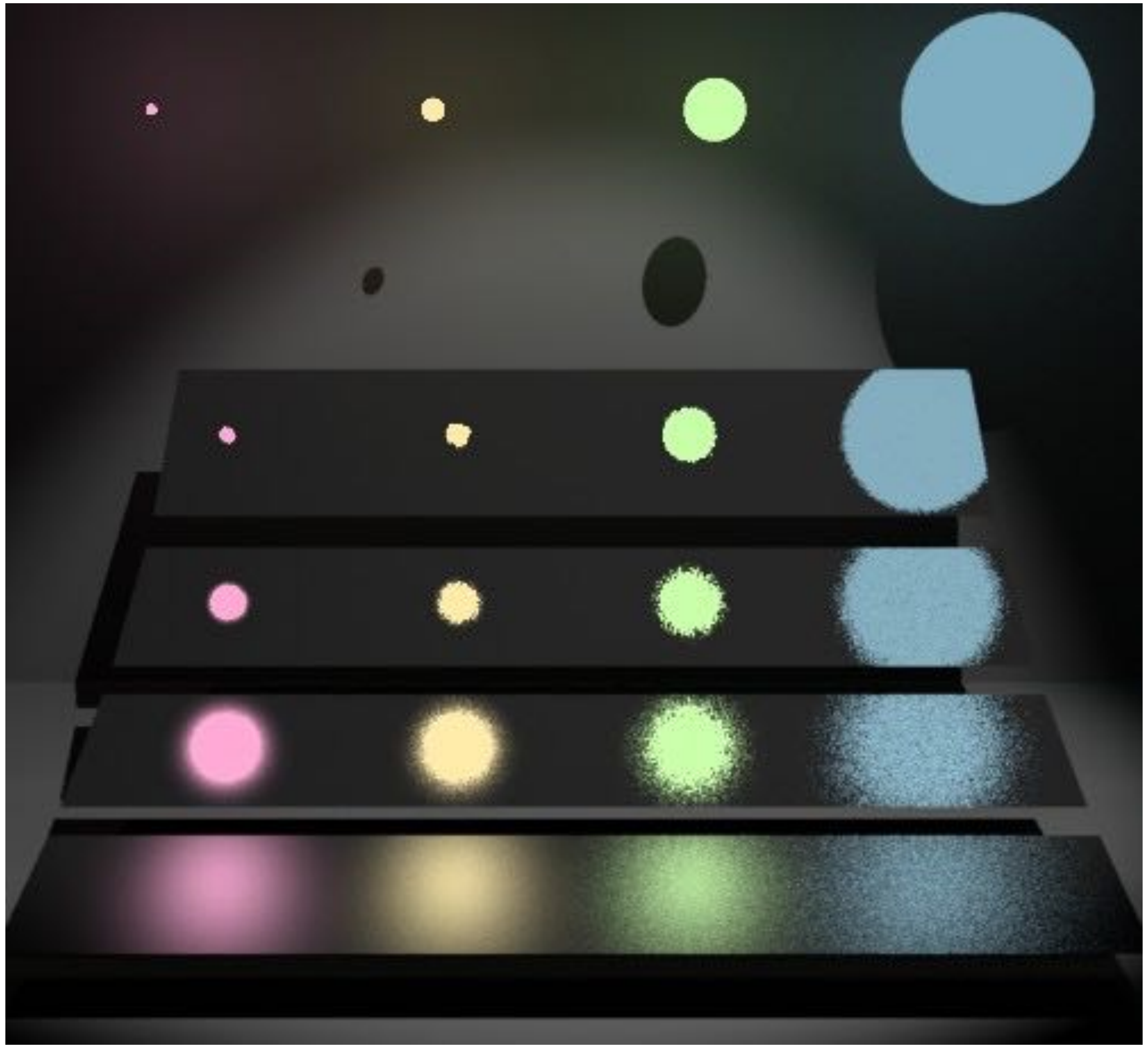


Image by Eric Veach

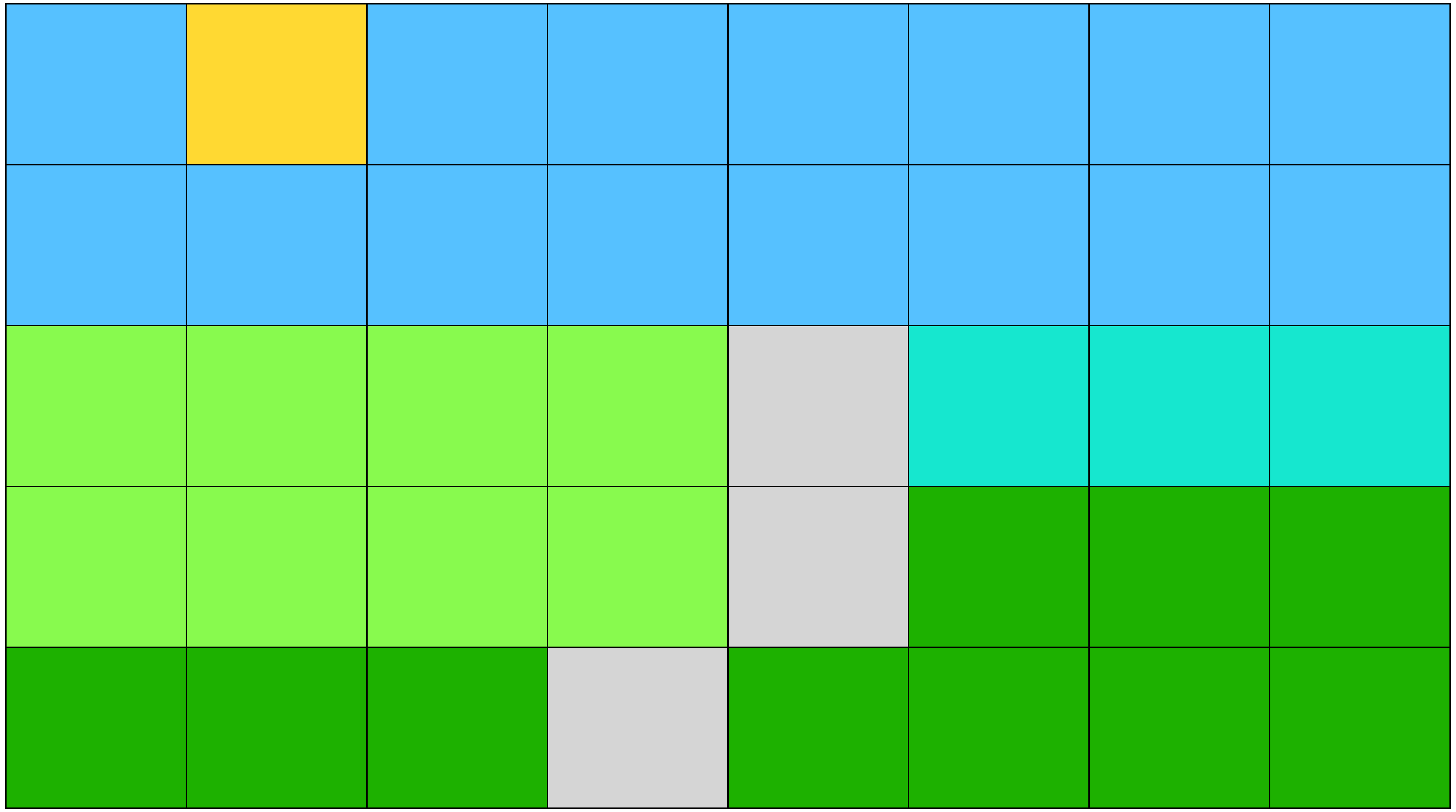
Path-Tracing

Complex Light Sources

- Typically, to have convincing light sources, they have to be spatially varying possibly using scene-referred photographs (i.e., calibrated):

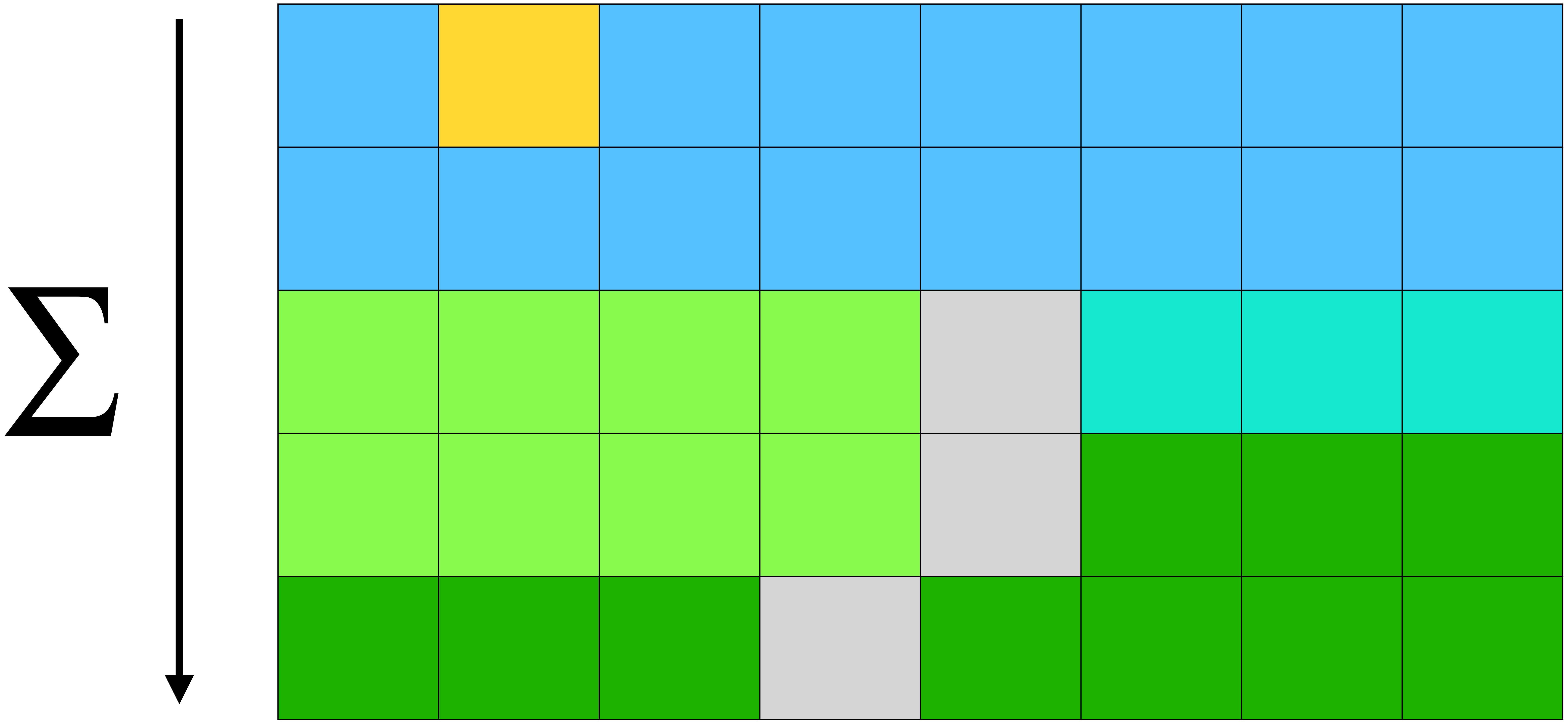
Path-Tracing

Complex Light Sources



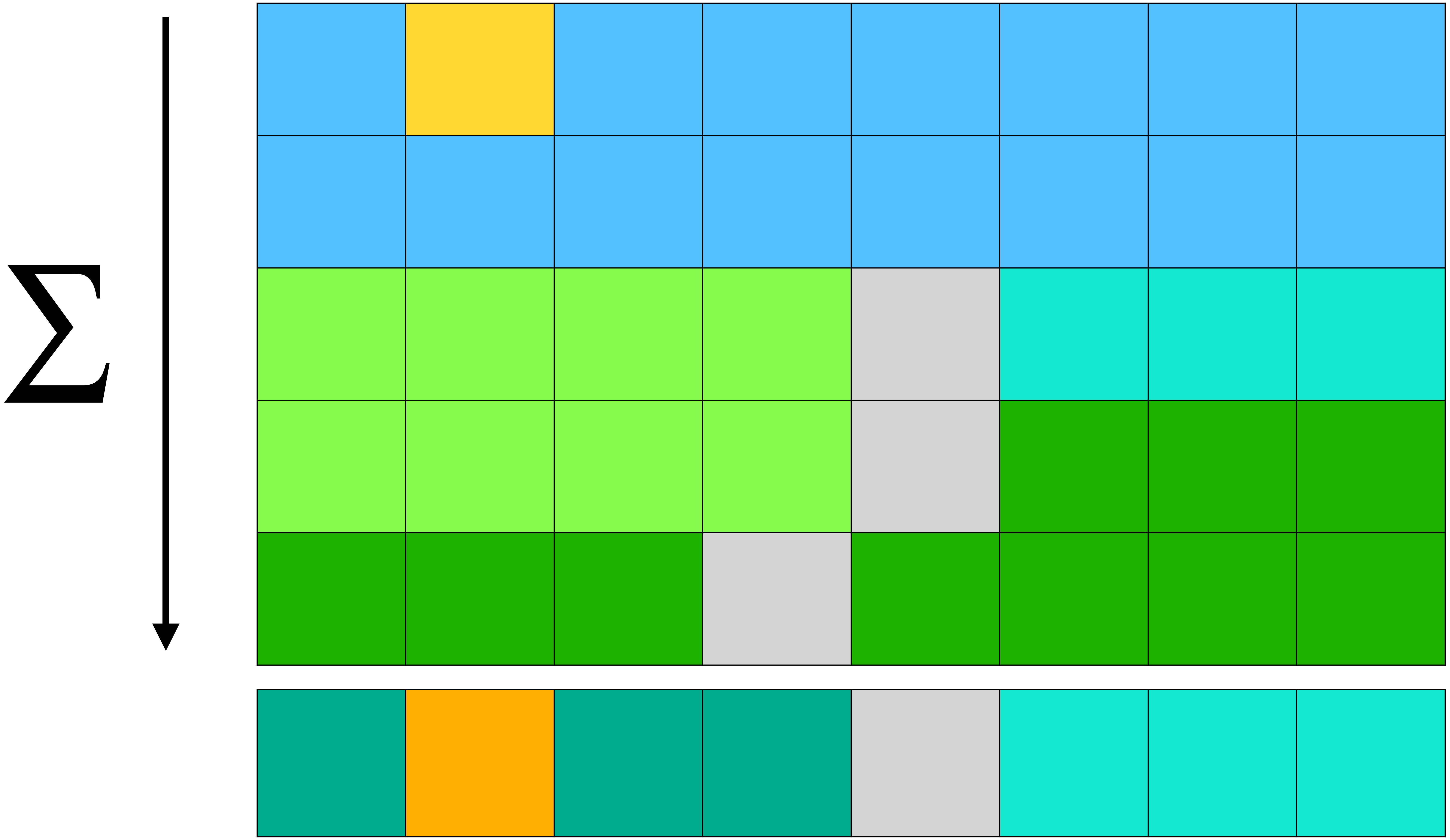
Path-Tracing

Complex Light Sources



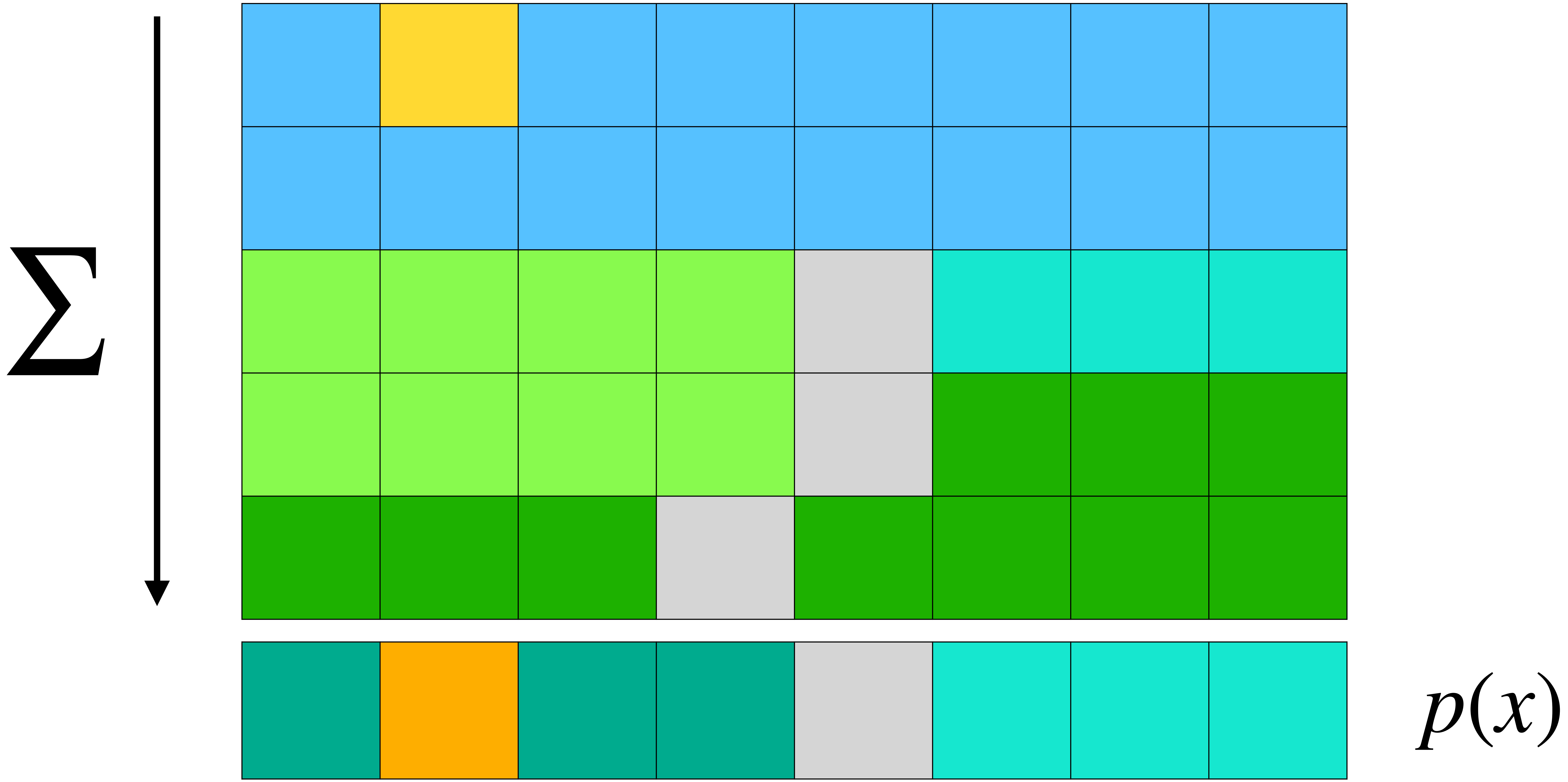
Path-Tracing

Complex Light Sources



Path-Tracing

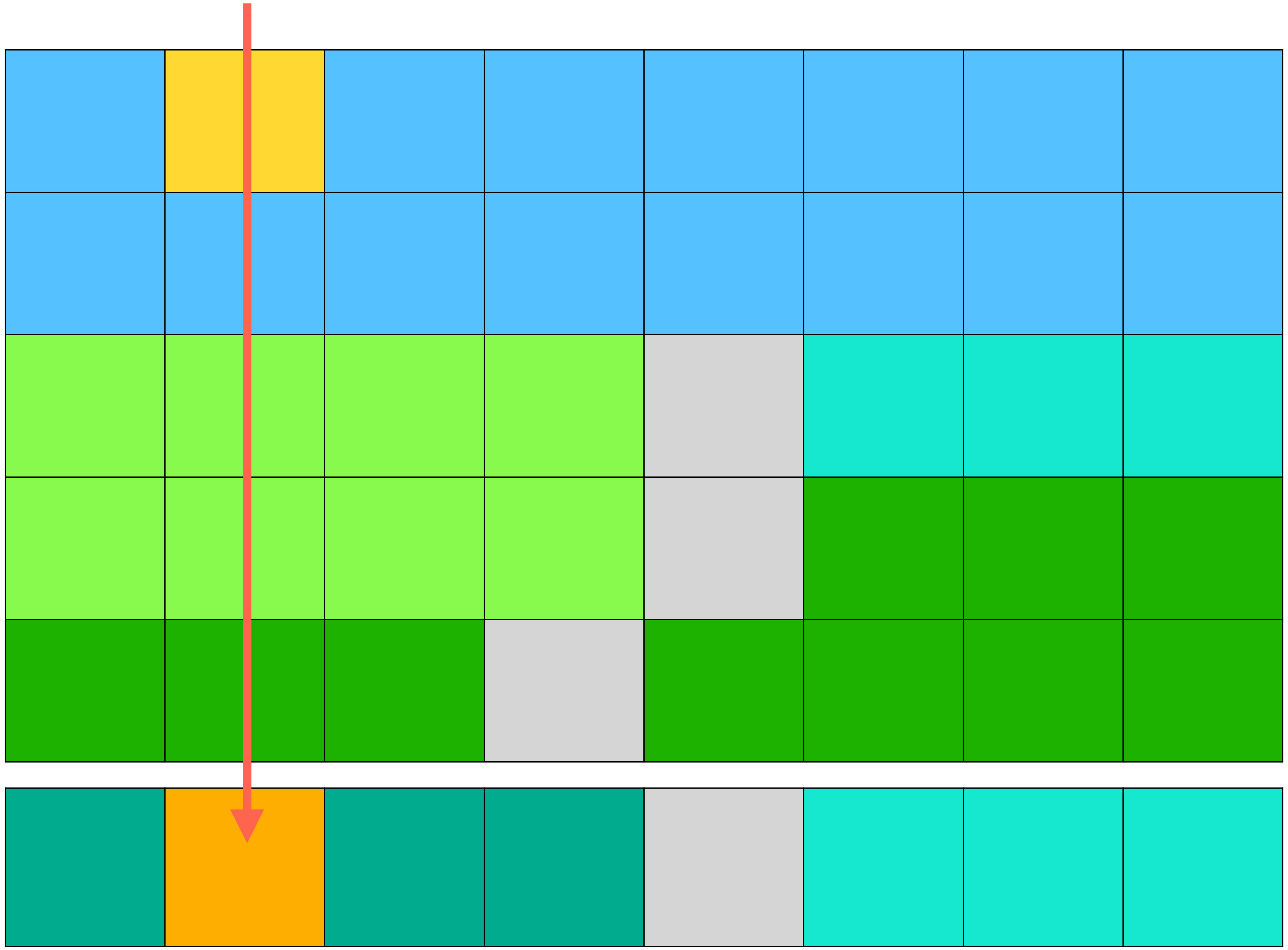
Complex Light Sources



Path-Tracing

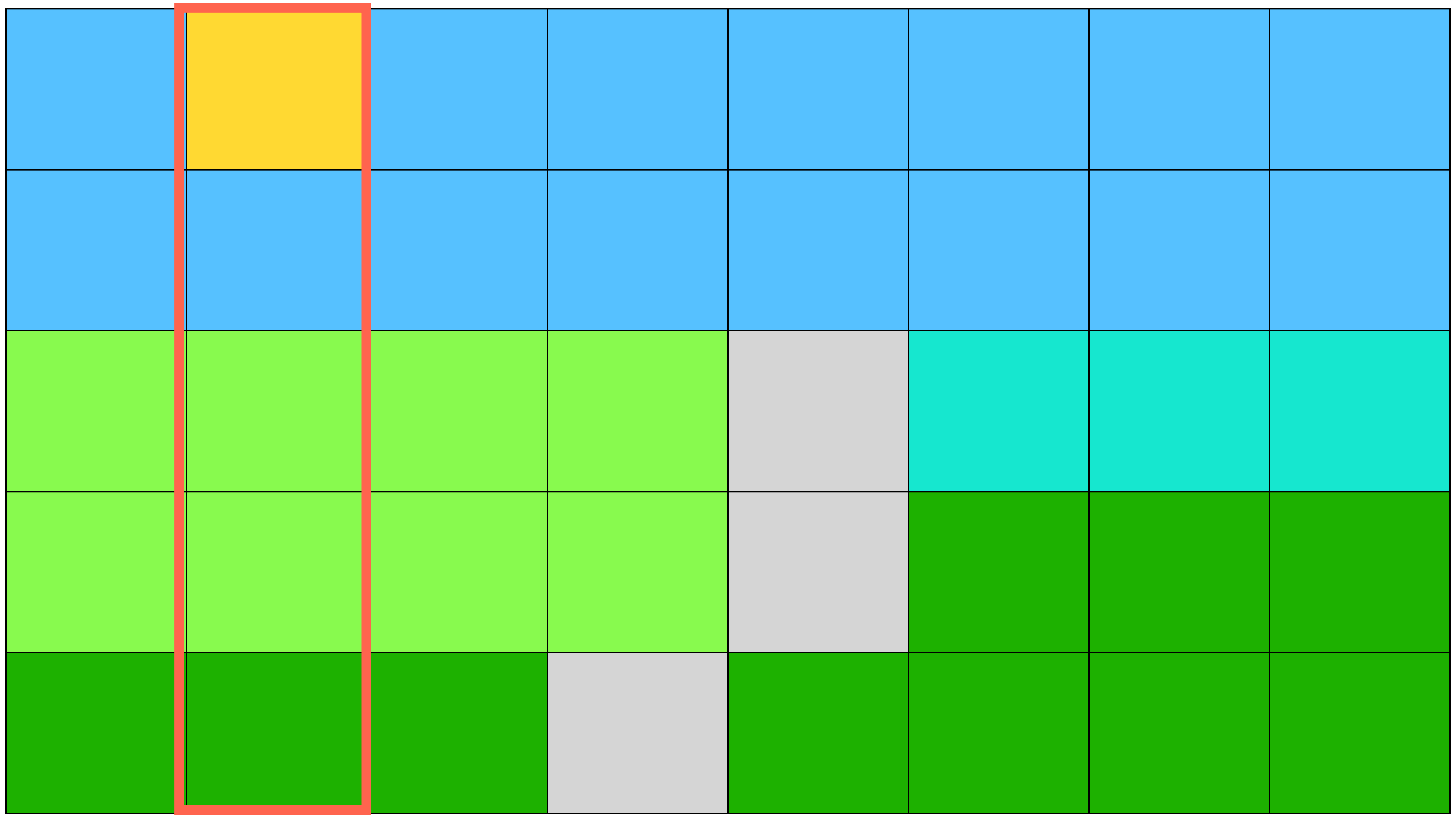
Complex Light Sources

$$X \sim p(x)$$



Path-Tracing

Complex Light Sources



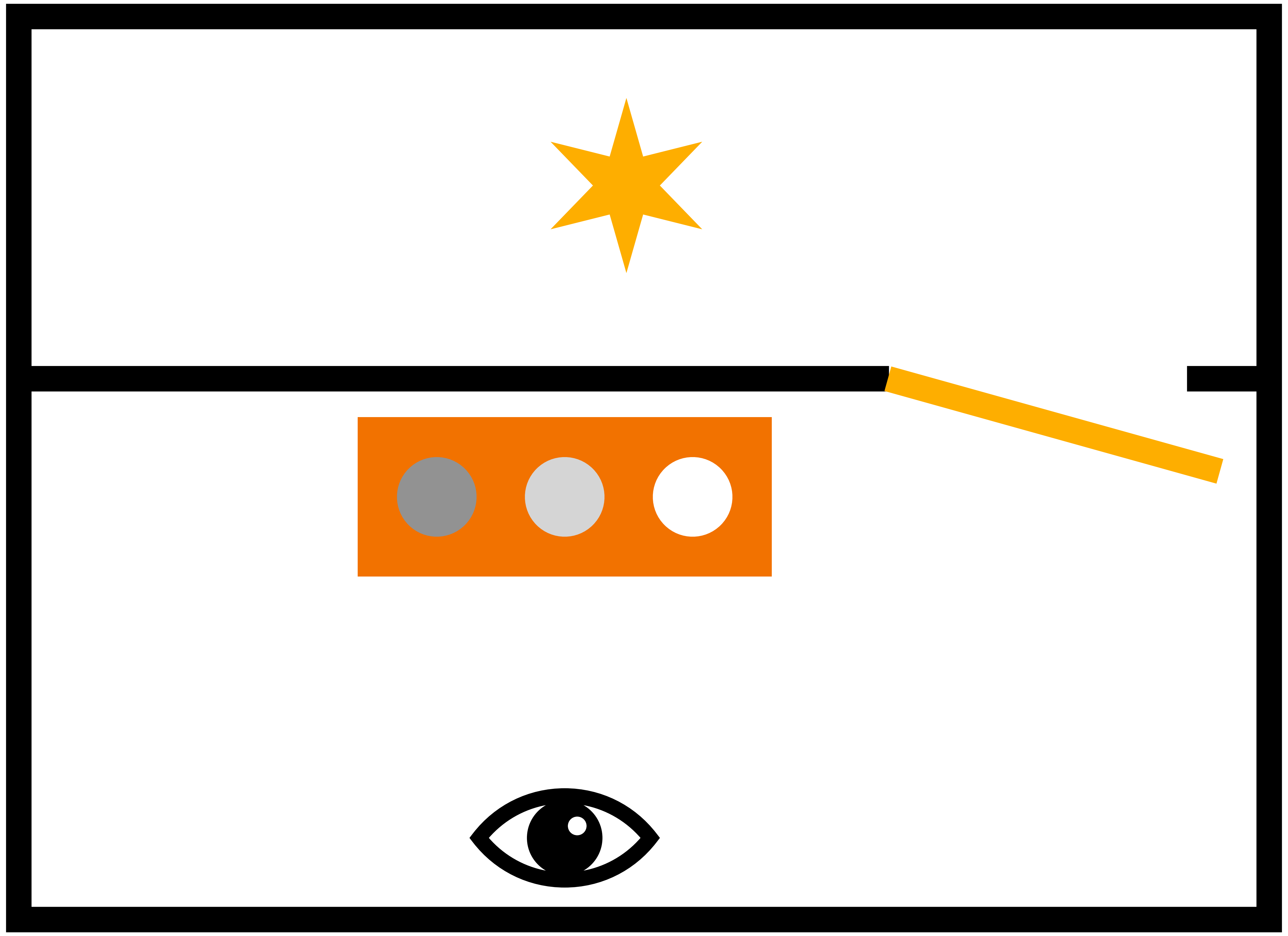
$p(y \mid x)$

Path-Tracing

Complex Light Sources: Example

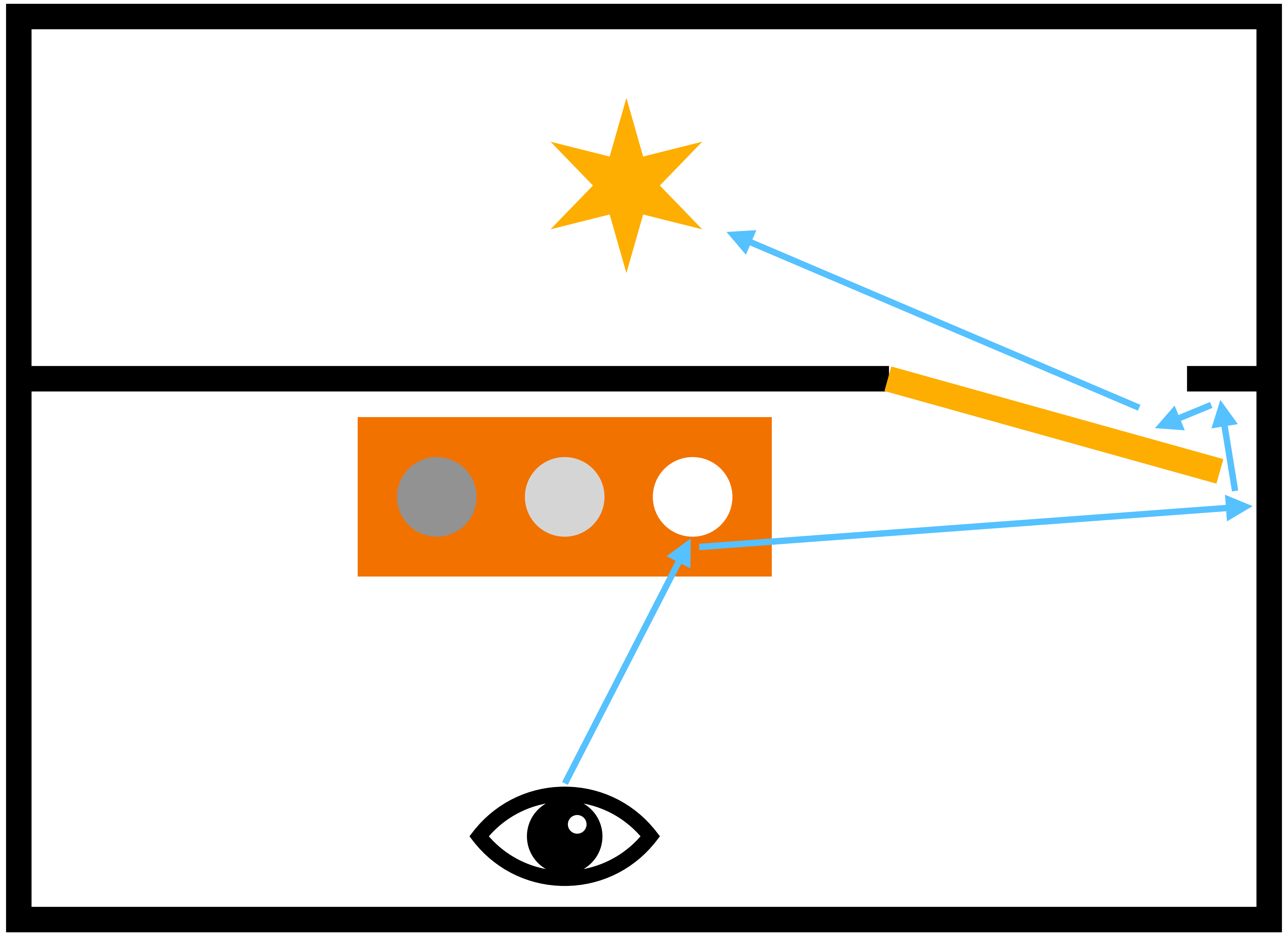
Path-Tracing

Metropolis Sampling



Path-Tracing

Metropolis Sampling



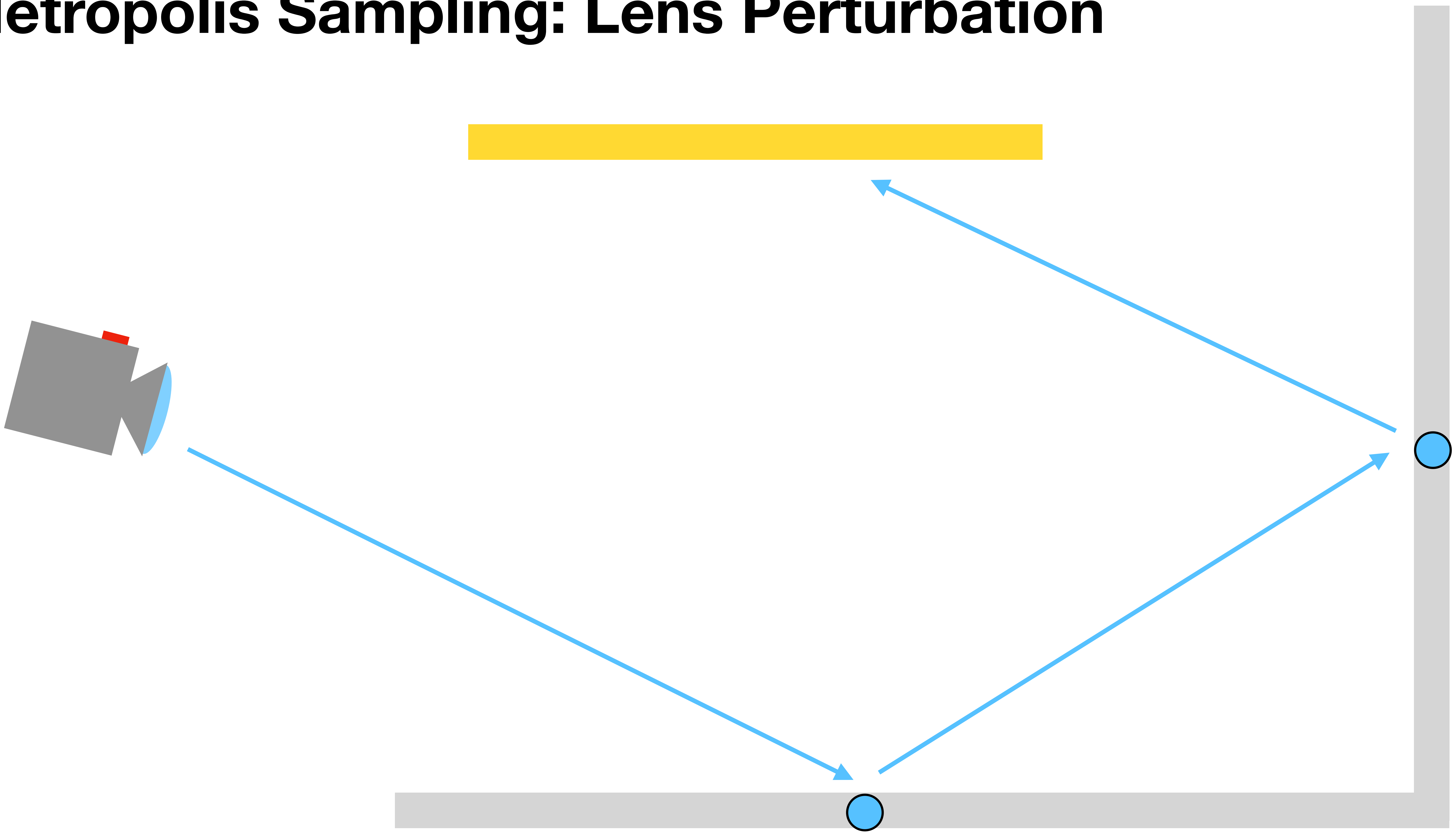
Path-Tracing

Metropolis Sampling

Image by Eric Veach

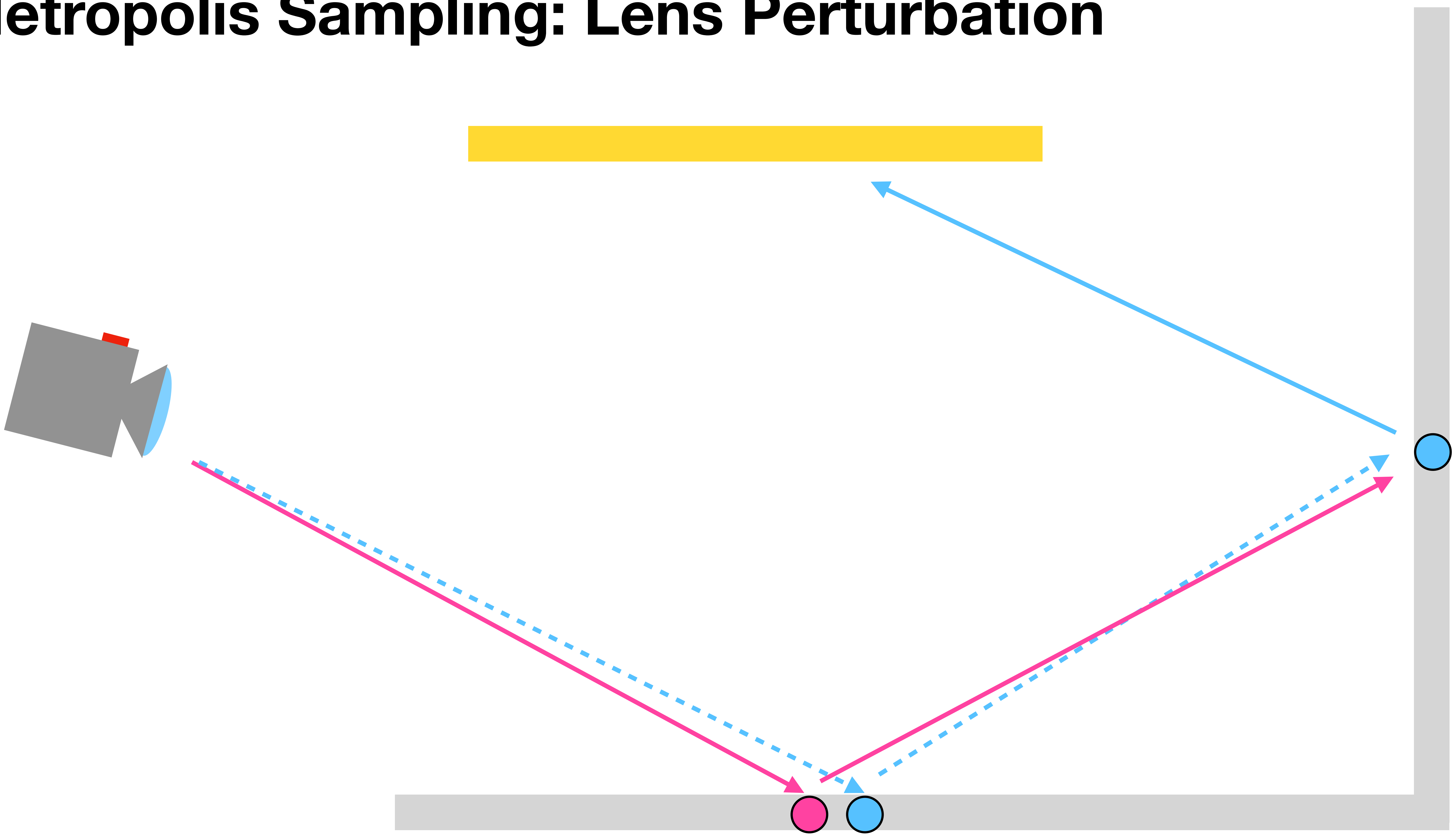
Path-Tracing

Metropolis Sampling: Lens Perturbation



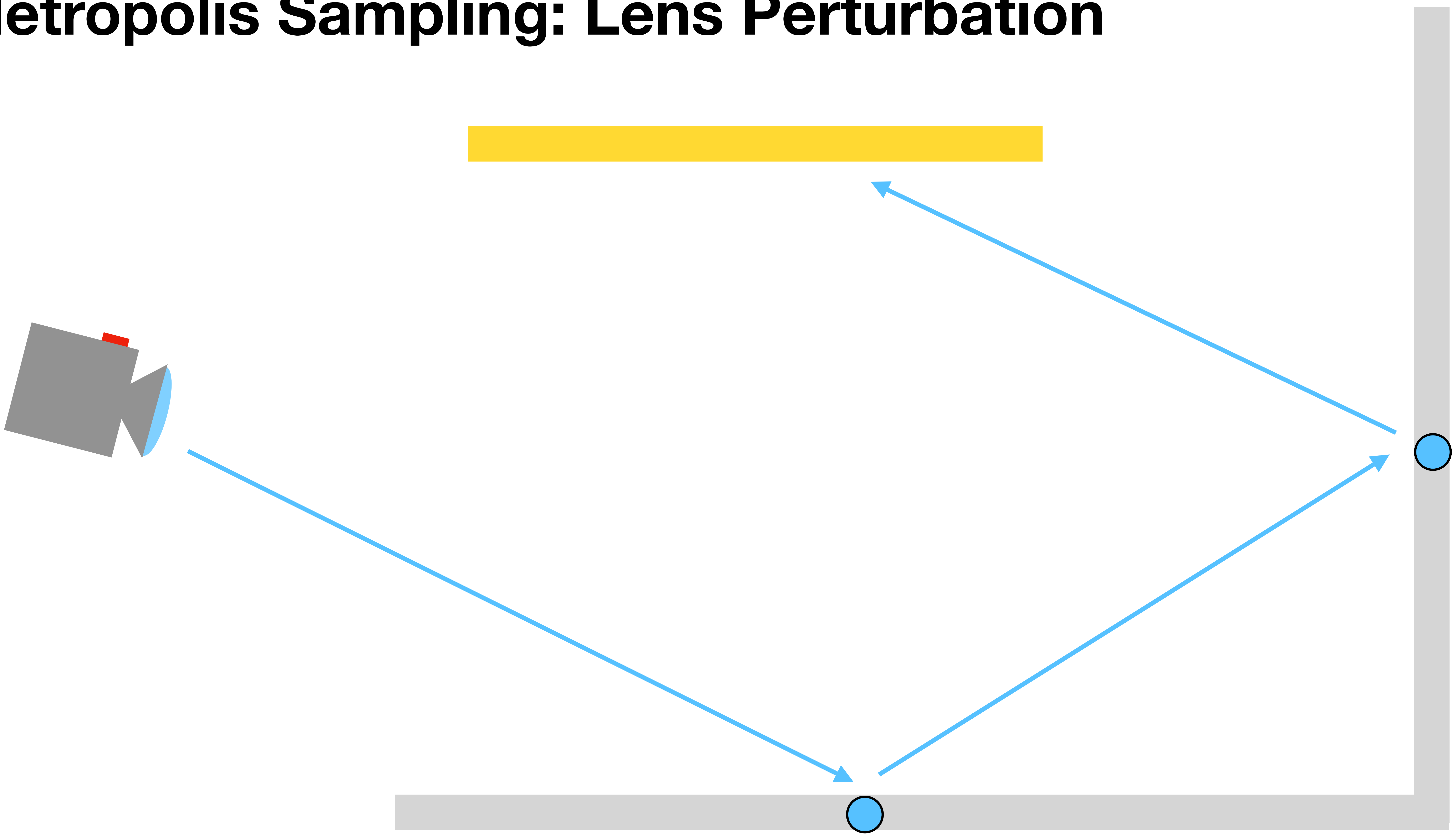
Path-Tracing

Metropolis Sampling: Lens Perturbation



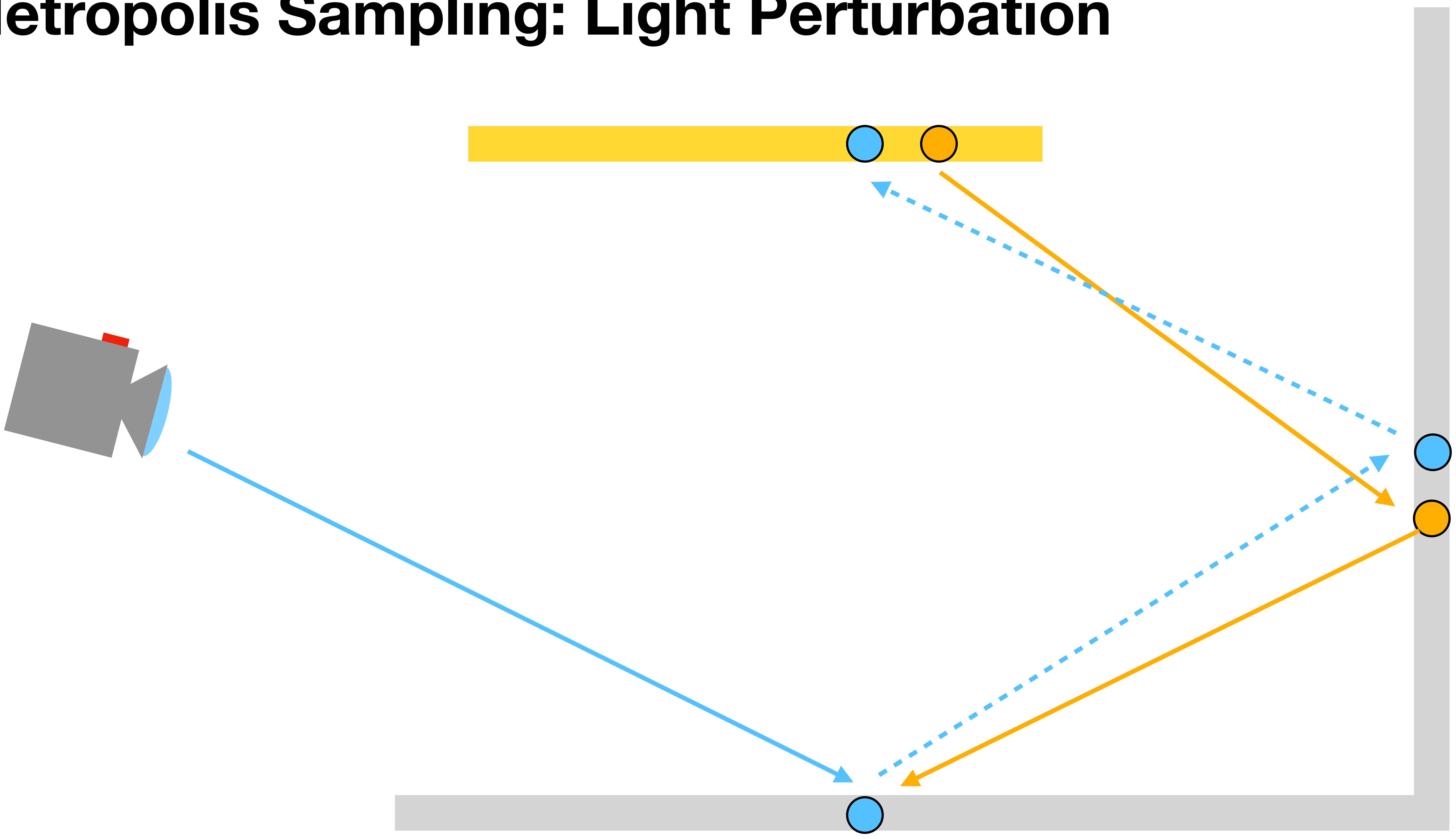
Path-Tracing

Metropolis Sampling: Lens Perturbation



Path-Tracing

Metropolis Sampling: Light Perturbation



Path-Tracing

Metropolis Sampling

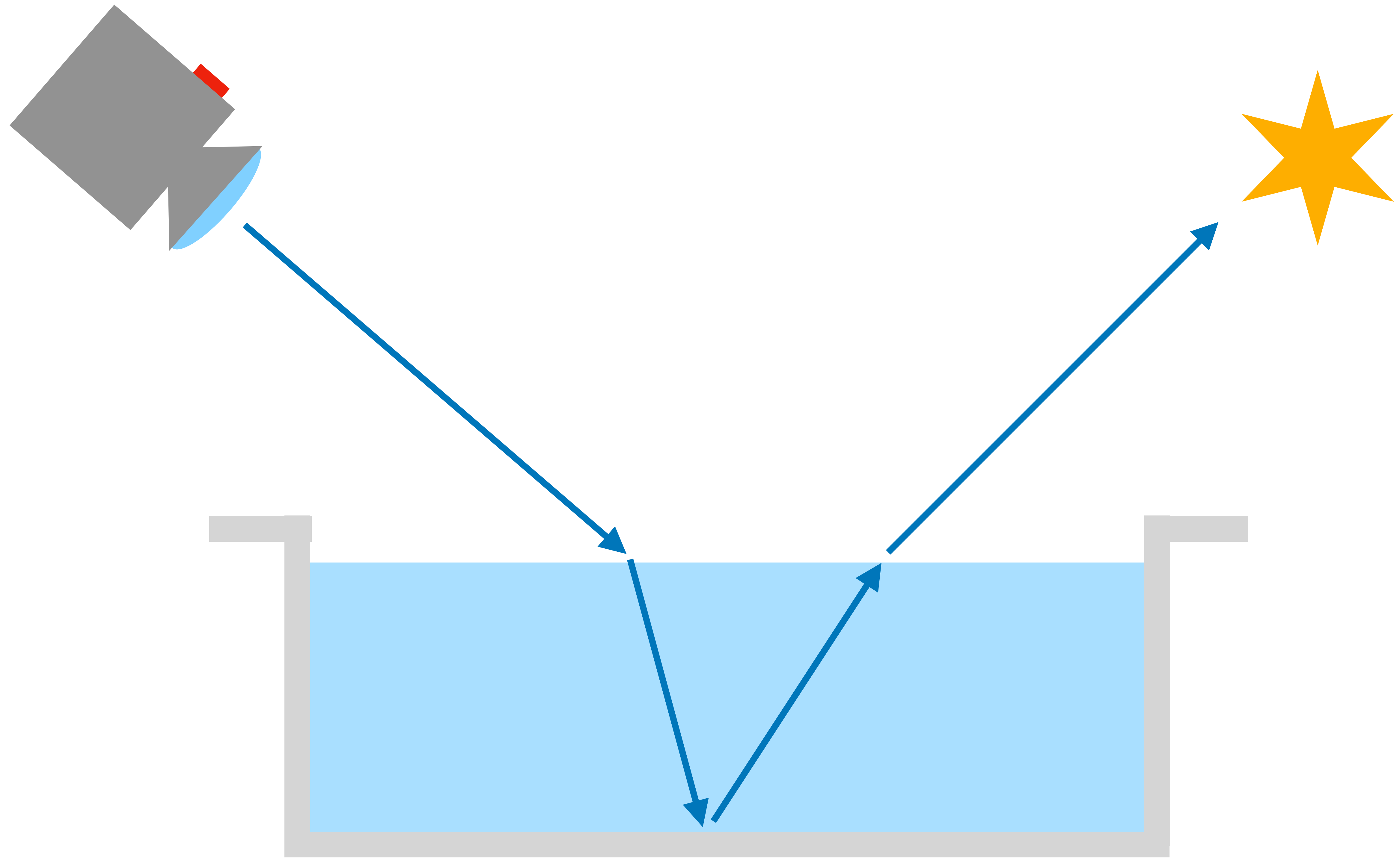
Image by Eric Veach

Path-Tracing
Metropolis Sampling

Image by Eric Veach

Path-Tracing

Metropolis Sampling: Multi-Chains



Path-Tracing

Metropolis Sampling: Multi-Chains

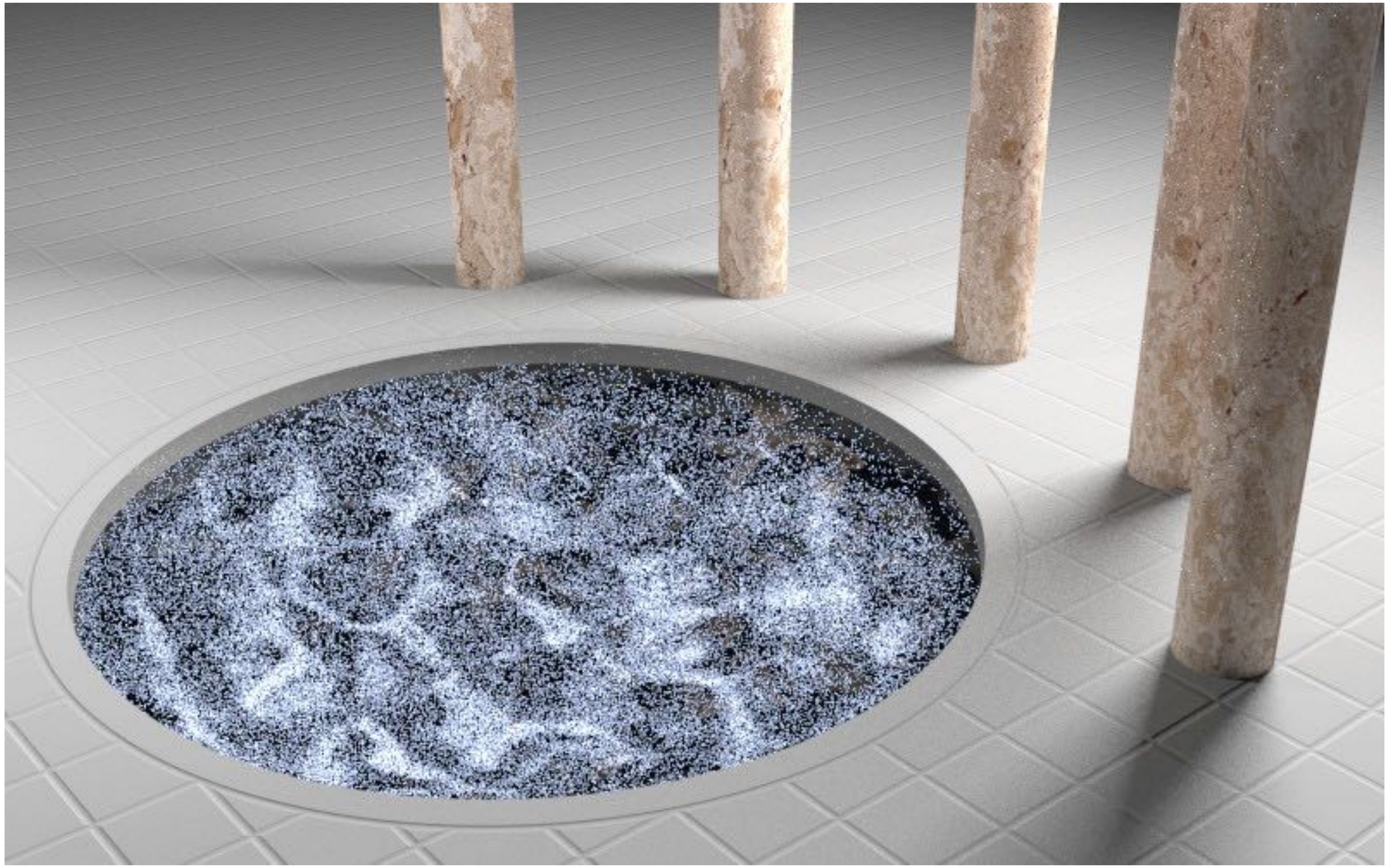
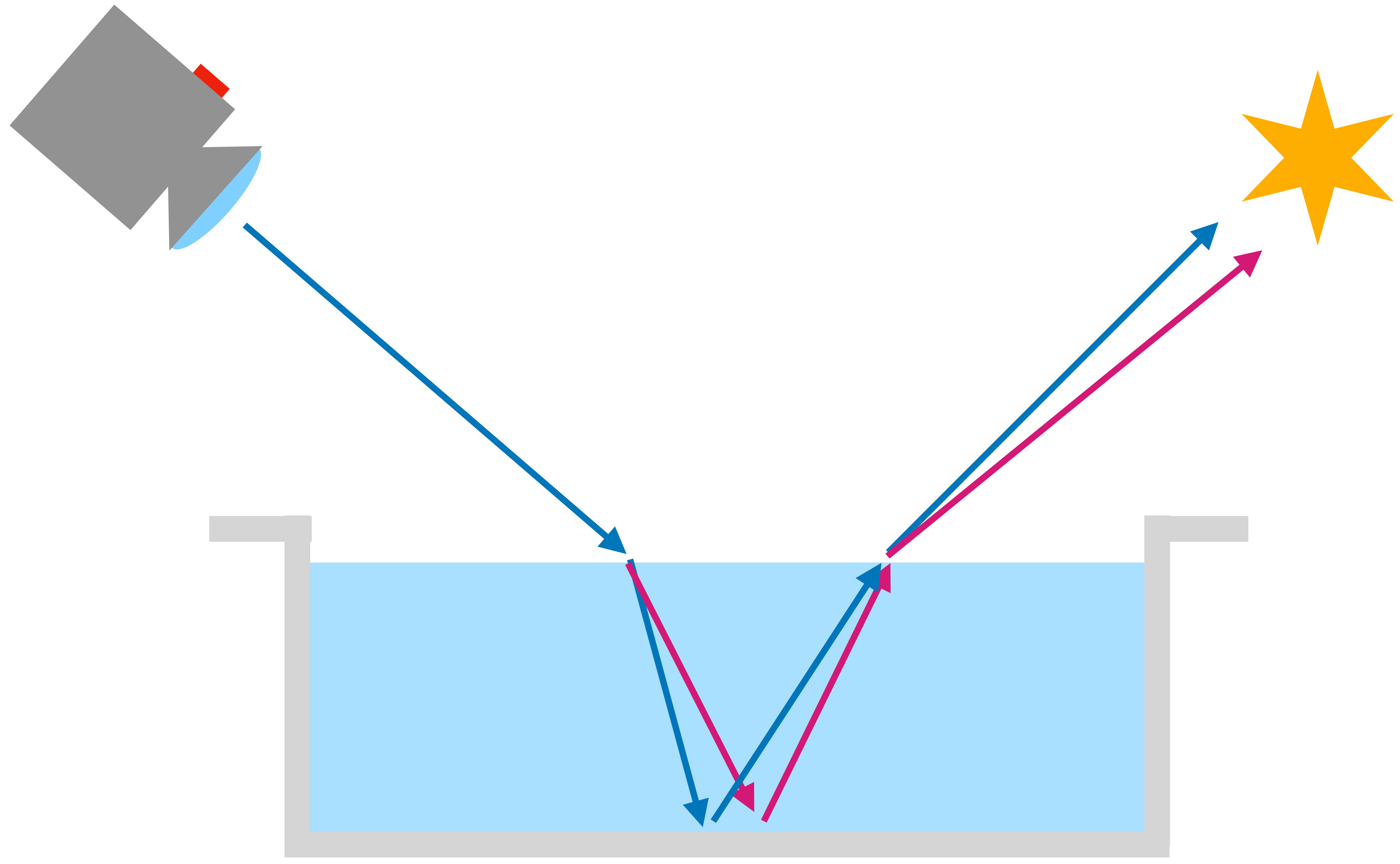


Image by Eric Veach

Path-Tracing

Metropolis Sampling: Multi-Chains



Path-Tracing

Metropolis Sampling: Multi-Chains

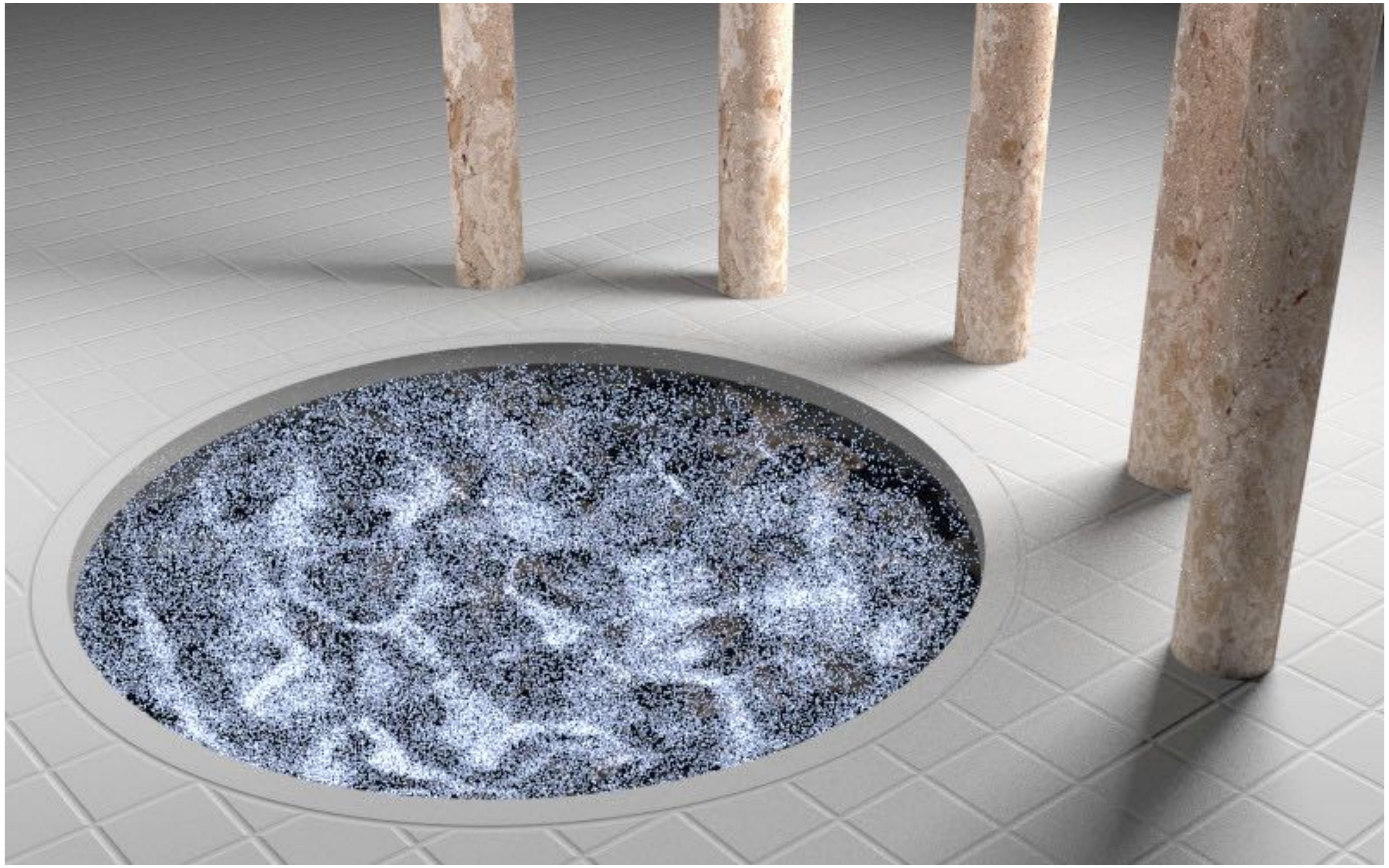


Image by Eric Veach

Path-Tracing

Metropolis Sampling: Multi-Chains

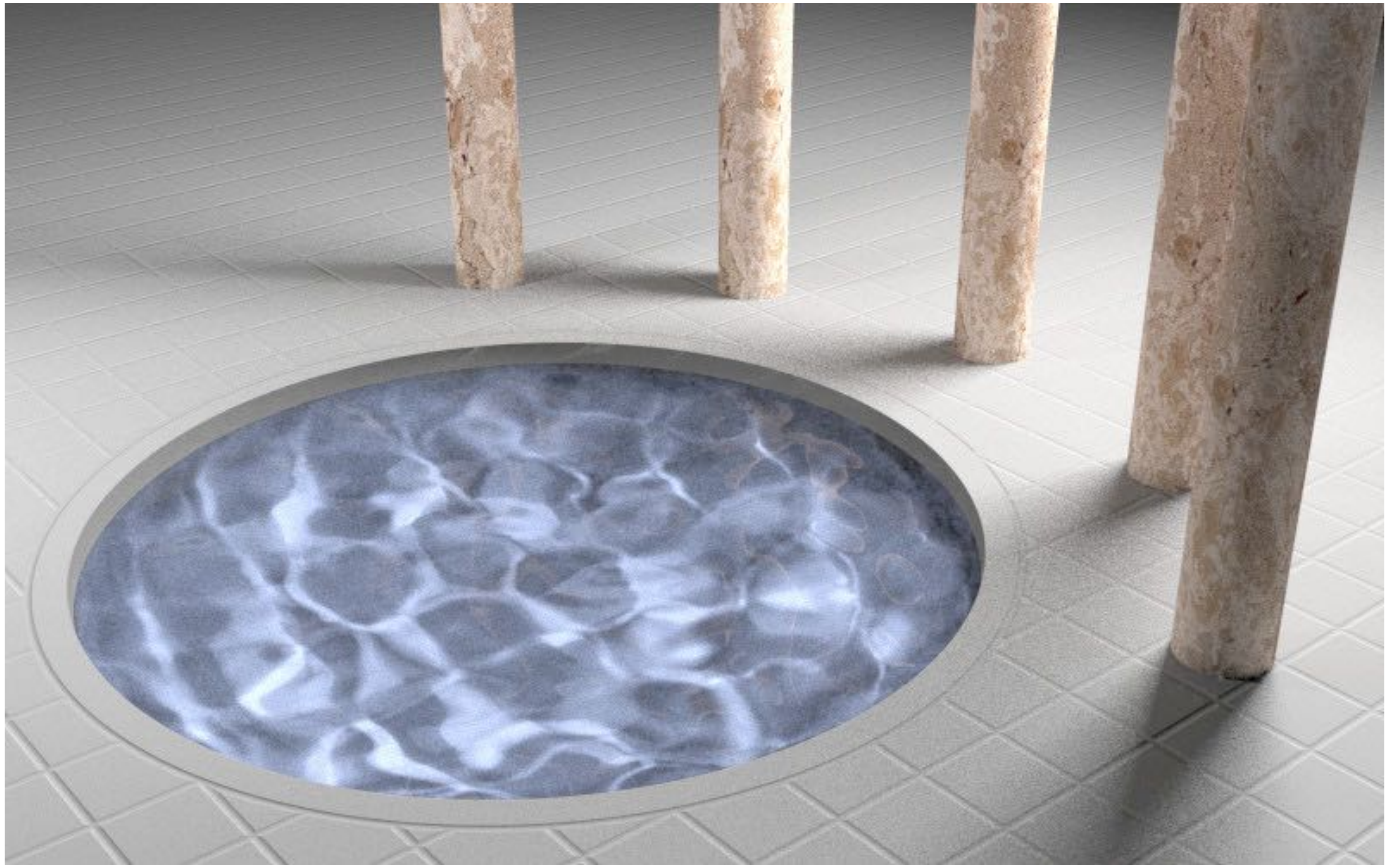


Image by Eric Veach

Bibliography

- Peter Shirley. “Ray Tracing: The Rest of Your Life”. 2018-2020. raytracing.github.io
- Matt Pharr, Wenzel Jakob, and Greg Humphreys. Chapter 13: “Monte Carlo Integration” from the book “Physically Based Rendering: From Theory To Implementation”. Morgan Kaufmann. 2016.
- Eric Veach and Leonidas Guibas. “Metropolis Light Transport”. ACM SIGGRAPH 1997.
- Francesco Banterle, Massimiliano Corsini, Paolo Cignoni, Roberto Scopigno. “ A Low-Memory, Straightforward and Fast Bilateral Filter Through Subsampling in Spatial Domain”. In Computer Graphics Forum 31(1). 2012.

Thank you for your attention!