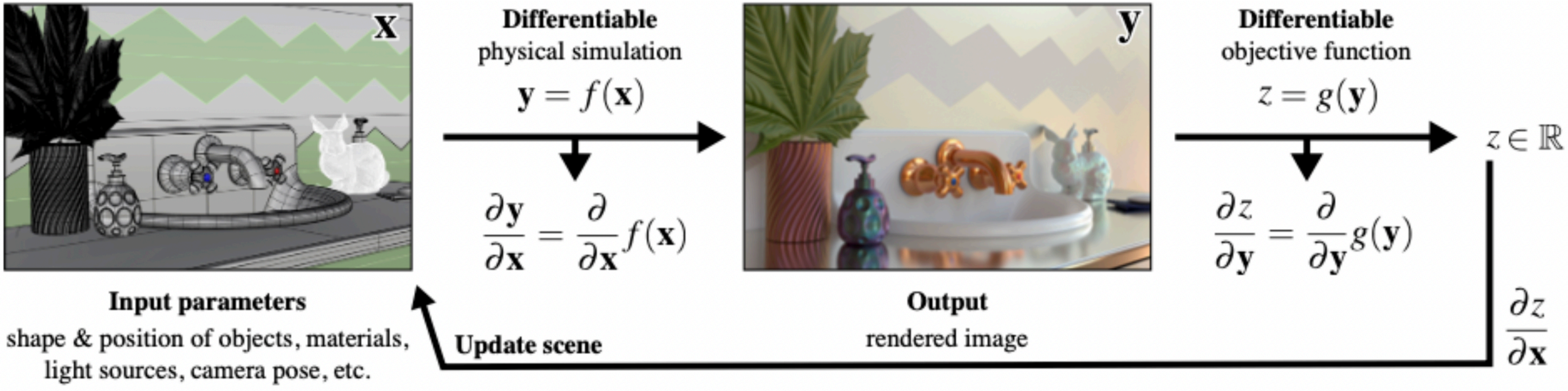
Differentiable Rendering



Gurprit Singh

UNIVERSITÄT DES SAARLANDES

SIGGRAPH THINK BEYOND

2020 S2020.SIGGRAPH.ORG

PHYSICS-BASED DIFFERENTIABLE RENDERING

A COMPREHENSIVE INTRODUCTION

Shuang Zhao — University of California, Irvine

Wenzel Jakob — EPFL, Lausanne, Switzerland

Tzu-Mao Li — MIT CSAIL, Cambridge

FORWARD VS. INVERSE RENDERING

Geometry, materials, emitters, ...

Scene: "bed classic" from jiraniano

FORWARD VS. INVERSE RENDERING

Rendering

Geometry, materials, emitters, ...

Scene: "bed classic" from jiraniano

FORWARD VS. INVERSE RENDERING

Rendering

$$\mathbf{y} = f(\mathbf{x})$$

Geometry, materials, emitters, ...

Scene: "bed classic" from jiraniano

FORWARD VS. INVERSE RENDERING

Rendering

$$\mathbf{y} = f(\mathbf{x})$$

Inverse rendering

$$\mathbf{x} = f^{-1}(\mathbf{y})?$$

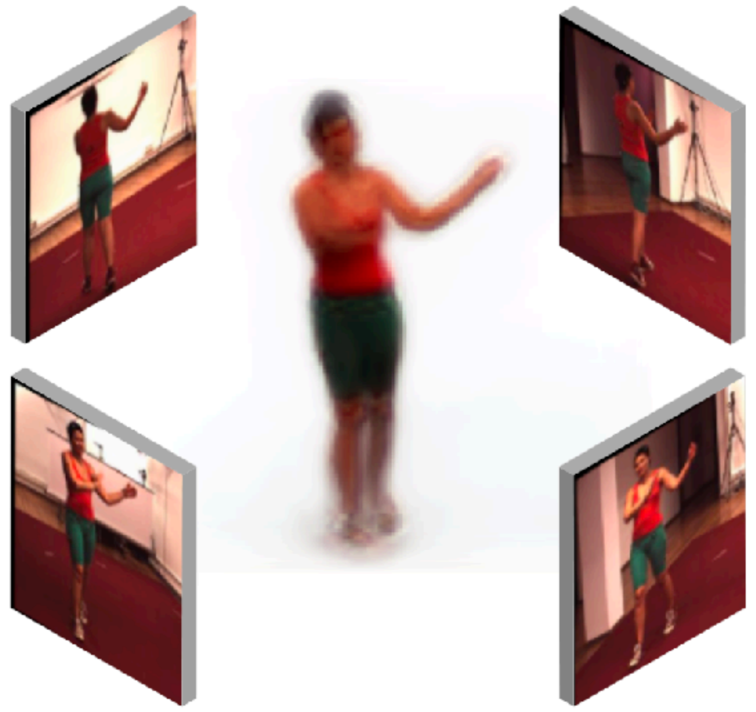
Geometry, materials, emitters, ...

Scene: "bed classic" from jiraniano

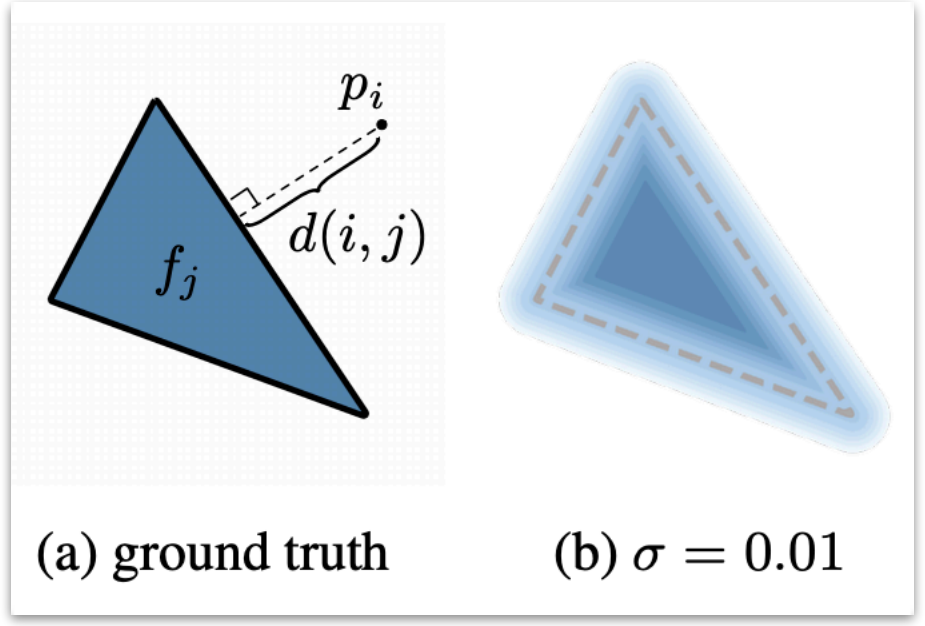
INVERSE RENDERING IN COMPUTER VISION

OpenDR: an Approximate Differentiable Renderer
[Loper et al. 2014]

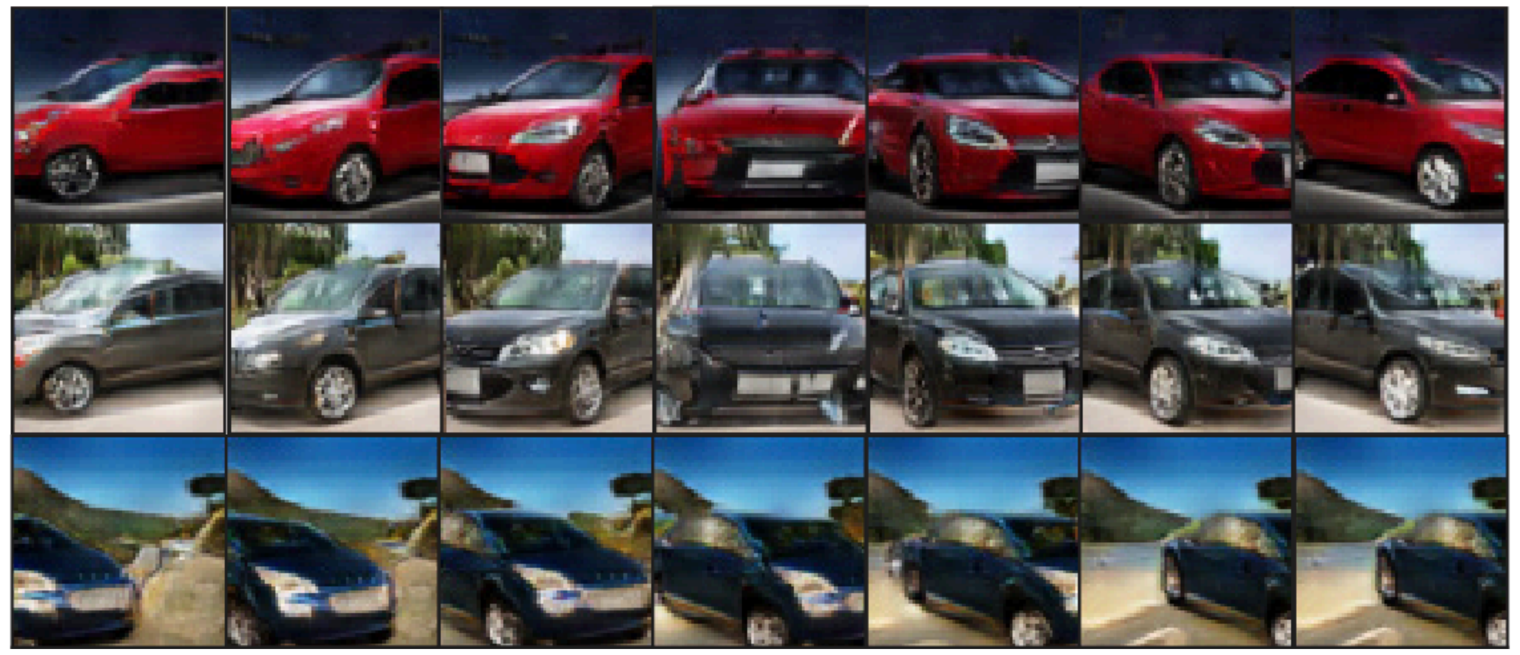
Neural 3D Mesh Renderer
[Kato et al. 2017]



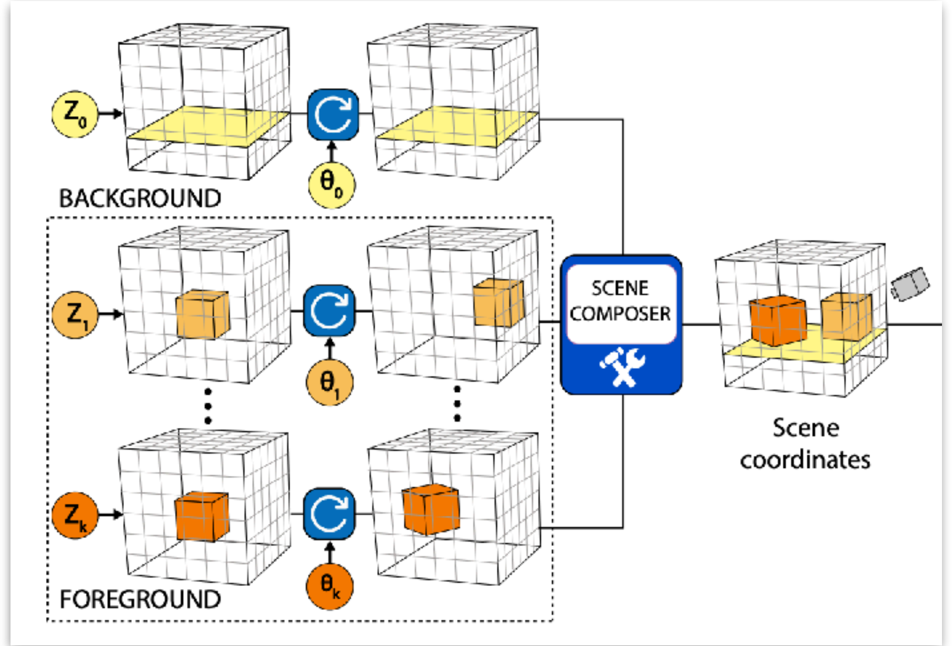
Unsupervised Geometry-Aware Representation for 3D Human Pose Estimation
[Rhodin et al., 2016]



Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction
[Liu et al. 2019]



HoloGAN: Unsupervised Learning of 3D Representations From Natural Images.
[Nguyen-Phuoc et al. 2019]



BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images
[Nguyen-Phuoc et al. 2020]

PHYSICS-BASED INVERSE RENDERING

- Focus on inverse rendering for realistic functions $f(\mathbf{x})$

Global illumination, complex materials, participating media, polarization, color spectra, etc.

PHYSICS-BASED INVERSE RENDERING

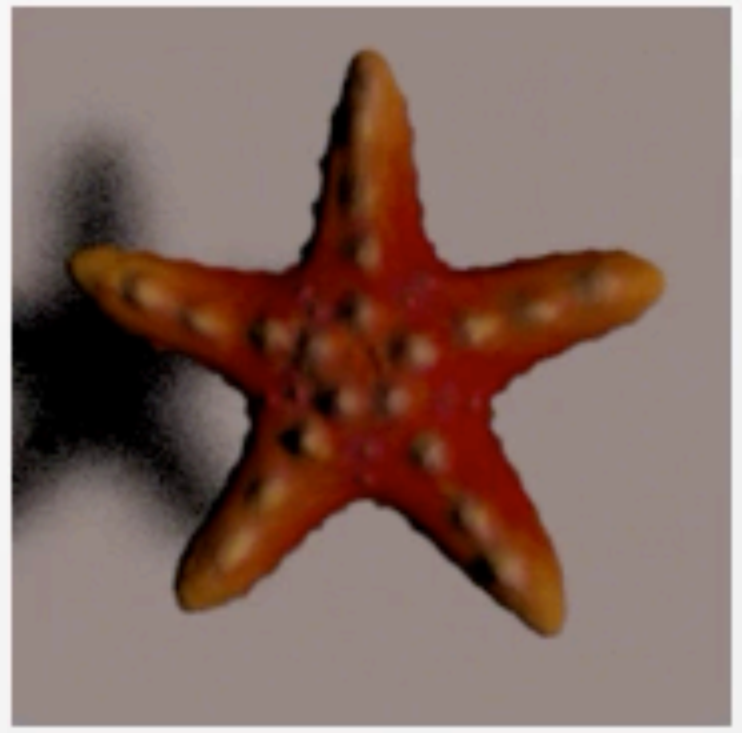
- **Focus on inverse rendering for realistic functions** $f(\mathbf{x})$

  Global illumination, complex materials, participating media, polarization, color spectra, etc.

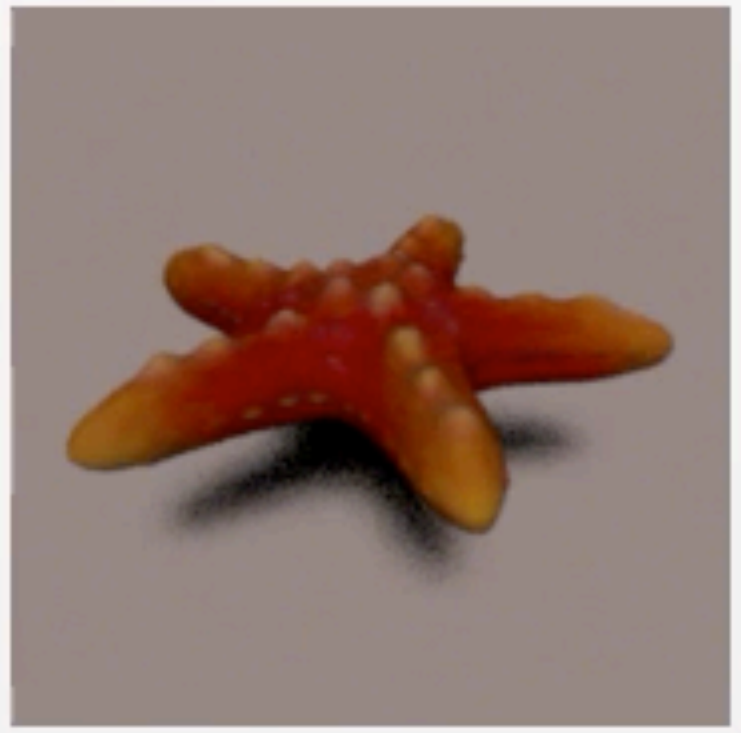
- **No neural networks.**

  Shouldn't need them, we understand the underlying equations.
  *(Of course still possible to use neural nets **inside or outside** of the renderer)*

SHAPE & MATERIAL RECONSTRUCTION



Target



Target

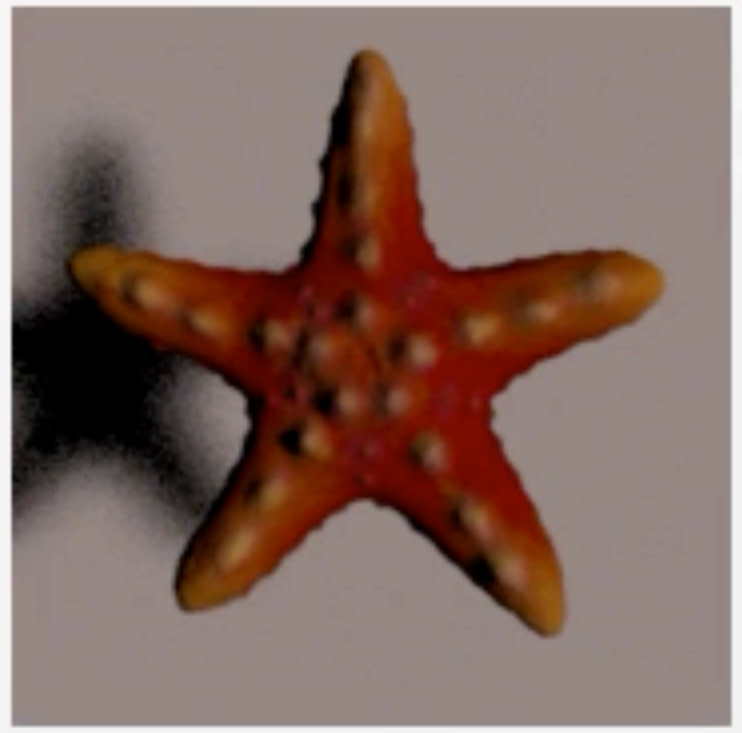
Target

Target

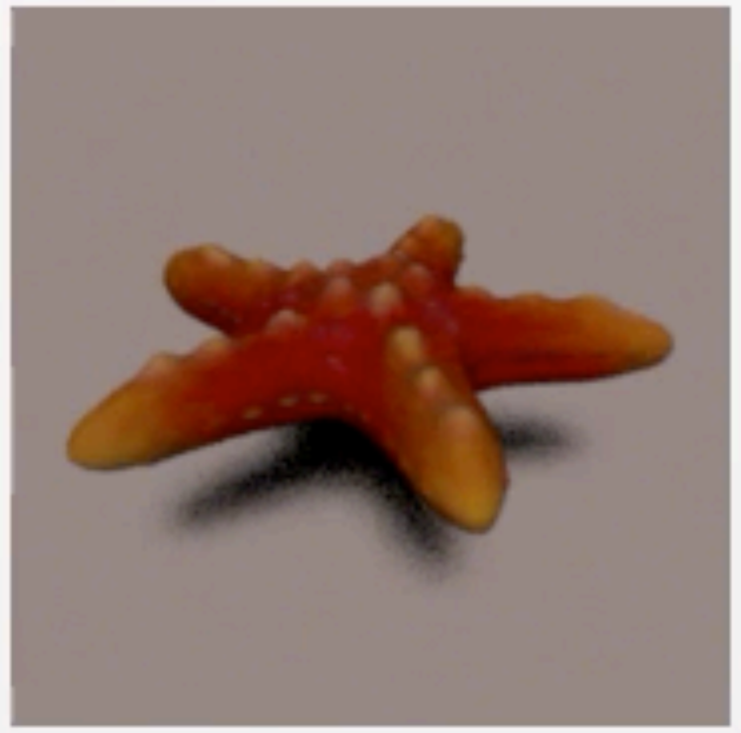
Reparameterizing discontinuous integrands for differentiable rendering [Loubet et al. 2019]

SHAPE & MATERIAL RECONSTRUCTION


Input scene


Target


Input scene


Target


Input scene

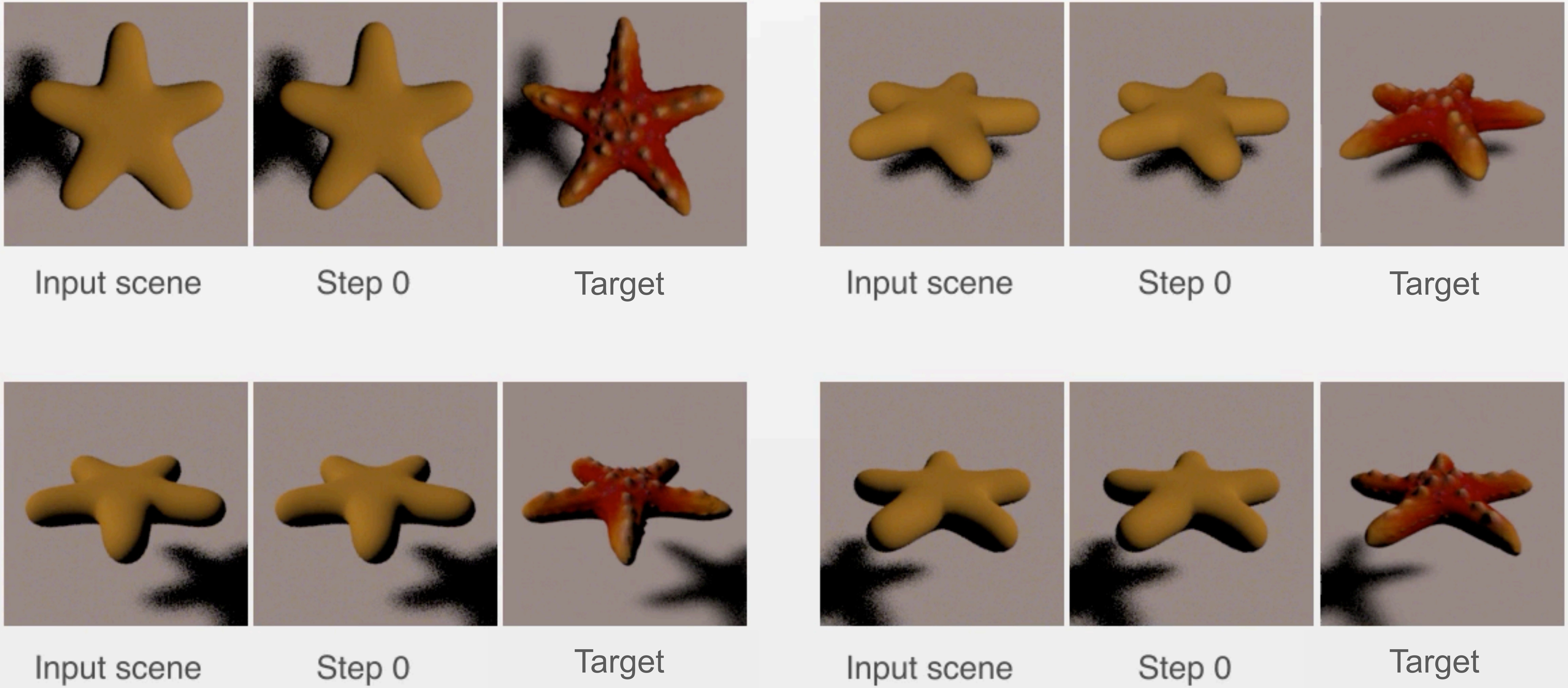

Target


Input scene


Target

Reparameterizing discontinuous integrands for differentiable rendering [Loubet et al. 2019]

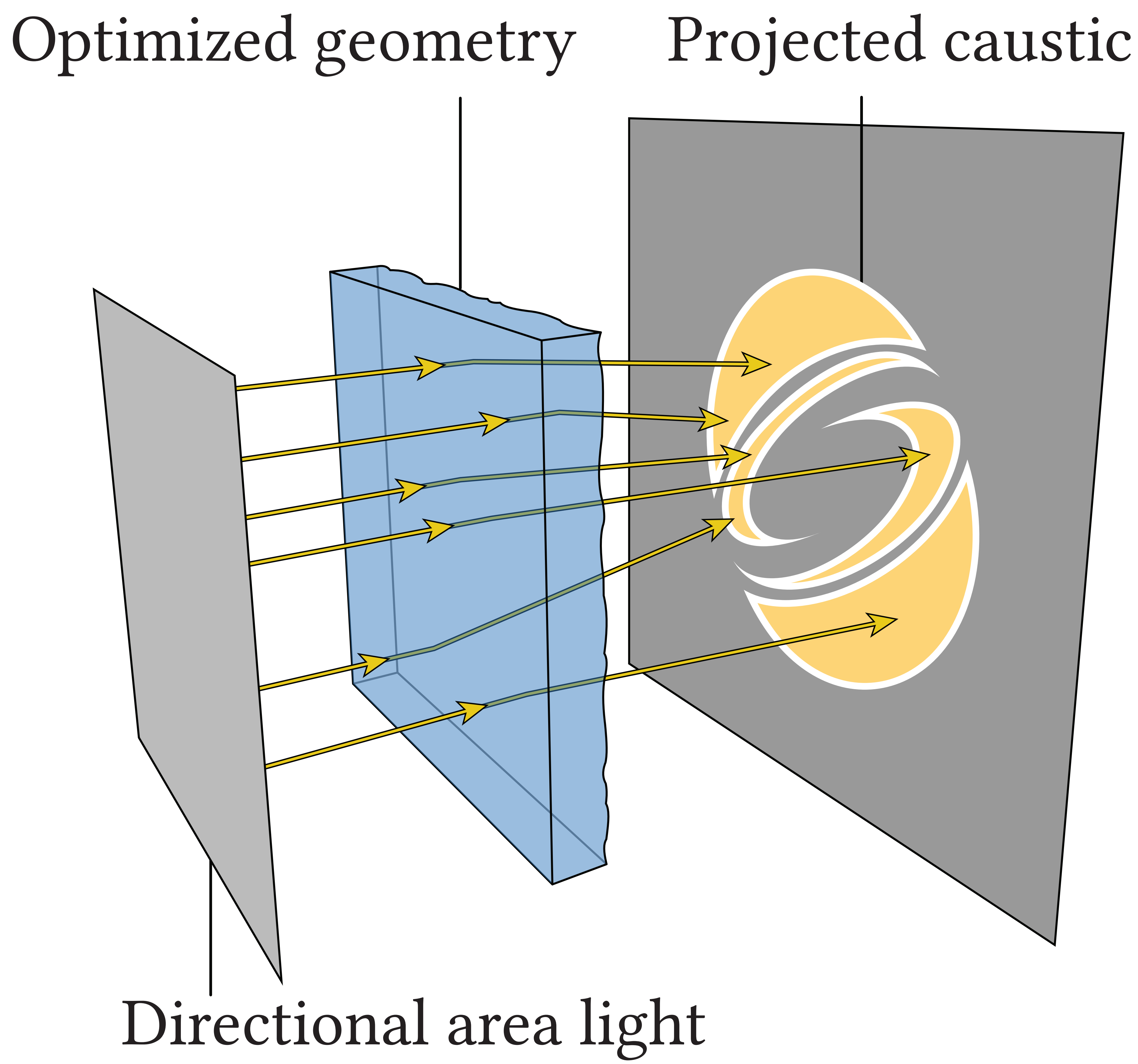
SHAPE & MATERIAL RECONSTRUCTION



Reparameterizing discontinuous integrands for differentiable rendering [Loubet et al. 2019]

CAUSTIC DESIGN

Schwartzburg et al. 2014



(META-) MATERIAL DESIGN

Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]

(META-) MATERIAL DESIGN

Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]

(META-) MATERIAL DESIGN

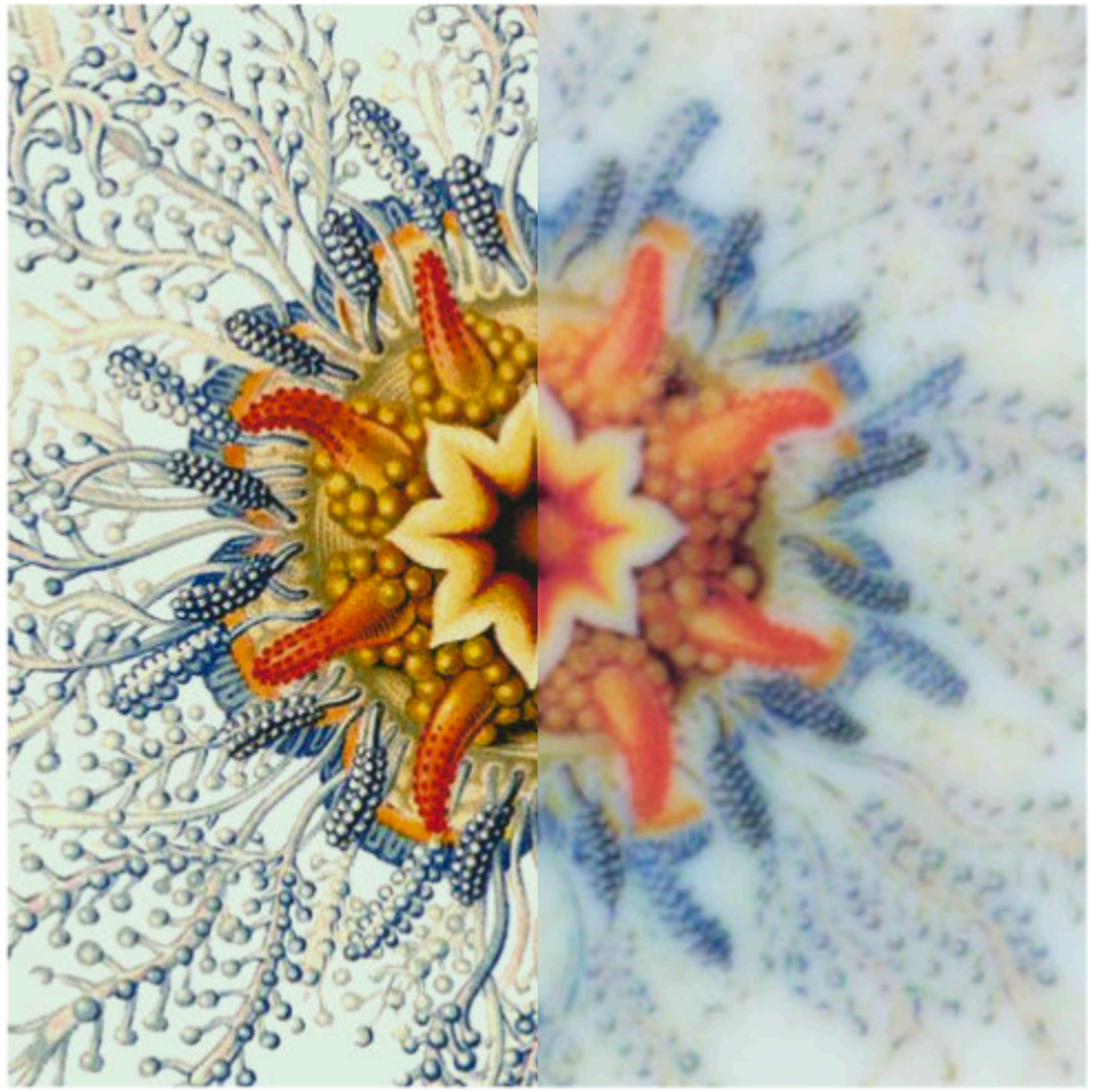
Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]

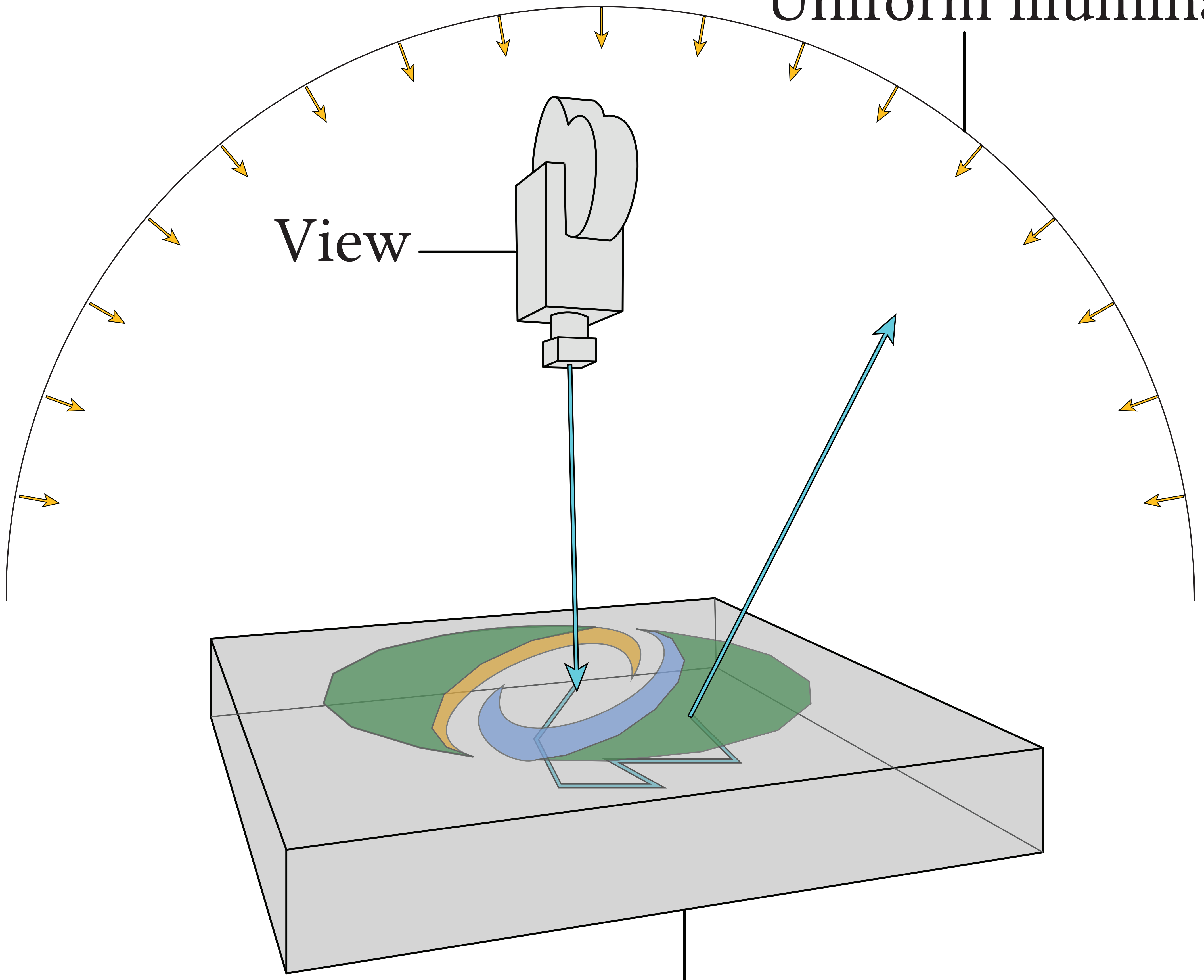
(META-) MATERIAL DESIGN

Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]

FABRICATION: 3D PRINT OPTIMIZATION

Elek et al. 2017





Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]

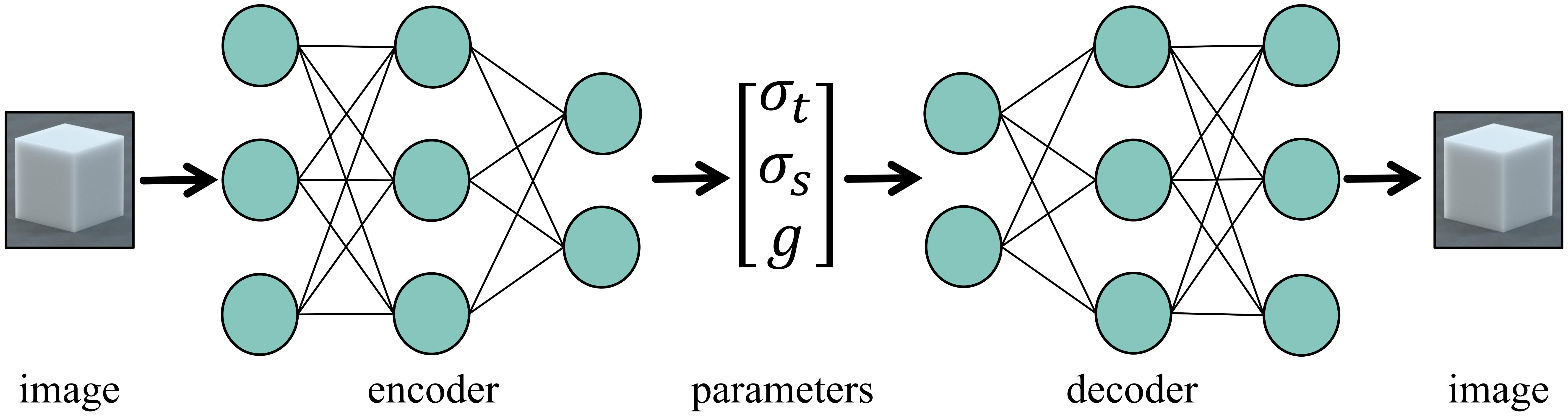
Reference: diffuse surface texture

Reference: diffuse surface texture

WHY DIFFERENTIABLE RENDERING

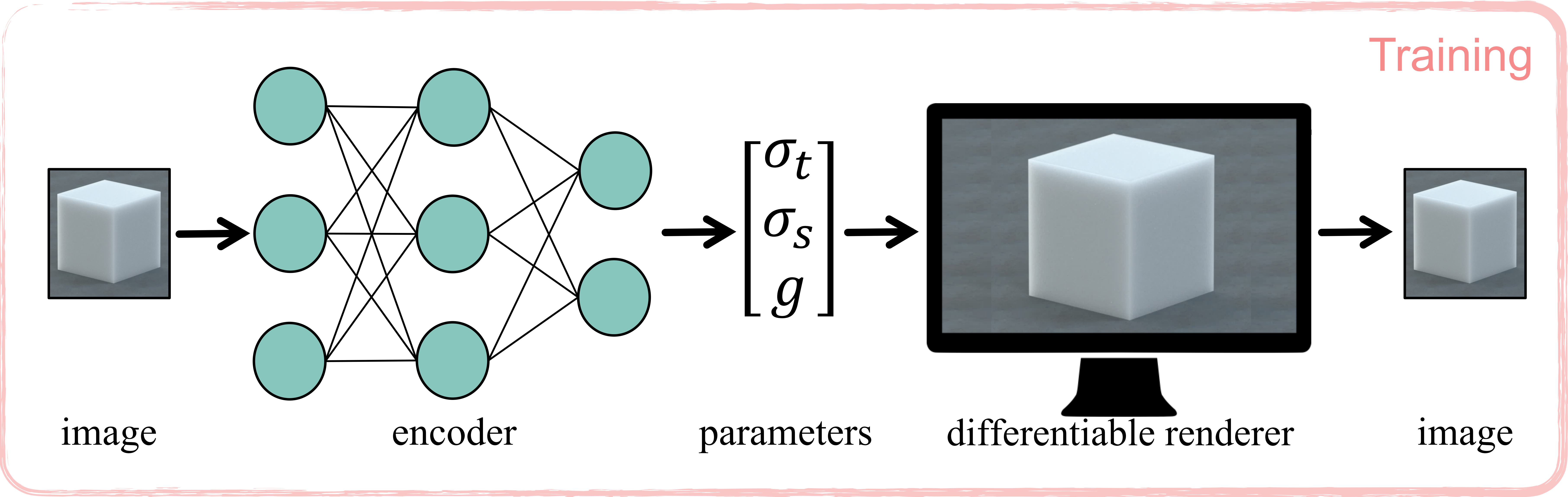
- Integrating physics-based rendering into **machine learning** & **probabilistic inference** pipelines

- Inverse subsurface scattering [Che et al. 2020]



WHY DIFFERENTIABLE RENDERING

- Integrating physics-based rendering into **machine learning** & **probabilistic inference** pipelines
- Inverse subsurface scattering [Che et al. 2020]

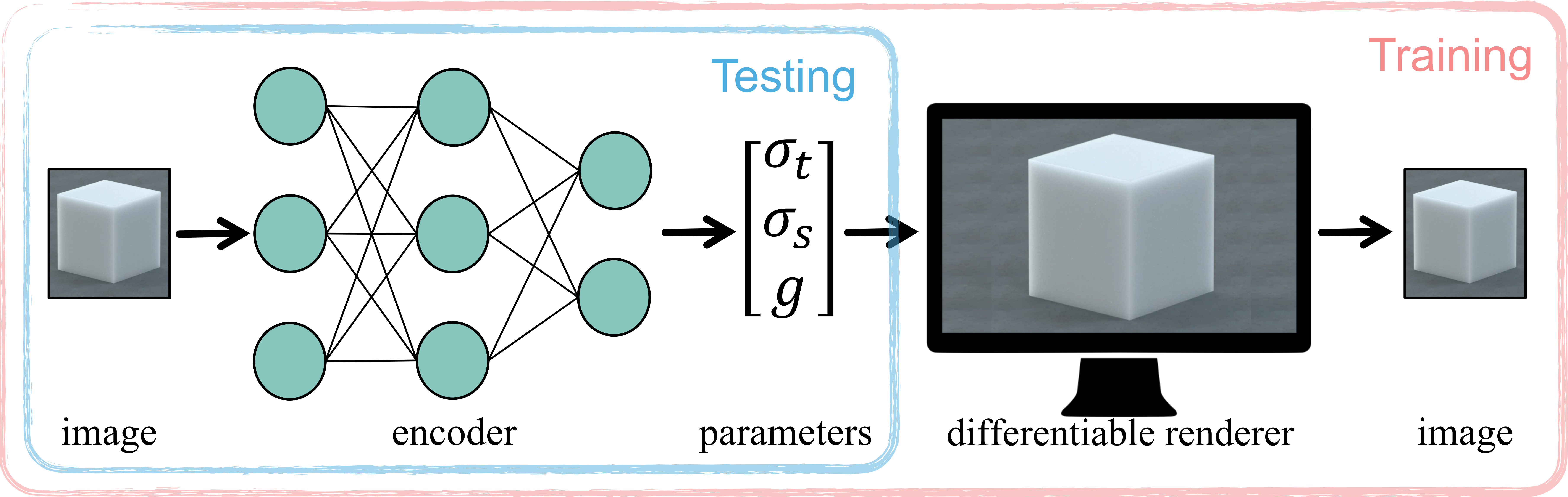


  – Utilizing *image loss* (provided by a volume path tracer) to regularize training

WHY DIFFERENTIABLE RENDERING

- Integrating physics-based rendering into **machine learning** & **probabilistic inference** pipelines

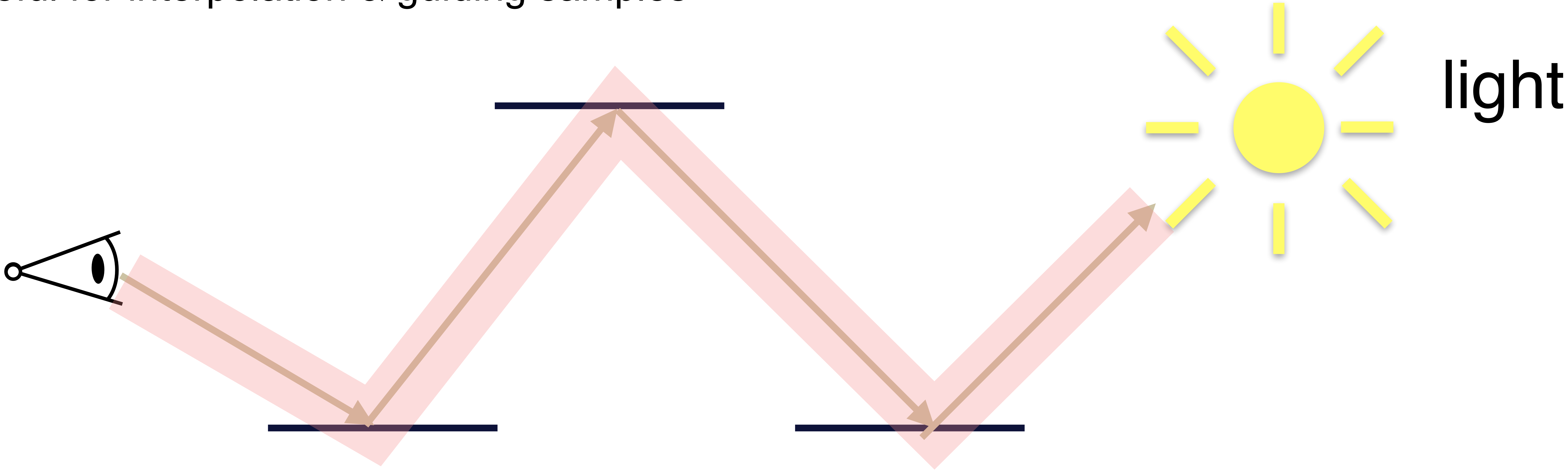
- Inverse subsurface scattering [Che et al. 2020]

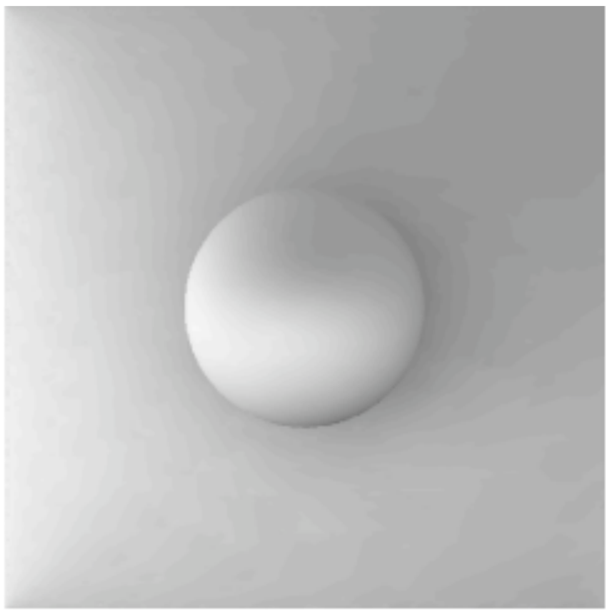


  - Utilizing *image loss* (provided by a volume path tracer) to regularize training
  - Use the trained encoder to solve inverse problems during testing

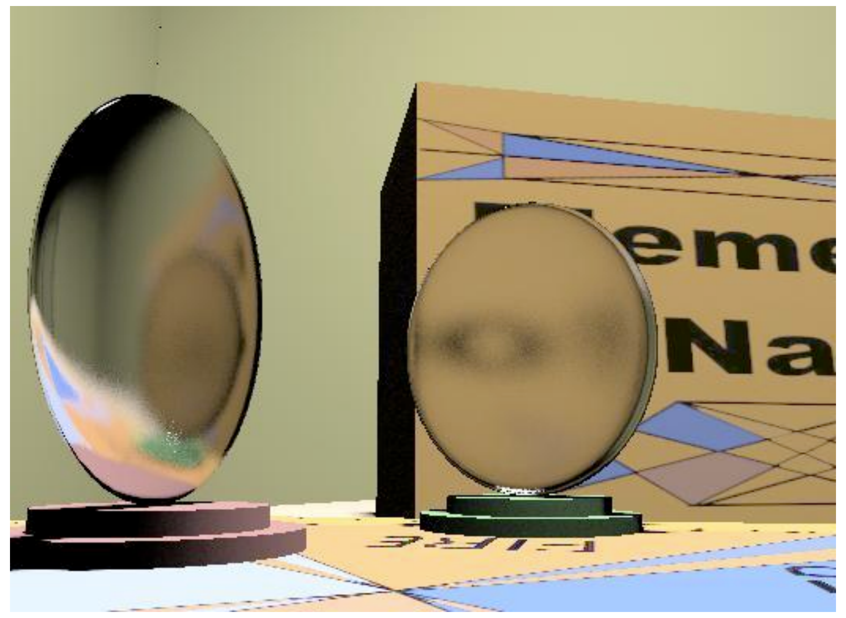
DIFFERENTIABLE RENDERING MAKES RENDERING FASTER

- Derivatives reveal neighborhood information of light paths
  - useful for interpolation & guiding samples

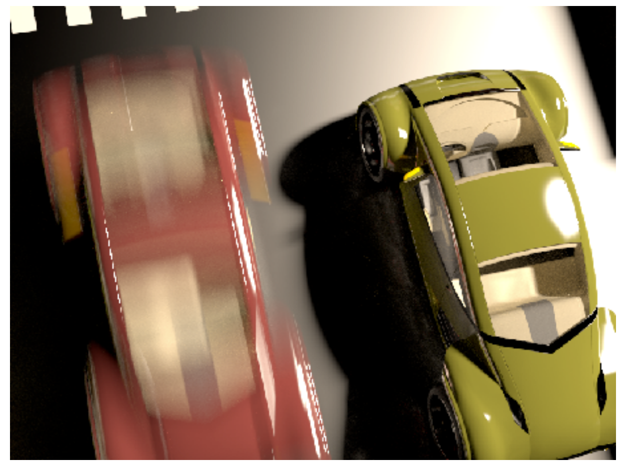




irradiance gradient
[Ward 1992]



path differentials
[Suykens and Williams 2001]



H2MC
[Li et al. 2015]

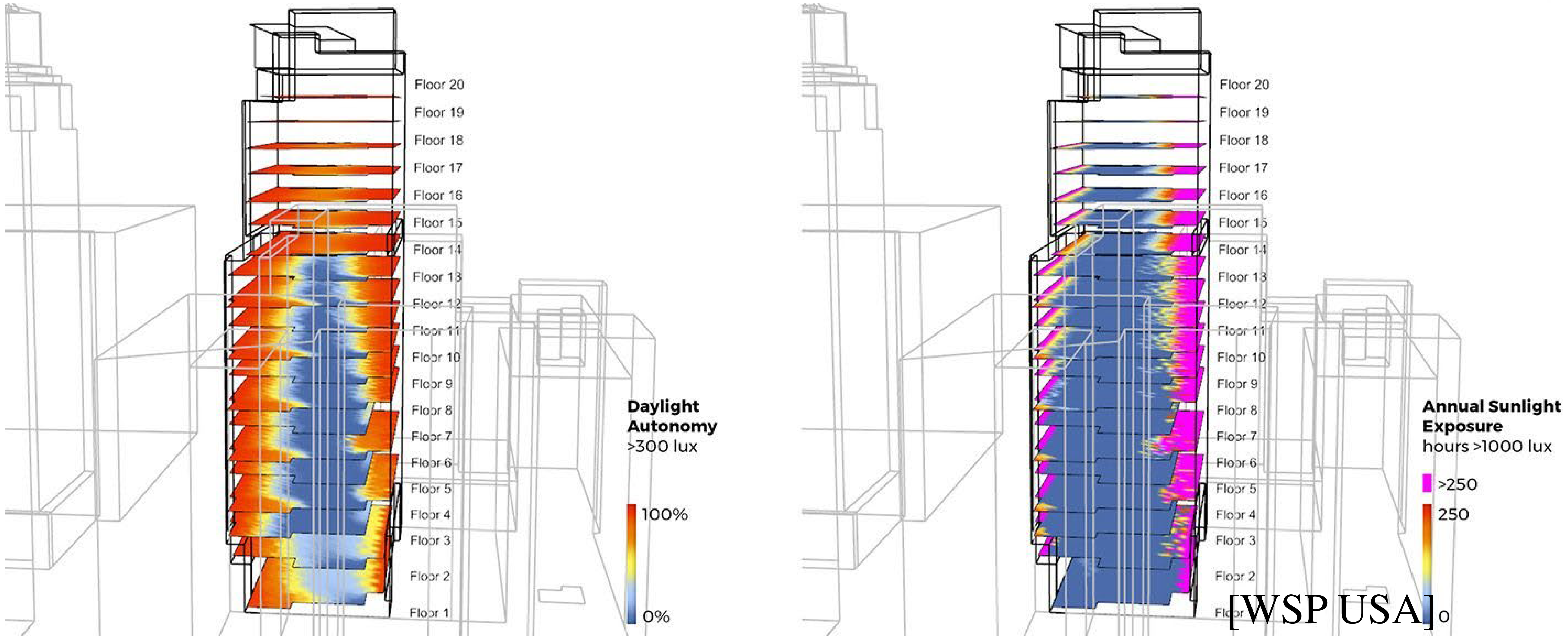
Langevin MC
[Luan et al. 2020]

BEYOND GRAPHICS: A WORLD OF APPLICATIONS

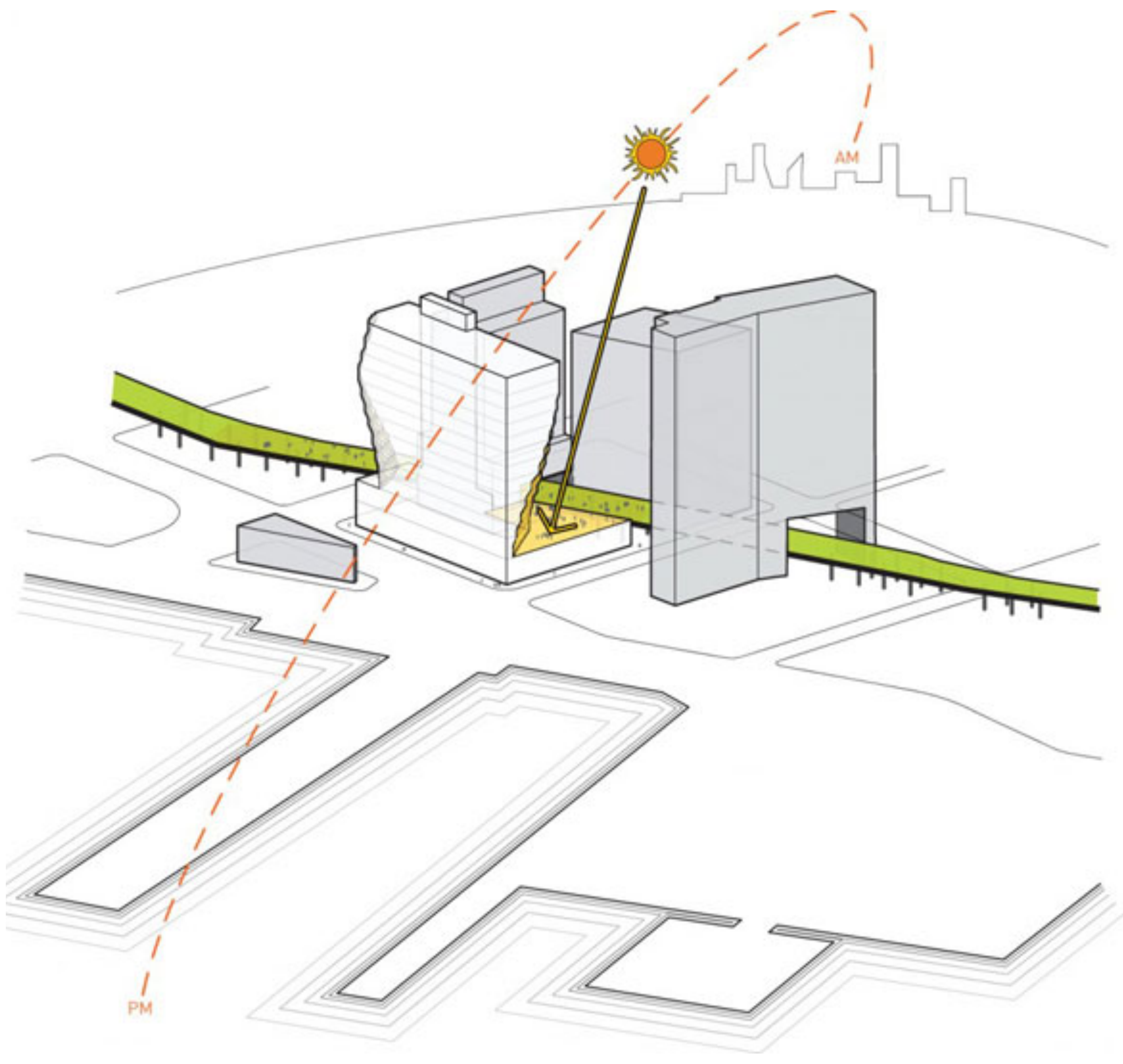
- Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.

BEYOND GRAPHICS: A WORLD OF APPLICATIONS

- Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.



[WSP USA]

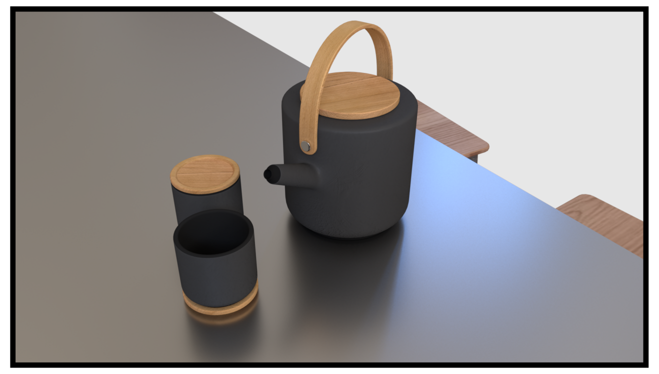
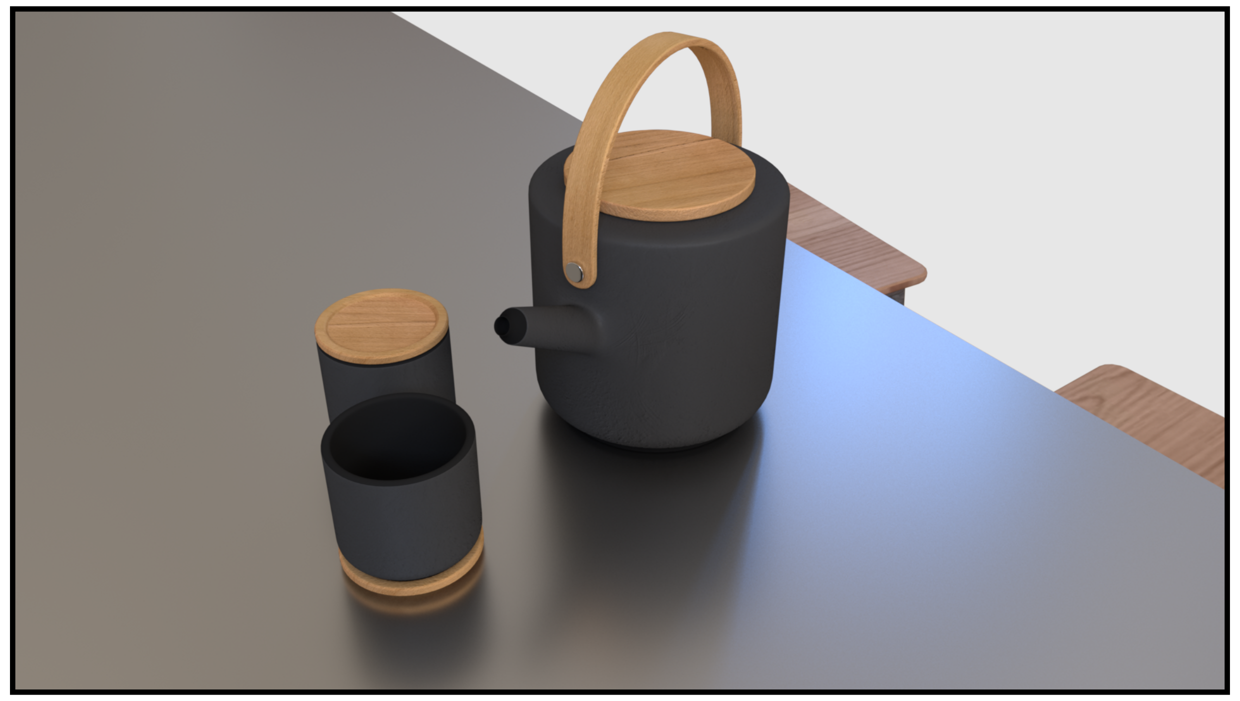
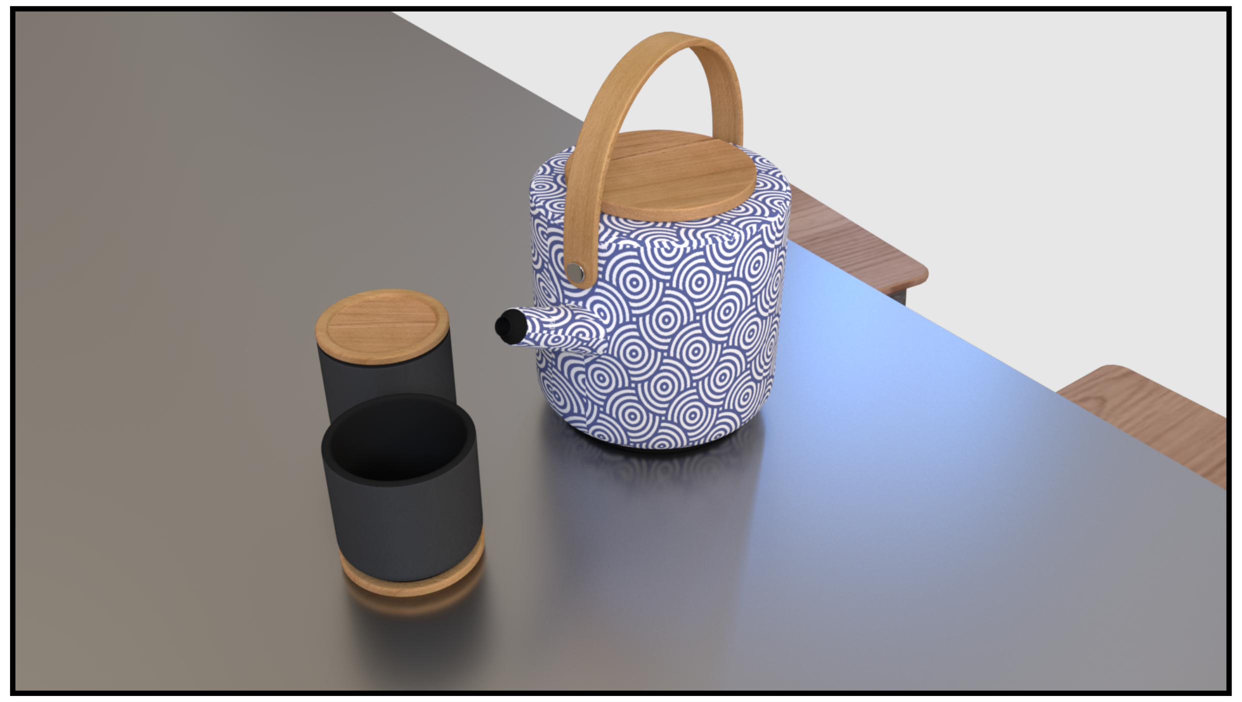


[Solar Carve Tower - Studio Gang]

Current rendering


Target

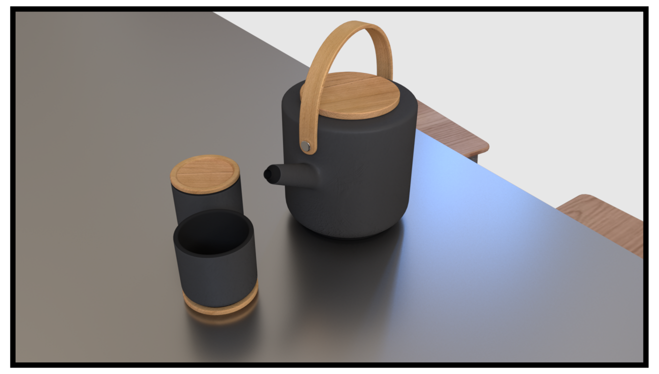
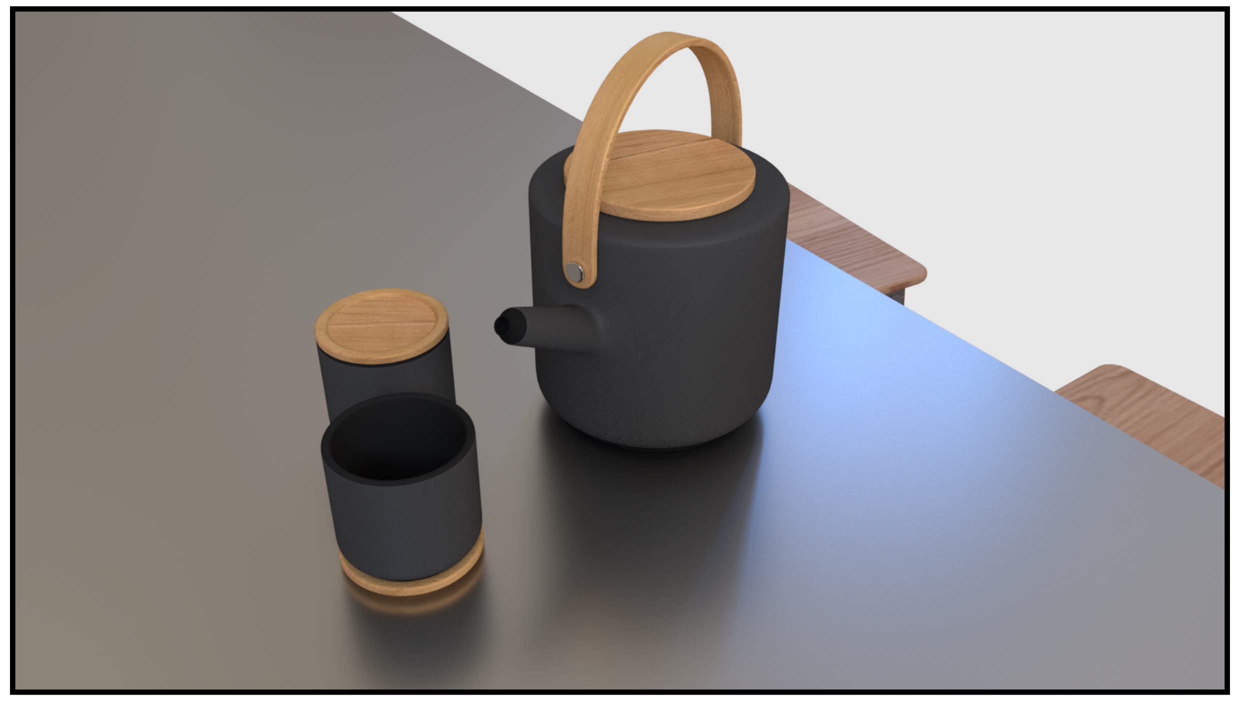
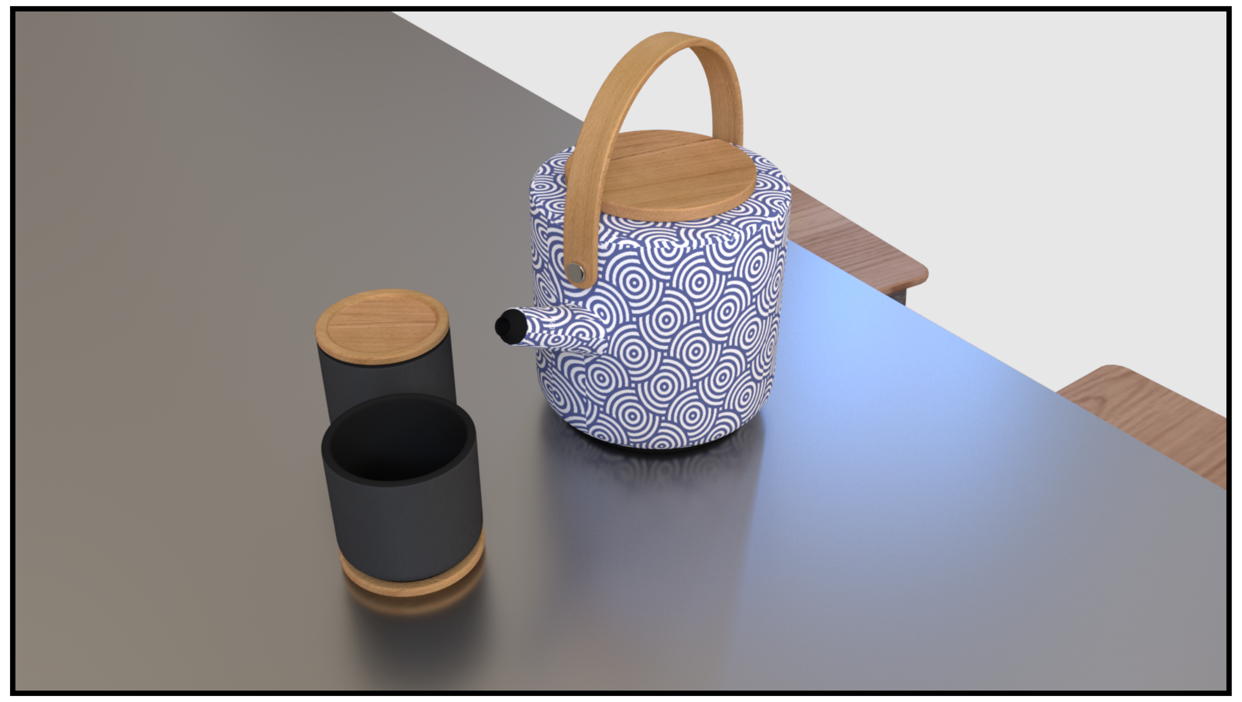
OBJECTIVE FUNCTION (A.K.A. "LOSS")

$g($  $) = \Big\|$  $-$  $\Big\|^2$

Rendering

Target

OBJECTIVE FUNCTION (A.K.A. "LOSS")

$g\Big($  $\Big) = \Big\|$  $-$  $\Big\|^2$

Rendering — Target

The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$

Scene parameters ($\mathbf{x}$)

Objective (g)

Rendering algorithm (f)

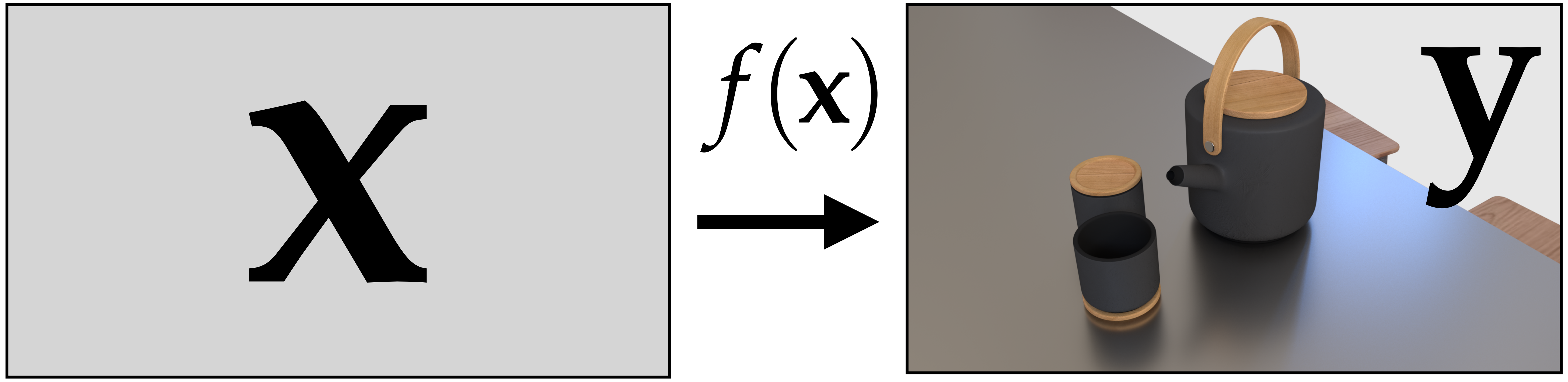
DIFFERENTIABLE RENDERING

The problem: $$\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$$

$\mathbf{x}$

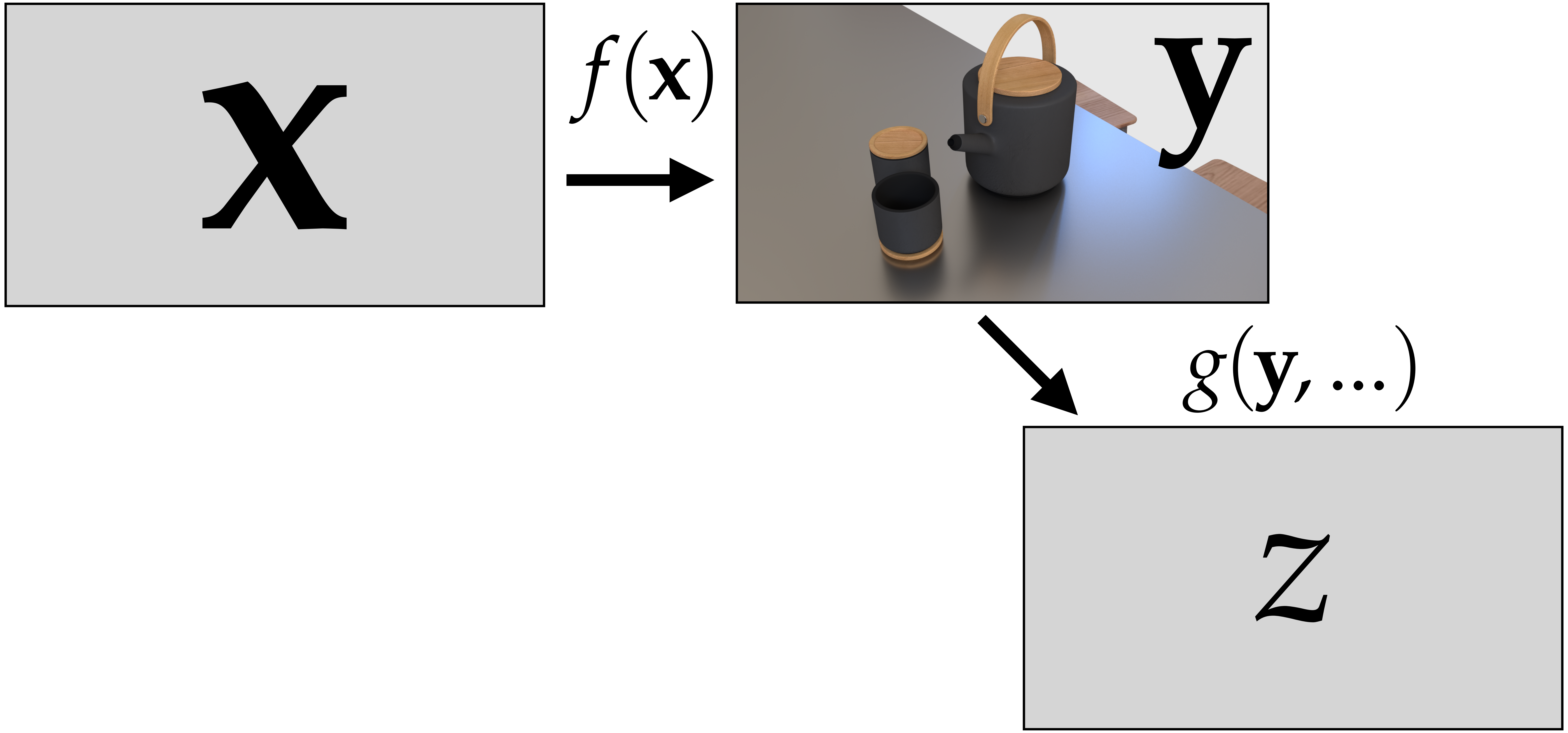
DIFFERENTIABLE RENDERING

The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$

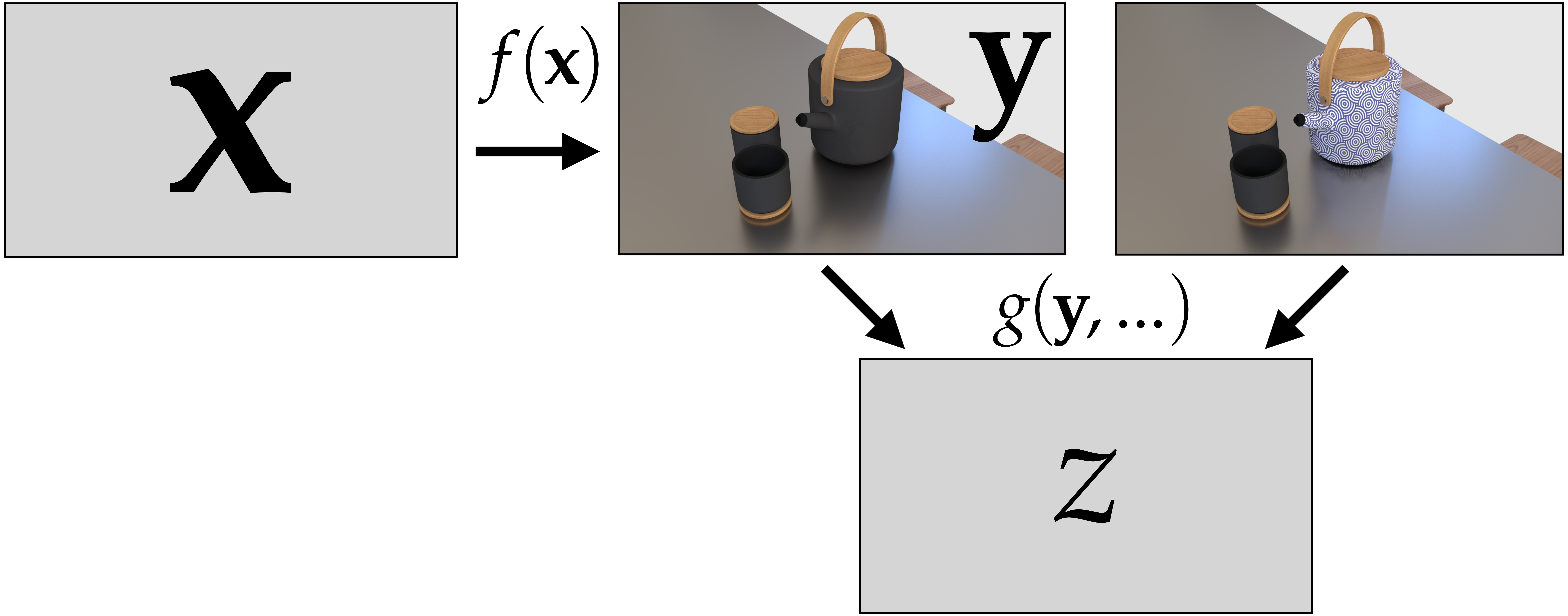


- meshes
- material (BSDF) parameters
  - textures, etc.
- parameters of procedural models
- volumes, light sources, …

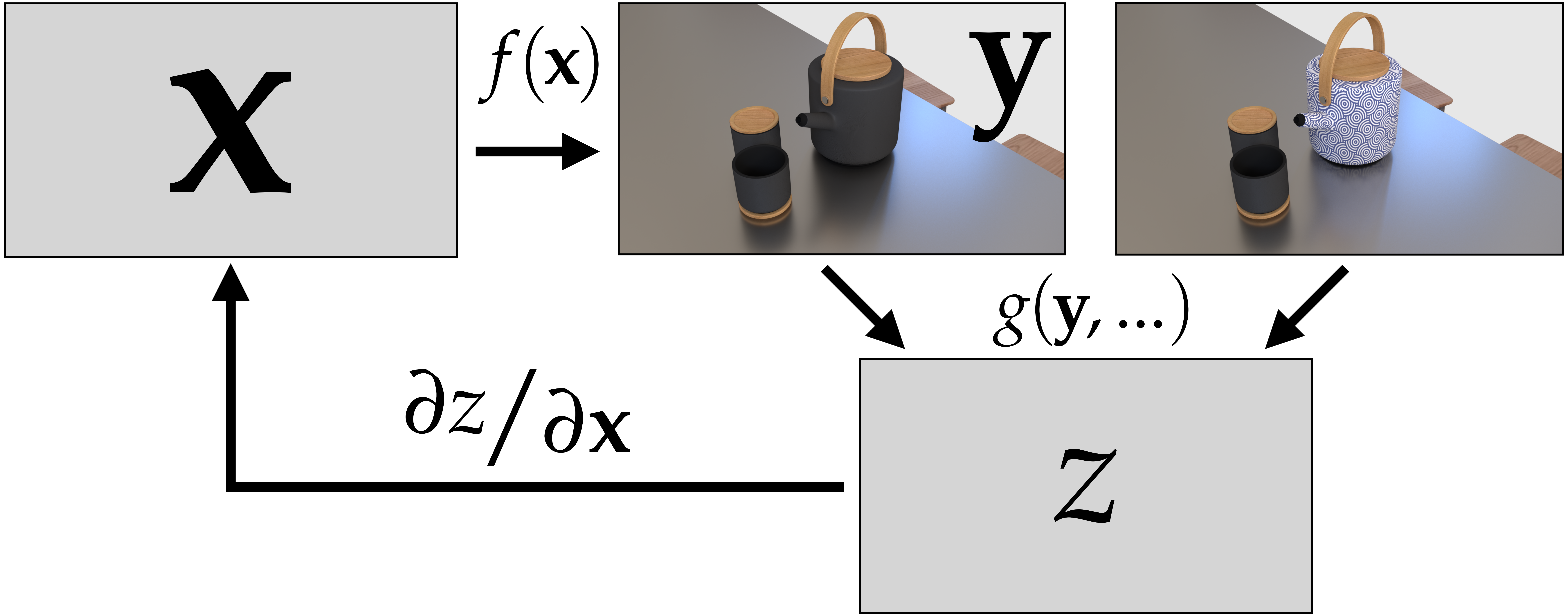
The problem: $$\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$$



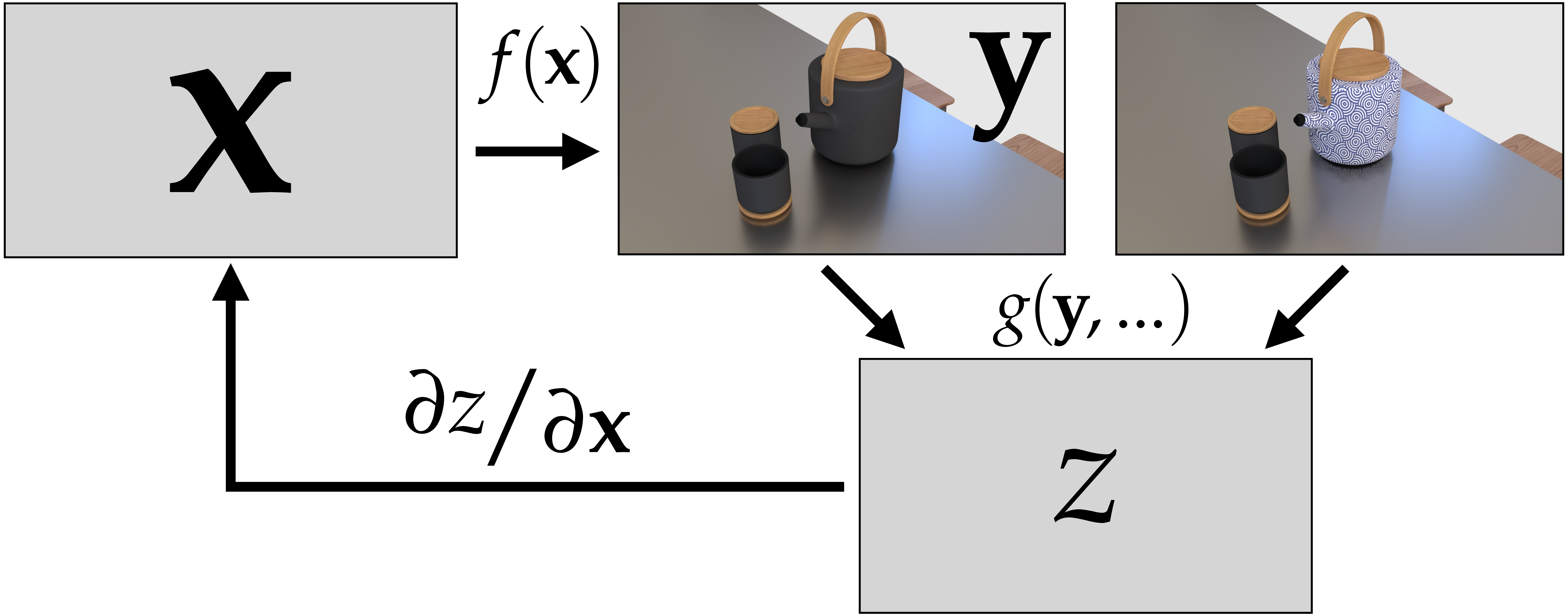
The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\ g(f(\mathbf{x}))$



The problem: $\underset{\mathbf{x}\in\mathcal{X}}{\text{minimize}}\; g(f(\mathbf{x}))$



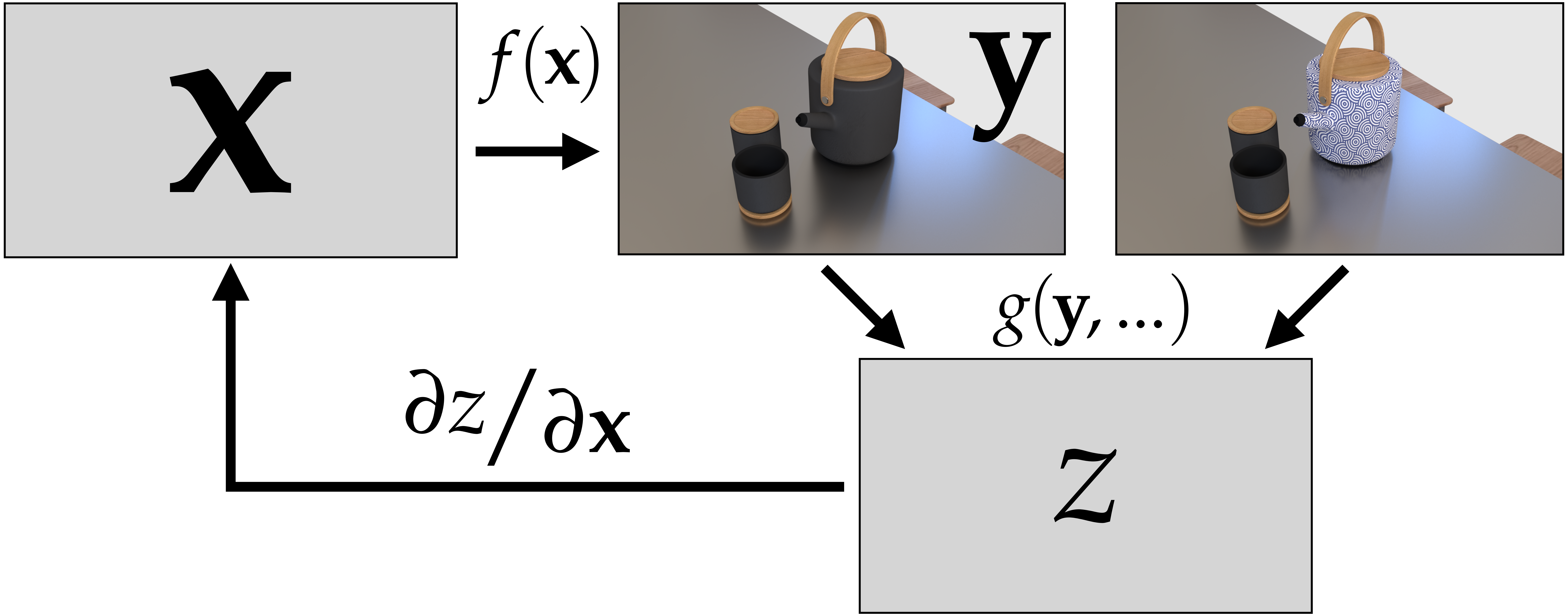
DIFFERENTIABLE RENDERING



DIFFERENTIABLE RENDERING

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

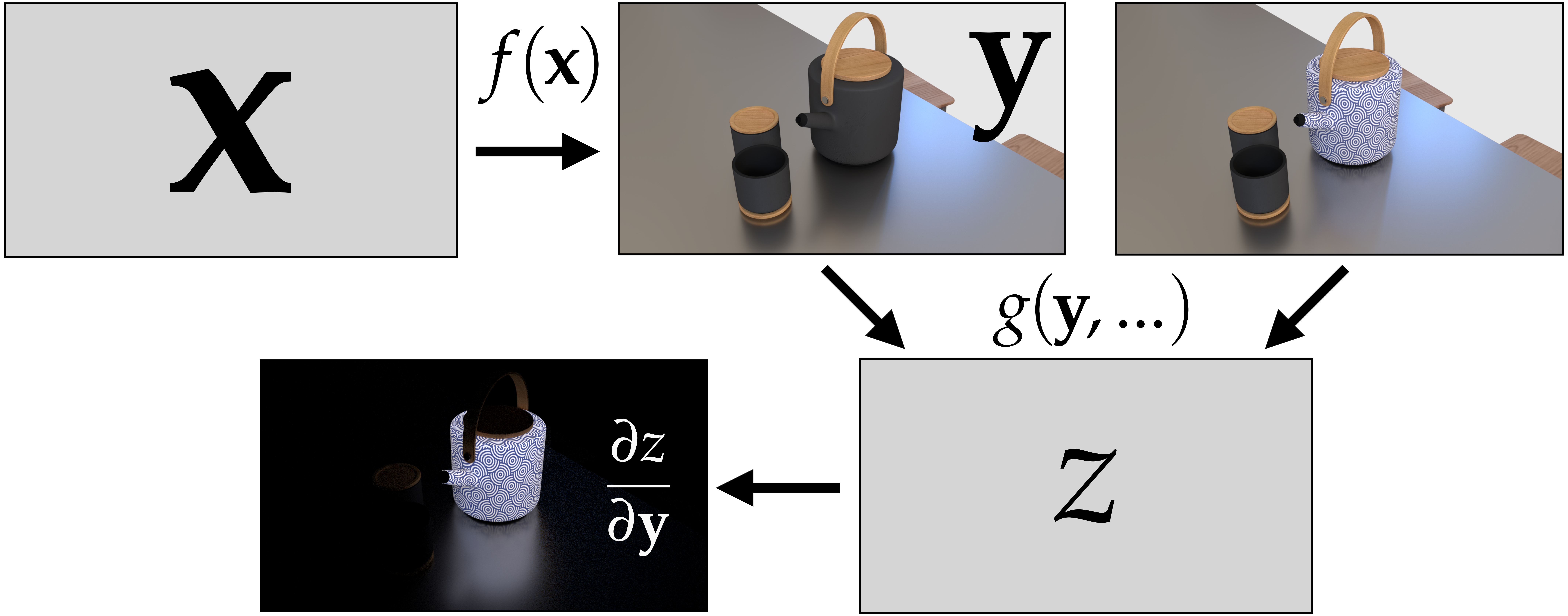
CHAIN RULE



DIFFERENTIABLE RENDERING

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

CHAIN RULE

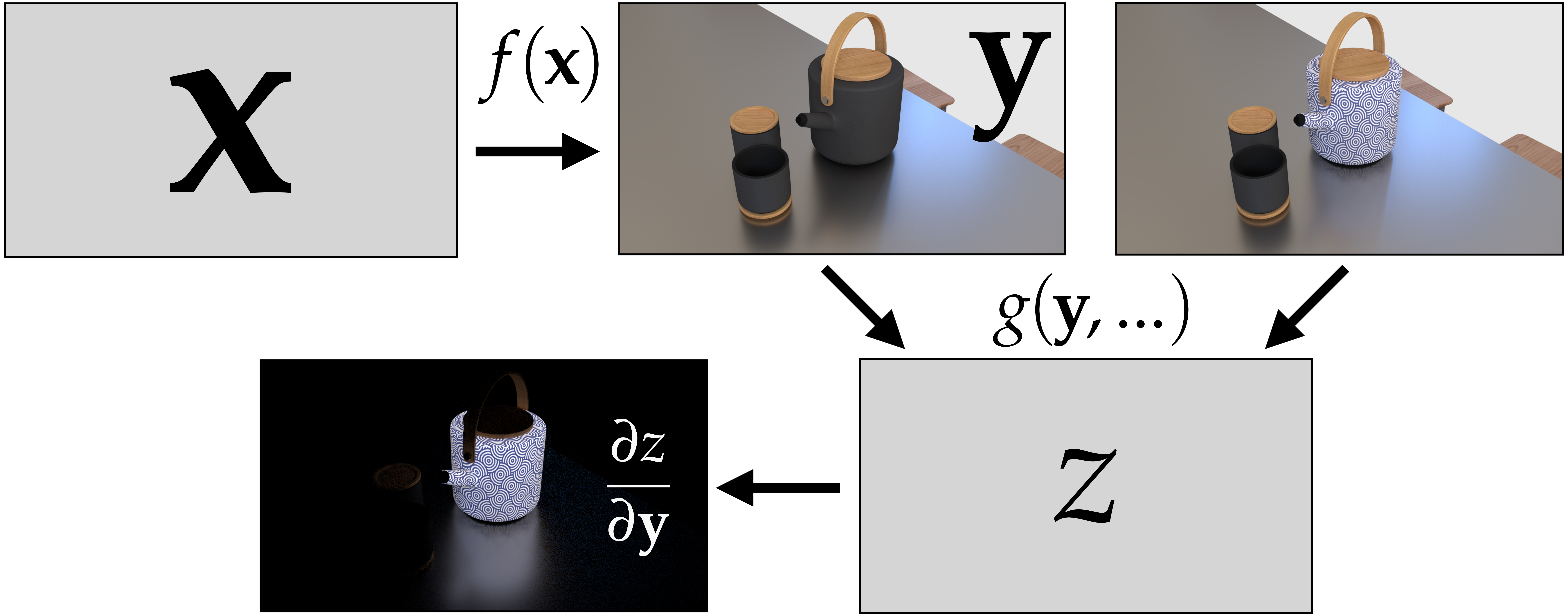


DIFFERENTIABLE RENDERING

Vector

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

CHAIN RULE

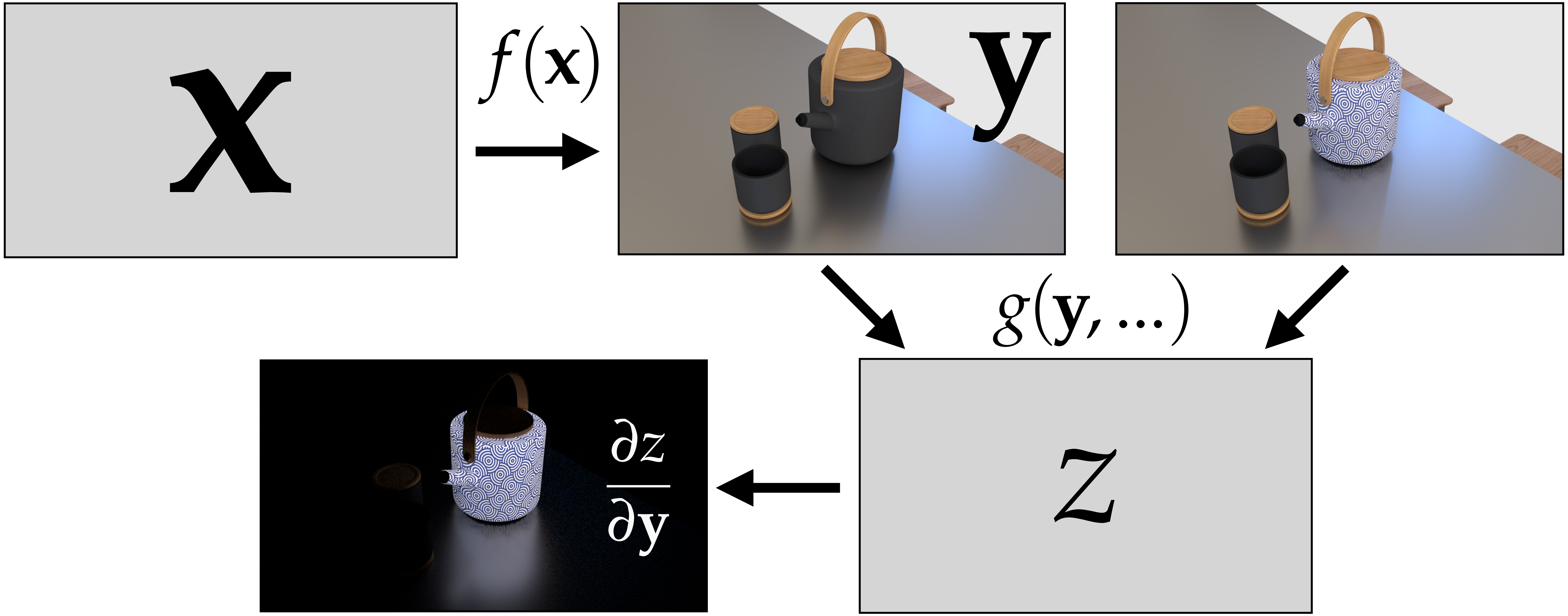


DIFFERENTIABLE RENDERING

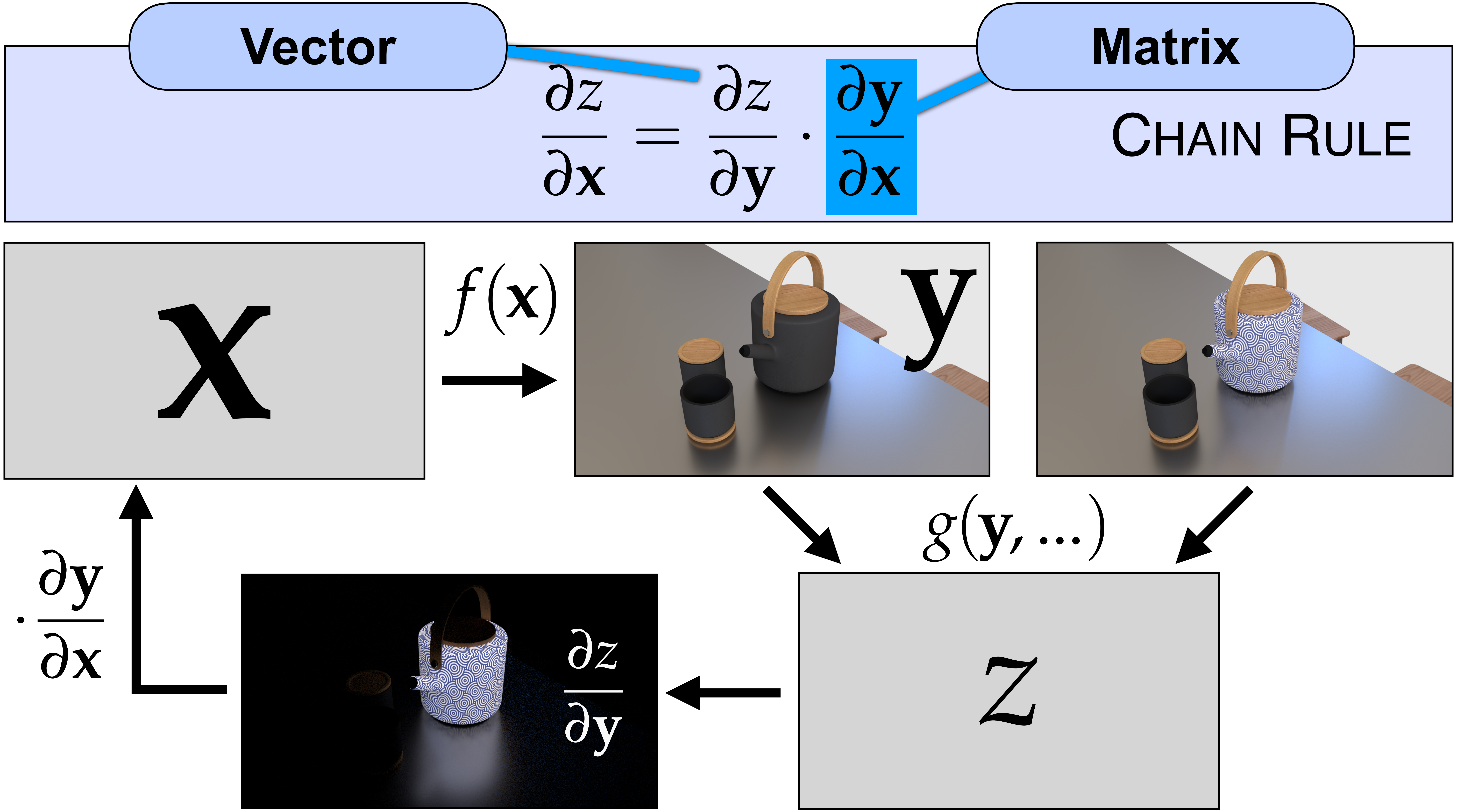
Vector | Matrix

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

CHAIN RULE



DIFFERENTIABLE RENDERING



DIFFERENTIABLE RENDERING

$\mathbf{x}$

$\cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

$\frac{\partial z}{\partial \mathbf{y}}$

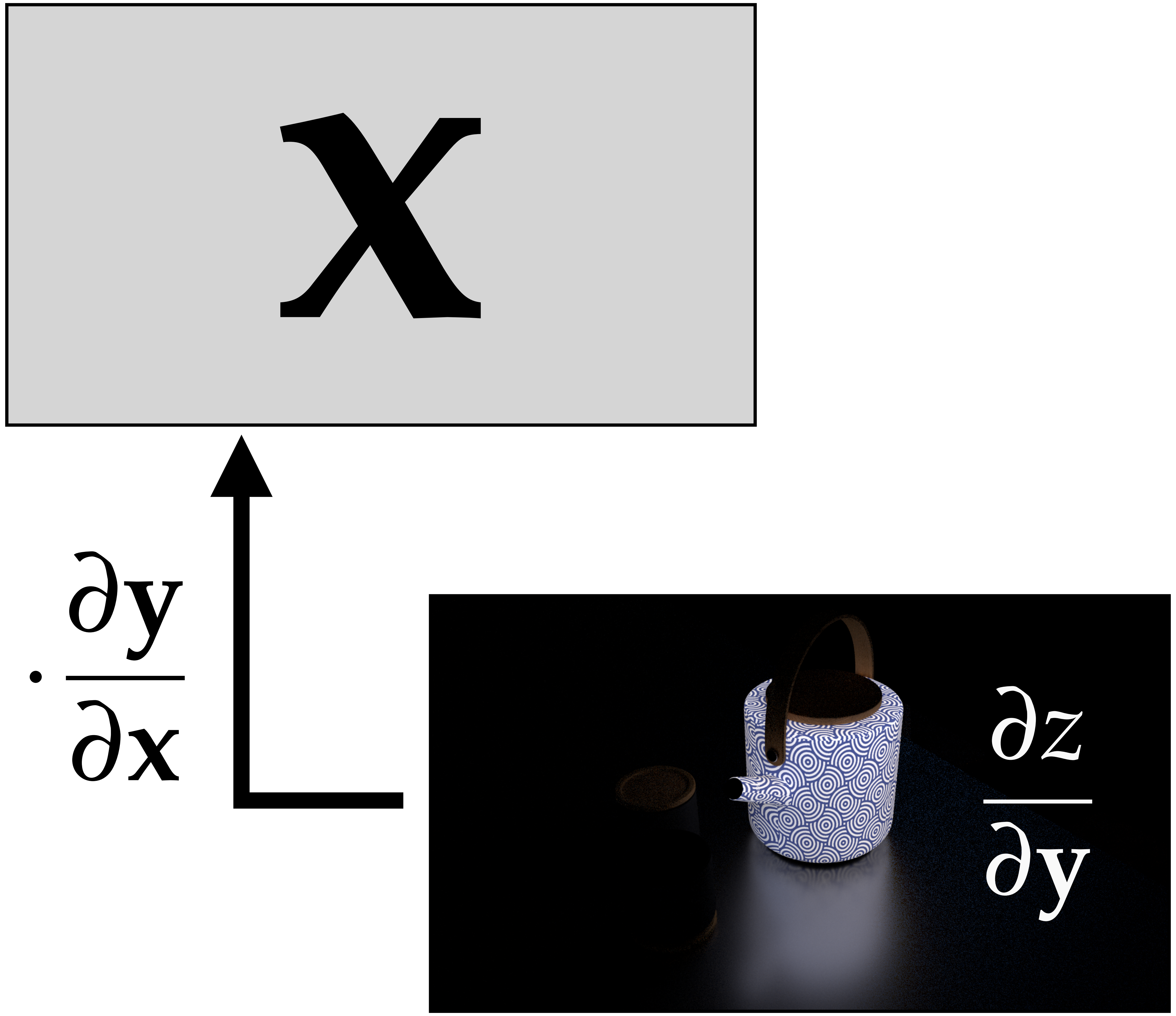
DIFFERENTIABLE RENDERING

$$\mathbf{x}$$

$$\cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

$$\frac{\partial z}{\partial \mathbf{y}}$$

DIFFERENTIABLE RENDERING



DIFFERENTIABLE RENDERING

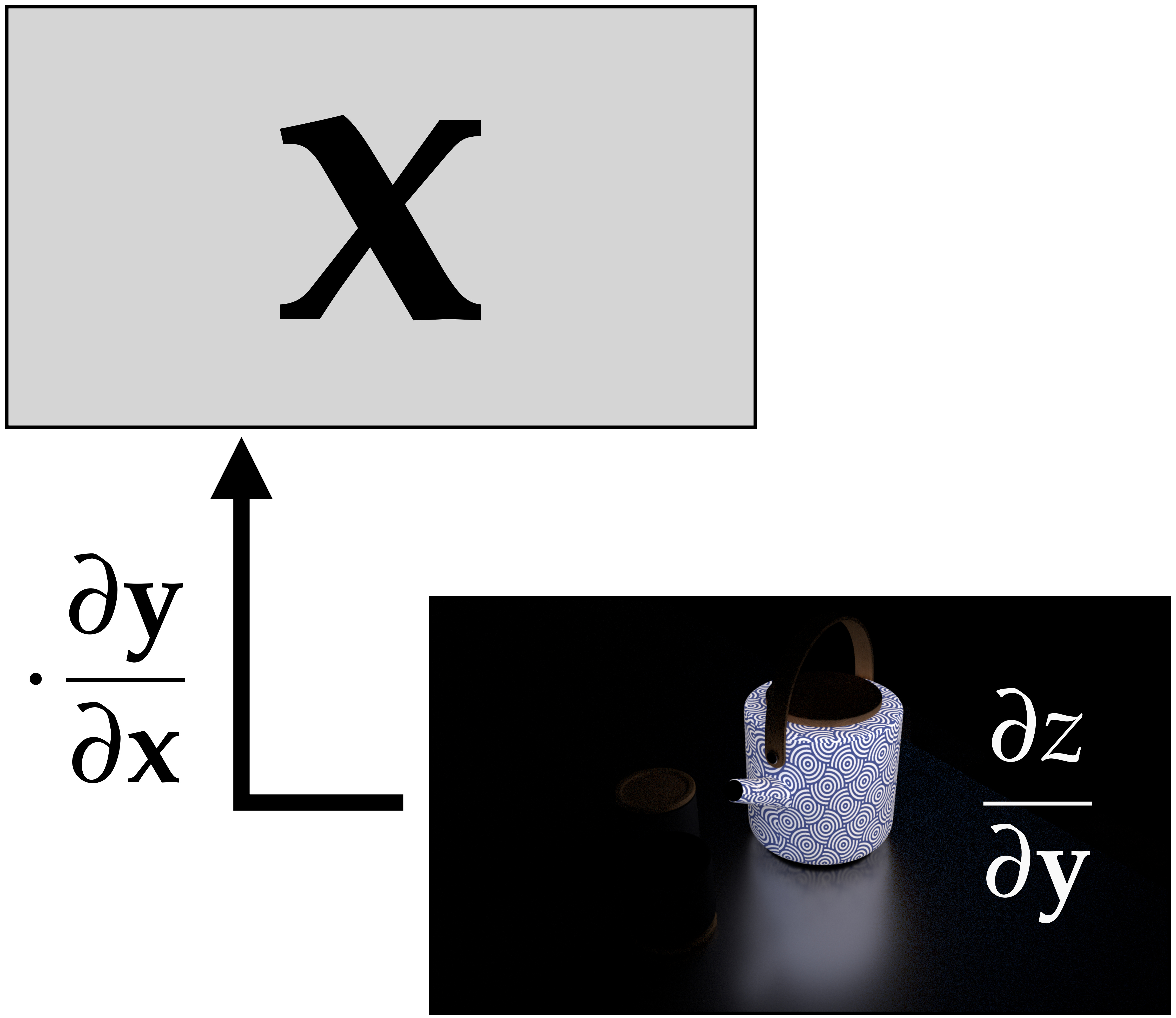
$\mathbf{x}$

$\cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$

$\frac{\partial z}{\partial \mathbf{y}}$

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

DIFFERENTIABLE RENDERING



$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

Challenges

1. Differentiating f
2. Matrix multiplication
3. Efficiency?
4. How to deal with edges?

HOW TO DO THIS (AT ALL?)

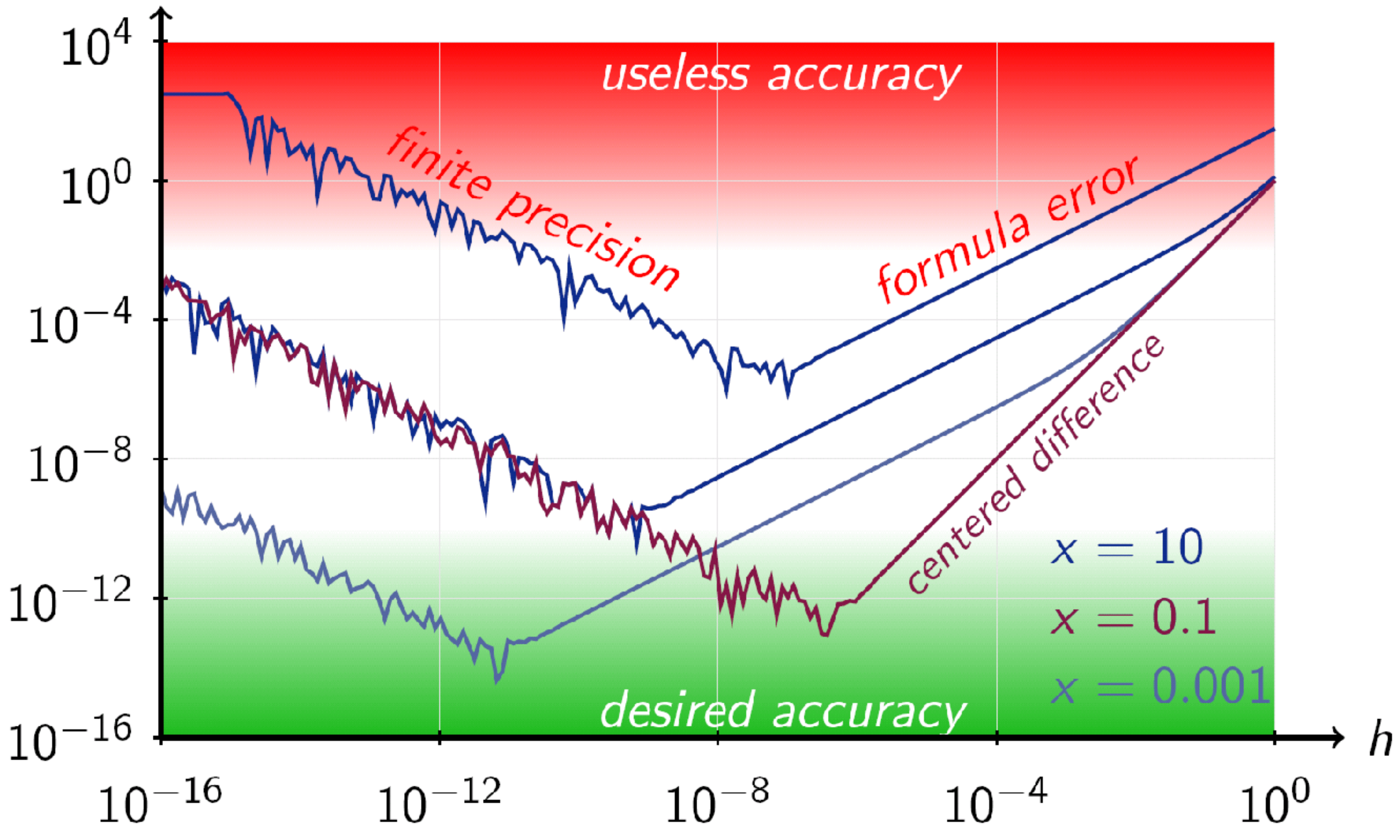
Use finite differences!

$$\frac{\partial \mathbf{y}}{\partial x_i} = \frac{f(\mathbf{x} + \varepsilon\,\mathbf{e}_i) - f(\mathbf{x} - \varepsilon\,\mathbf{e}_i)}{2\,\varepsilon}$$

HOW TO DO THIS (AT ALL?)

Use finite differences!

$$\frac{\partial \mathbf{y}}{\partial x_i} = \frac{f(\mathbf{x} + \varepsilon\,\mathbf{e}_i) - f(\mathbf{x} - \varepsilon\,\mathbf{e}_i)}{2\,\varepsilon}$$



[Wikipedia]

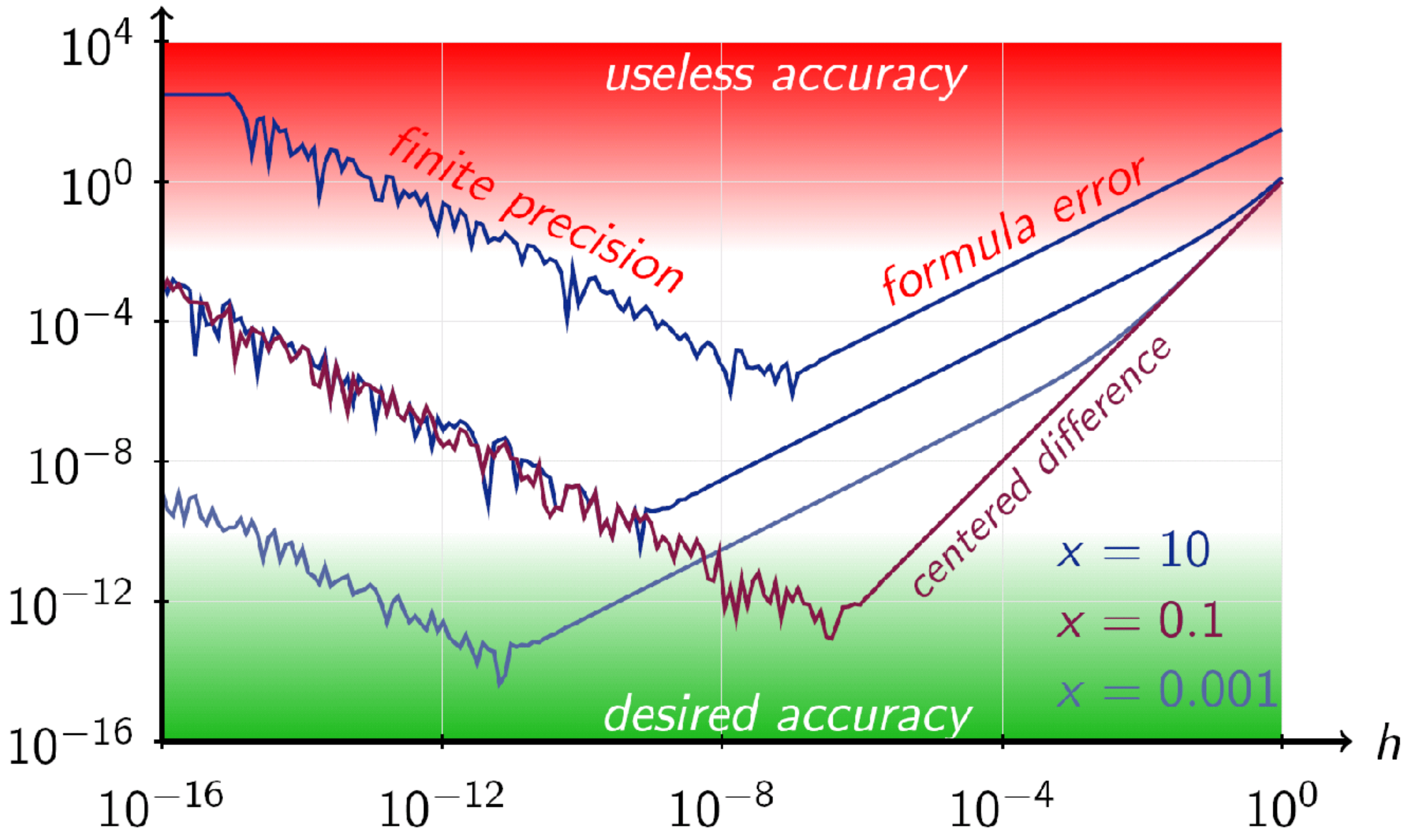
Potential problems:

- Bad approximation (big ε), rounding error (small ε)

HOW TO DO THIS (AT ALL?)

Use finite differences!

$$\frac{\partial \mathbf{y}}{\partial x_i} = \frac{f(\mathbf{x} + \varepsilon\, \mathbf{e}_i) - f(\mathbf{x} - \varepsilon\, \mathbf{e}_i)}{2\,\varepsilon}$$



[Wikipedia]

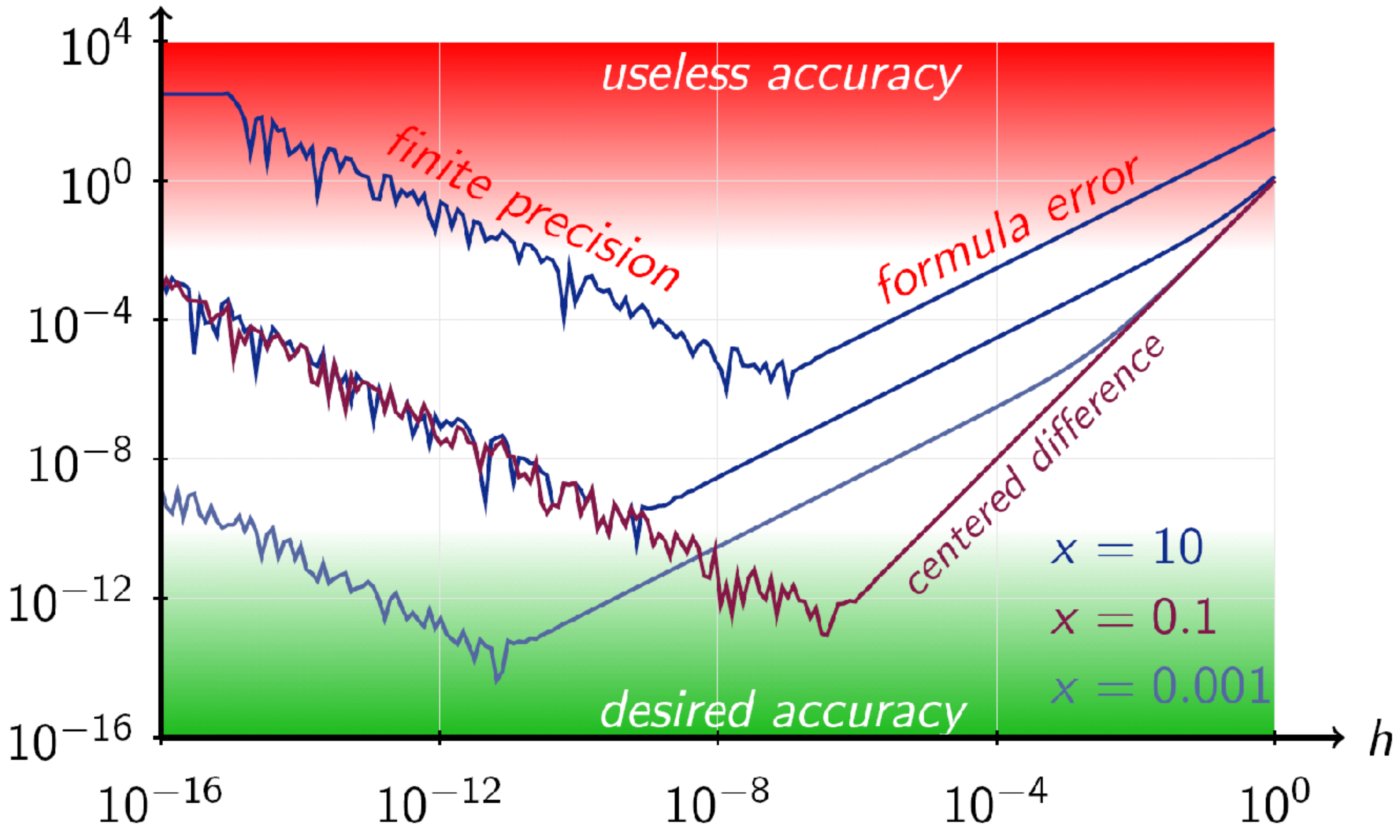
Potential problems:

- Bad approximation (big ε), rounding error (small ε)
- Need to correlate Monte Carlo samples

HOW TO DO THIS (AT ALL?)

Use finite differences!

$$\frac{\partial \mathbf{y}}{\partial x_i} = \frac{f(\mathbf{x} + \varepsilon\, \mathbf{e}_i) - f(\mathbf{x} - \varepsilon\, \mathbf{e}_i)}{2\,\varepsilon}$$



[Wikipedia]

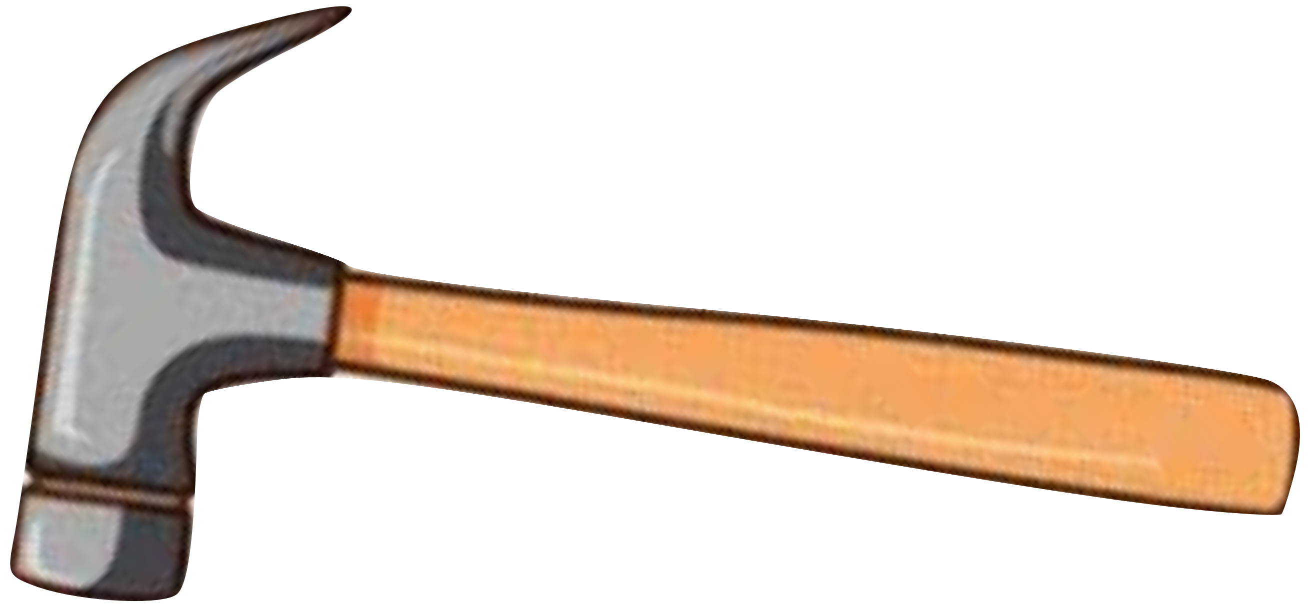
Potential problems:

- Bad approximation (big ε), rounding error (small ε)
- Need to correlate Monte Carlo samples
- Extremely slow when many there are many parameters.

AUTOMATIC DIFFERENTIATION

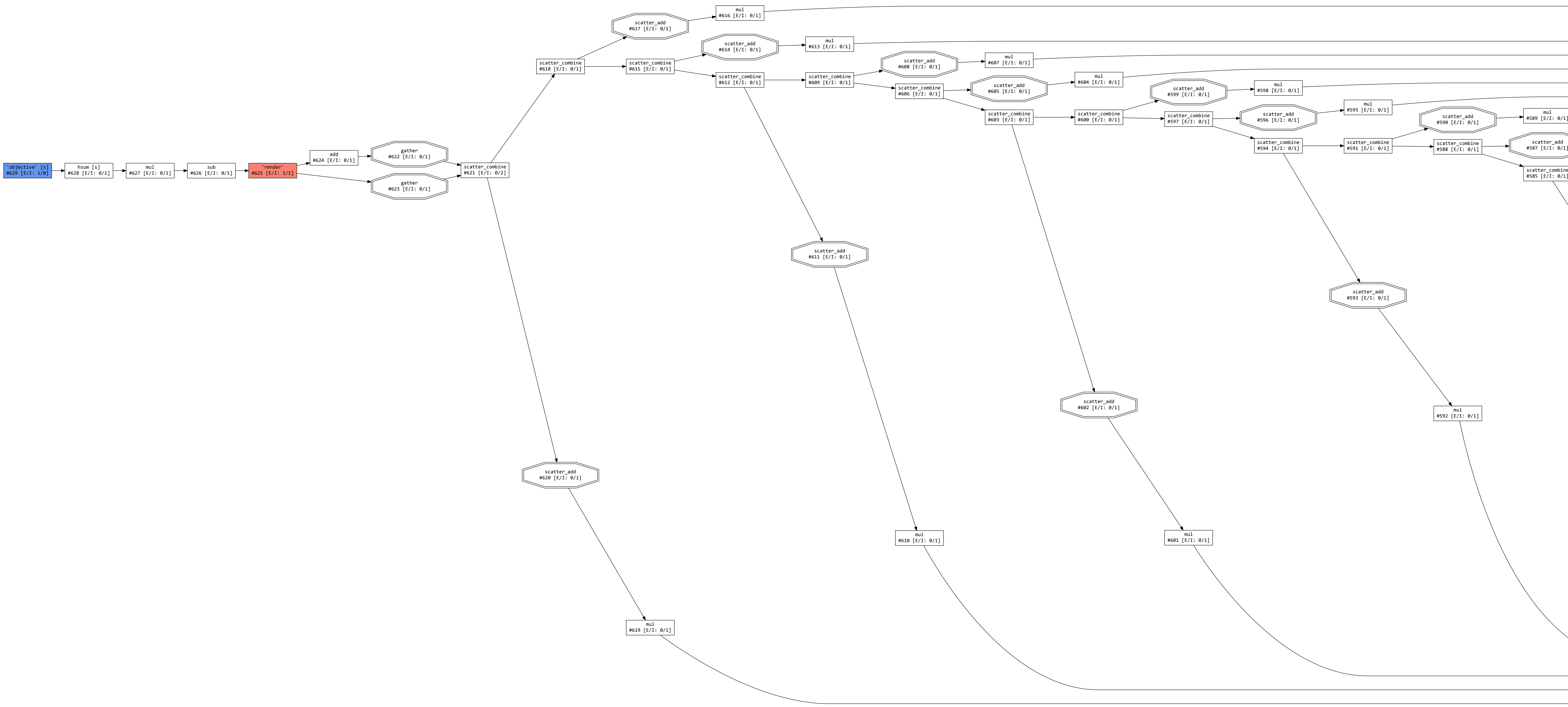
$$f(\mathbf{x})$$

AUTOMATIC DIFFERENTIATION

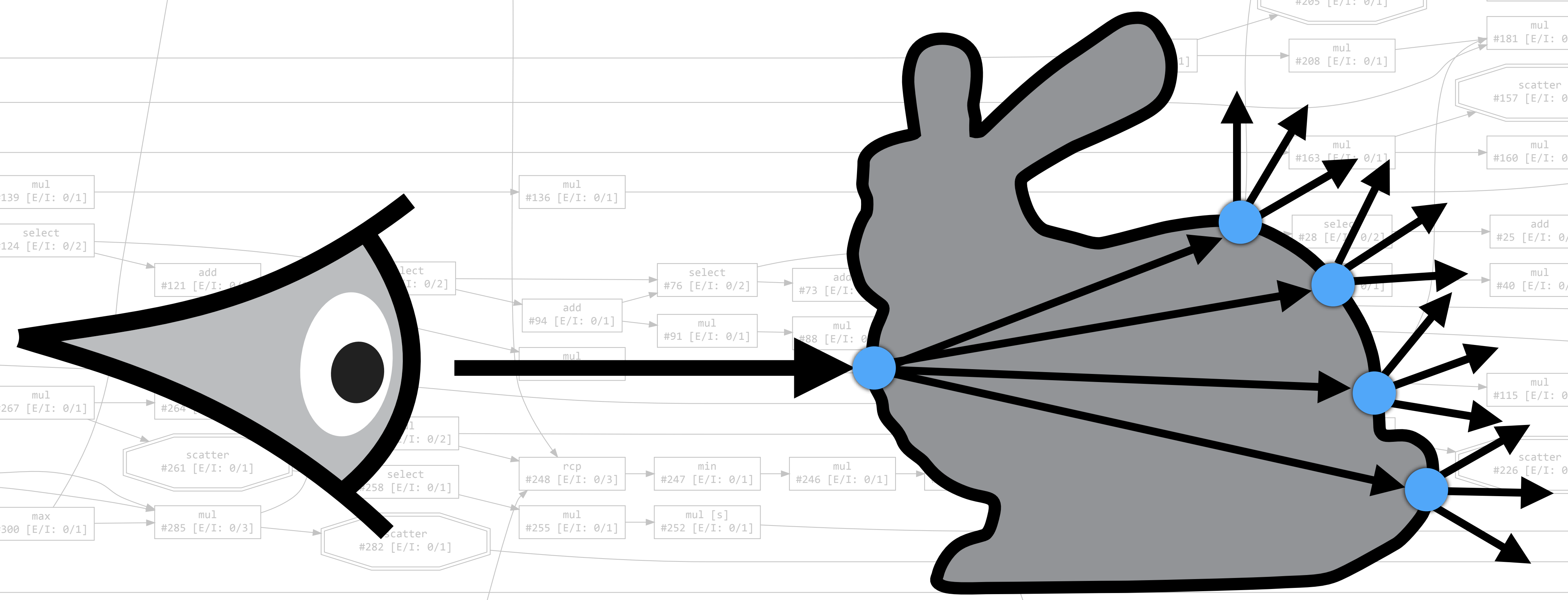


$$f(\mathbf{x}) \xrightarrow{\text{AD}} \frac{\partial}{\partial \mathbf{x}} f(\mathbf{x})$$

ISSUES WITH AUTOMATIC DIFFERENTIATION (AD)



ISSUES WITH AUTOMATIC DIFFERENTIATION (AD)



WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters

Estimate $\int_0^\infty f(\lambda, x)\,\mathrm{d}x$ (with λ given)

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ # This is the pdf of $\mathrm{Exp}[\lambda]$
- Return f/p

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters

Estimate $\int_0^\infty f(\lambda, x)\,\mathrm{d}x$ (with λ given)

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ # This is the pdf of $\mathrm{Exp}[\lambda]$
- Return f/p

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^\infty \frac{\partial f}{\partial \lambda}(\lambda, x)\,\mathrm{d}x$

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$
- $f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$
- Return f'/p

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  – Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters

Estimate $\int_0^\infty f(\lambda, x)\,\mathrm{d}x$ (with λ given)

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ # This is the pdf of $\mathrm{Exp}[\lambda]$
- Return f/p

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^\infty \frac{\partial f}{\partial \lambda}(\lambda, x)\,\mathrm{d}x$

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$ x has zero gradient
- $f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ p is NOT differentiated
- Return f'/p

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters, with $\xi = \mathrm{e}^{-\lambda x}$

Estimate $\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{f(\lambda, x)}{\lambda \xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ # $x \sim \mathrm{Exp}(\lambda)$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ # $p = \lambda \xi$
- Return f/p

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters, with $\xi = \mathrm{e}^{-\lambda x}$

Estimate $\displaystyle\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{f(\lambda, x)}{\lambda\xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ $\quad \# \, x \sim \mathrm{Exp}(\lambda)$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# \, p = \lambda\xi$
- Return f/p

⇒

Estimate $\displaystyle\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{\partial}{\partial\lambda}\frac{f(\lambda, x)}{\lambda\xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# \, p = \lambda\xi$
- Return $\partial(f/p)/\partial\lambda$

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters, with $\xi = \mathrm{e}^{-\lambda x}$

Estimate $\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{f(\lambda, x)}{\lambda \xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ $\quad \# x \sim \mathrm{Exp}(\lambda)$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# p = \lambda \xi$
- Return f/p

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda} \int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{\partial}{\partial \lambda} \frac{f(\lambda, x)}{\lambda \xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ $\quad$ x has nonzero gradient
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# p = \lambda \xi$
- Return $\partial(f/p)/\partial \lambda$ $\quad$ f and p are both differentiated

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^\infty \frac{\partial f}{\partial \lambda}(\lambda, x)\,\mathrm{d}x$

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$ — x has zero gradient
- $f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ — p is NOT differentiated
- Return f'/p

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda, x)\,\mathrm{d}x = \int_0^1 \frac{\partial}{\partial \lambda}\frac{f(\lambda, x)}{\lambda \xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ — x has nonzero gradient
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# \, p = \lambda\xi$
- Return $\partial(f/p)/\partial\lambda$ — f and p are both differentiated

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 1:** Distributional parameters

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda,x)\,\mathrm{d}x = \int_0^\infty \frac{\partial f}{\partial \lambda}(\lambda,x)\,\mathrm{d}x$

(Single-sample) Monte Carlo estimator:

- Draw $x \sim \mathrm{Exp}[\lambda]$ — x has zero gradient
- $f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda,x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ — p is NOT differentiated
- Return f'/p

Estimate $\frac{\mathrm{d}}{\mathrm{d}\lambda}\int_0^\infty f(\lambda,x)\,\mathrm{d}x = \int_0^1 \frac{\partial}{\partial \lambda}\frac{f(\lambda,x)}{\lambda\xi}\,\mathrm{d}\xi$

(Single-sample) Monte Carlo estimator:

- Draw $\xi \sim U[0,1)$
- $x \leftarrow -\log(\xi)/\lambda$ — x has nonzero gradient
- $f \leftarrow f(\lambda,x)$
- $p \leftarrow \lambda \mathrm{e}^{-\lambda x}$ $\quad \# \, p = \lambda\xi$
- Return $\partial(f/p)/\partial\lambda$ — f and p are both differentiated

Whether to differentiate the *sampling* and the *pdf* should be **consistent**!

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 2:** Discontinuities

Estimate $\int_0^1 (x < p\ ?\ 1 : 0.5)\,\mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0,1)$
- Return $X < p\ ?\ 1 : 0.5$

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 2:** Discontinuities

Estimate $\int_0^1 (x < p \ ? \ 1 : 0.5) \, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $X < p \ ? \ 1 : 0.5$

Ground-truth:

$$\int_0^1 (x < p \ ? \ 1 : 0.5) \, \mathrm{d}x = \int_0^p \mathrm{d}x + \int_p^1 0.5 \, \mathrm{d}x = \frac{1 + p}{2}$$

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  – Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 2:** Discontinuities

Estimate $\int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $X < p \; ? \; 1 : 0.5$

Ground-truth:

$$\int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x = \int_0^p \mathrm{d}x + \int_p^1 0.5 \, \mathrm{d}x = \frac{1+p}{2}$$

Estimate $\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $\mathrm{d}(X < p \; ? \; 1 : 0.5)/\mathrm{d}p$

Ground-truth:

$$\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}p} \frac{1+p}{2} = \frac{1}{2}$$

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 2:** Discontinuities

Estimate $\int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $X < p \; ? \; 1 : 0.5$

Ground-truth:

$$\int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x = \int_0^p \mathrm{d}x + \int_p^1 0.5 \, \mathrm{d}x = \frac{1+p}{2}$$

Estimate $\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $\mathrm{d}(X < p \; ? \; 1 : 0.5)/\mathrm{d}p$ Zero! (constant)

$\neq$

Ground-truth:

$$\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \; ? \; 1 : 0.5) \, \mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}p} \frac{1+p}{2} = \frac{1}{2}$$

WHY IS DIFFERENTIABLE RENDERING DIFFICULT

- Precautions must be taken to ensure **correctness**
  - Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

- **Example 2:** Discontinuities

Estimate $\int_0^1 (x < p \ ? \ 1 : 0.5)\, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $X < p \ ? \ 1 : 0.5$

Ground-truth:

$$\int_0^1 (x < p \ ? \ 1 : 0.5)\, \mathrm{d}x = \int_0^p \mathrm{d}x + \int_p^1 0.5\, \mathrm{d}x = \frac{1+p}{2}$$

Estimate $\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \ ? \ 1 : 0.5)\, \mathrm{d}x$ with $0 < p < 1$

(Single-sample) Monte Carlo estimator:

- Draw $X \sim U[0, 1)$
- Return $\mathrm{d}(X < p \ ? \ 1 : 0.5)/\mathrm{d}p$ Zero! (constant)

$\nparallel$

Ground-truth:

$$\frac{\mathrm{d}}{\mathrm{d}p} \int_0^1 (x < p \ ? \ 1 : 0.5)\, \mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}p} \frac{1+p}{2} = \frac{1}{2}$$

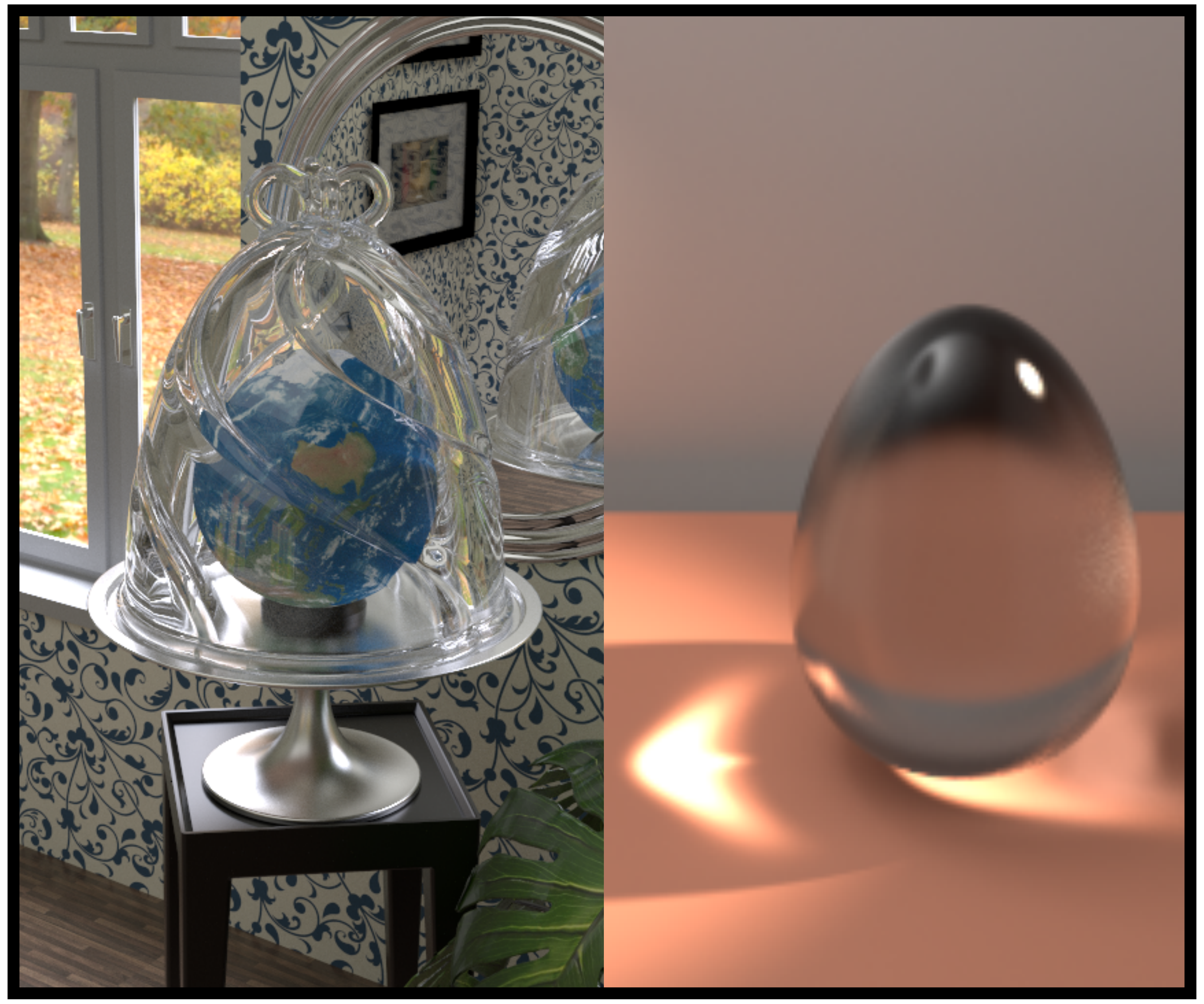
More on this example later

COURSE OUTLINE

Basics

COURSE OUTLINE

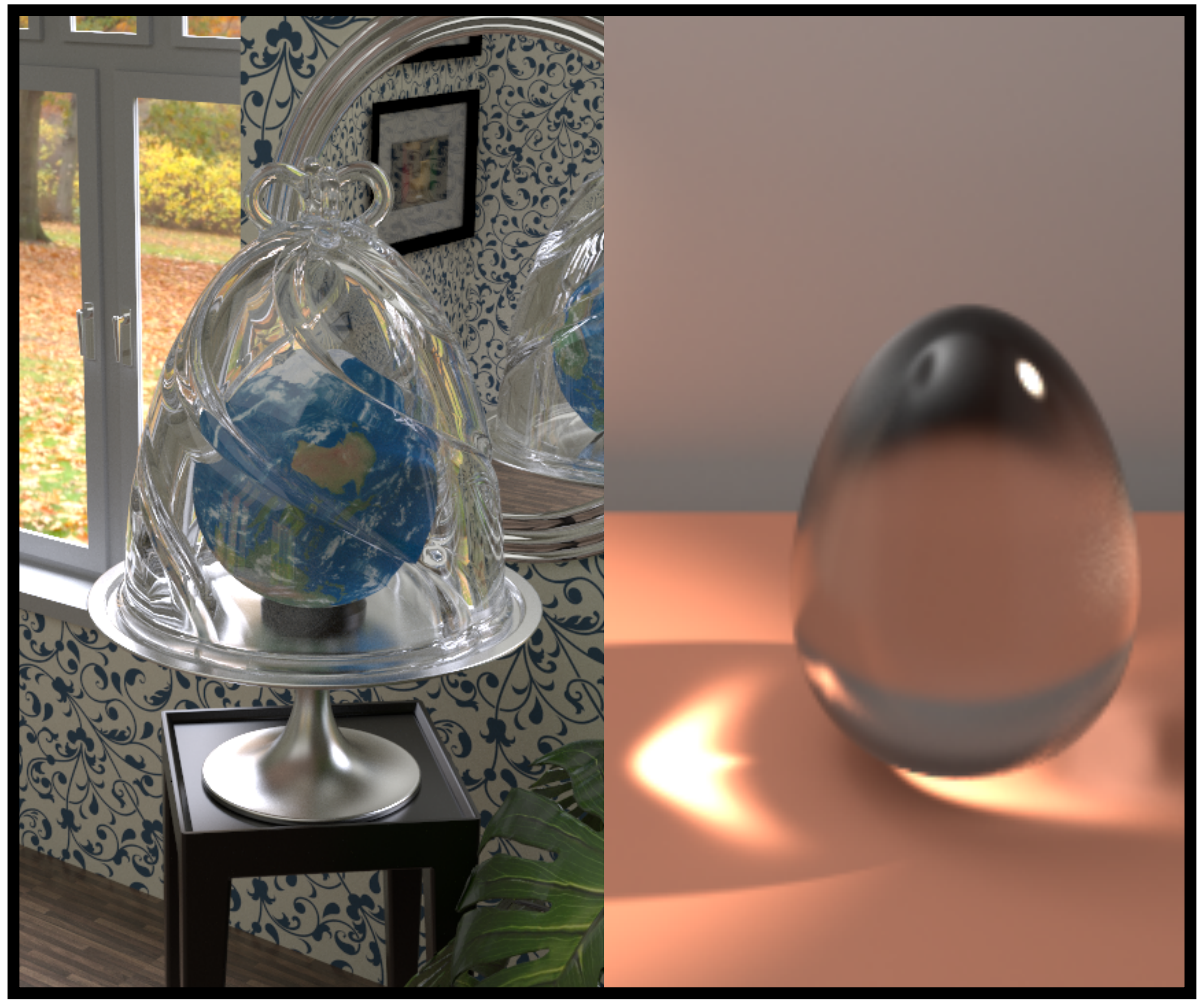
Basics



State-of-the-art theories and algorithms

COURSE OUTLINE

Basics



State-of-the-art theories and algorithms

Implementation details

BASICS

DIFFERENTIATING (RENDERING) PROGRAMS

- a crash course on automatic differentiation
- differentiating discontinuities in rendering
- discussions & limitations

∇

```
auto scatter_contrib = Vector3{0, 0, 0};
auto scatter_bsdf = Vector3{0, 0, 0};
if (bsdf_isect.valid()) {
    const auto &bsdf_shape = scene.shapes[bsdf_isect.shape_id];
    auto dir = bsdf_point.position - p;
    auto dist_sq = length_squared(dir);
    auto wo = dir / sqrt(dist_sq);
    auto pdf_bsdf = bsdf_pdf(material, shading_point, wi, wo, min_rough);
    if (dist_sq > 1e-20f && pdf_bsdf > 1e-20f) {
        auto bsdf_val = bsdf(material, shading_point, wi, wo, min_rough);
        if (bsdf_shape.light_id >= 0) {
            const auto &light = scene.area_lights[bsdf_shape.light_id];
            if (light.two_sided || dot(-wo, bsdf_point.shading_frame.n) >
                auto light_contrib = light.intensity;
                auto light_pmf = scene.light_pmf[bsdf_shape.light_id];
                auto light_area = scene.light_areas[bsdf_shape.light_id];
                auto inv_area = 1 / light_area;
                auto geometry_term = fabs(dot(wo, bsdf_point.geom_normal))
                auto pdf_nee = (light_pmf * inv_area) / geometry_term;
                auto mis_weight = Real(1 / (1 + square((double)pdf_nee / (
                scatter_contrib = (mis_weight / pdf_bsdf) * bsdf_val * lig
            }
        }
        scatter_bsdf = bsdf_val / pdf_bsdf;
        next_throughput = throughput * scatter_bsdf;
```

A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

- automatic differentiation v.s. symbolic differentiation

```
function f(x):
  result = x
  for i = 1 to 8:
    result = exp(result)
  return result
```

A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

- automatic differentiation v.s. symbolic differentiation

```
function f(x):
  result = x
  for i = 1 to 8:
    result = exp(result)
  return result
```

symbolic differentiation (37 exponents):

$$\frac{df(x)}{dx} = e^{x+e^{e^{e^{e^{e^{e^{e^{x}}}}}}}+e^{e^{e^{e^{e^{e^{x}}}}}}+e^{e^{e^{e^{e^{x}}}}}+e^{e^{e^{e^{x}}}}+e^{e^{e^{x}}}+e^{e^{x}}+e^{x}}$$

- automatic differentiation v.s. symbolic differentiation

```
function f(x):
  result = x
  for i = 1 to 8:
    result = exp(result)
  return result
```

symbolic differentiation (37 exponents):

$$\frac{df(x)}{dx} = e^{x+e^{e^{e^{e^{e^{e^{e^{x}}}}}}}+e^{e^{e^{e^{e^{e^{x}}}}}}+e^{e^{e^{e^{e^{x}}}}}+e^{e^{e^{e^{x}}}}+e^{e^{e^{x}}}+e^{e^{x}}+e^{x}}$$

forward-mode automatic differentiation
(8 exponents):

```
function d_f(x):
  result = x
  d_result = 1
  for i = 1 to 8:
    result = exp(result)
    d_result = d_result * result
  return d_result
```

A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

- key idea: chain rules, but applied in a smart way

```
y = f(x)
z = g(y)
```

A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

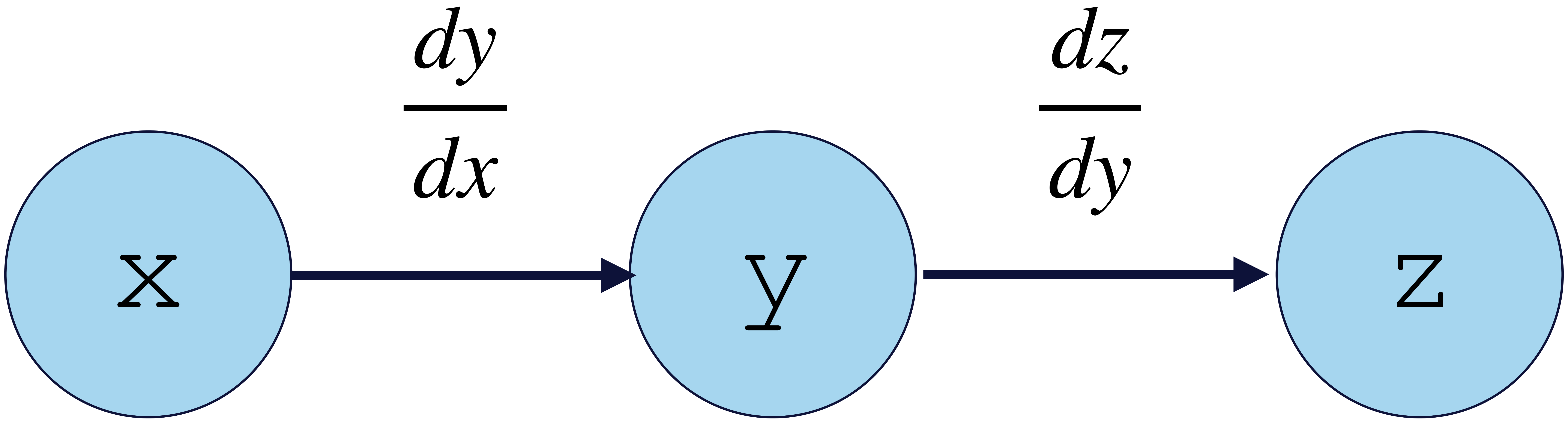
- key idea: chain rules, but applied in a smart way

```
y = f(x)
z = g(y)
```

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$$

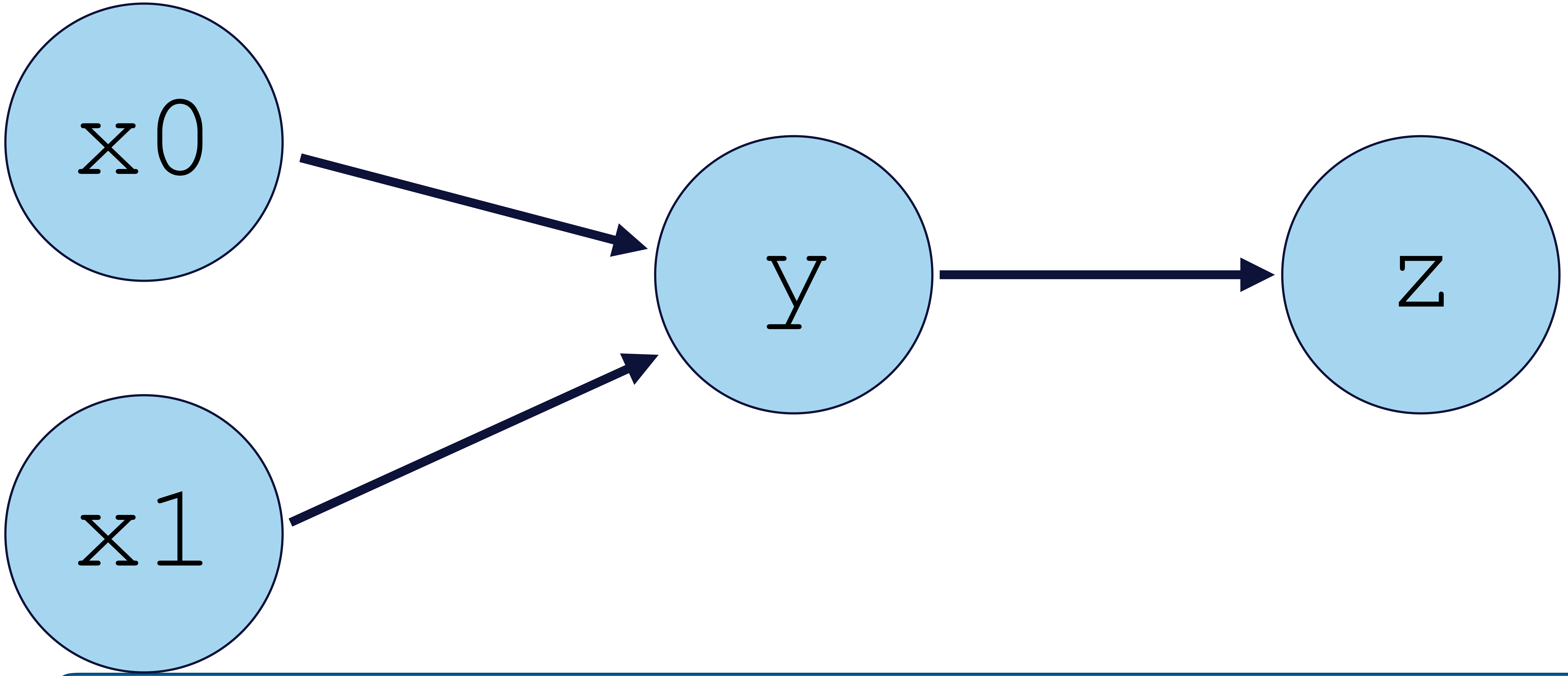
MENTAL MODEL: COMPUTATIONAL GRAPH

```
y = f(x)
z = g(y)
```



MULTIVARIATE EXAMPLE

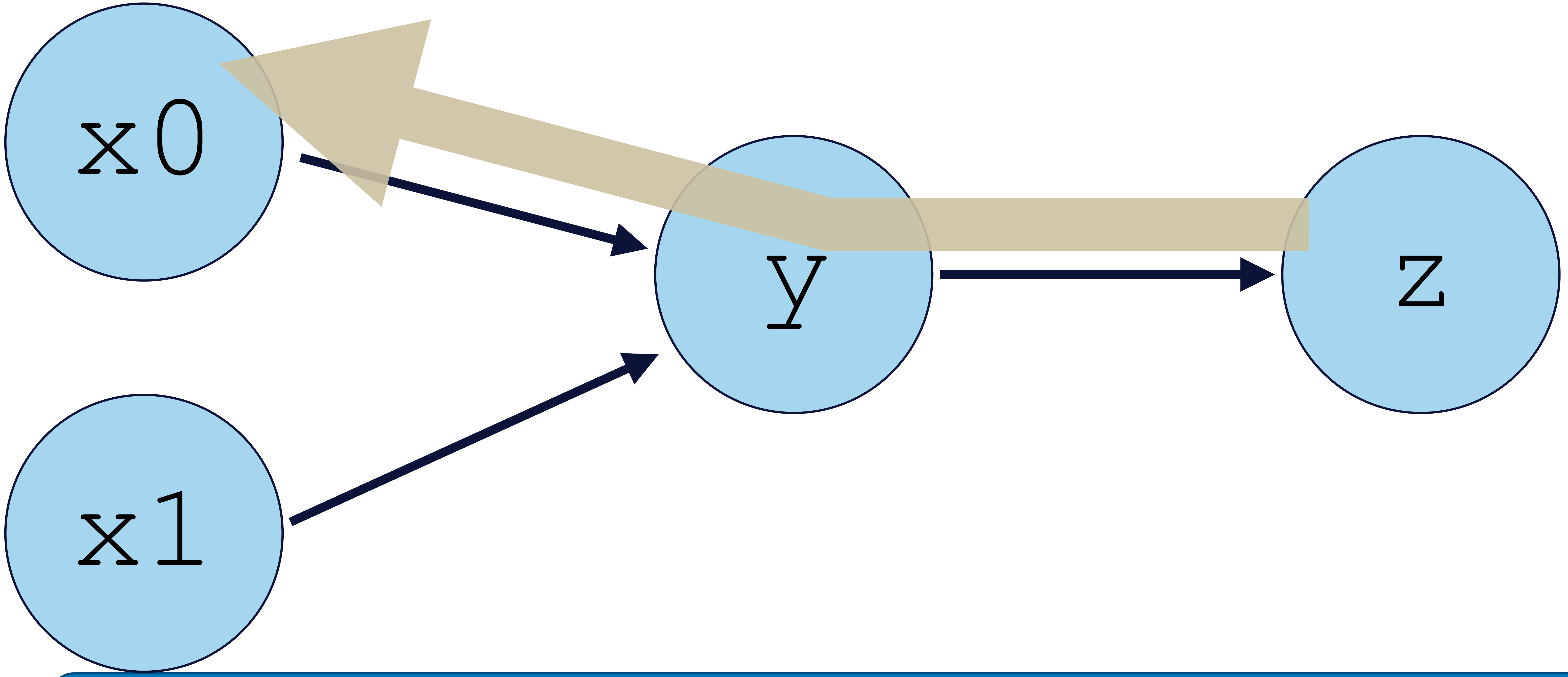
```
y = f(x0, x1)
z = g(y)
```



MULTIVARIATE EXAMPLE

```
y = f(x0, x1)
z = g(y)
```

$$\frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_0}$$

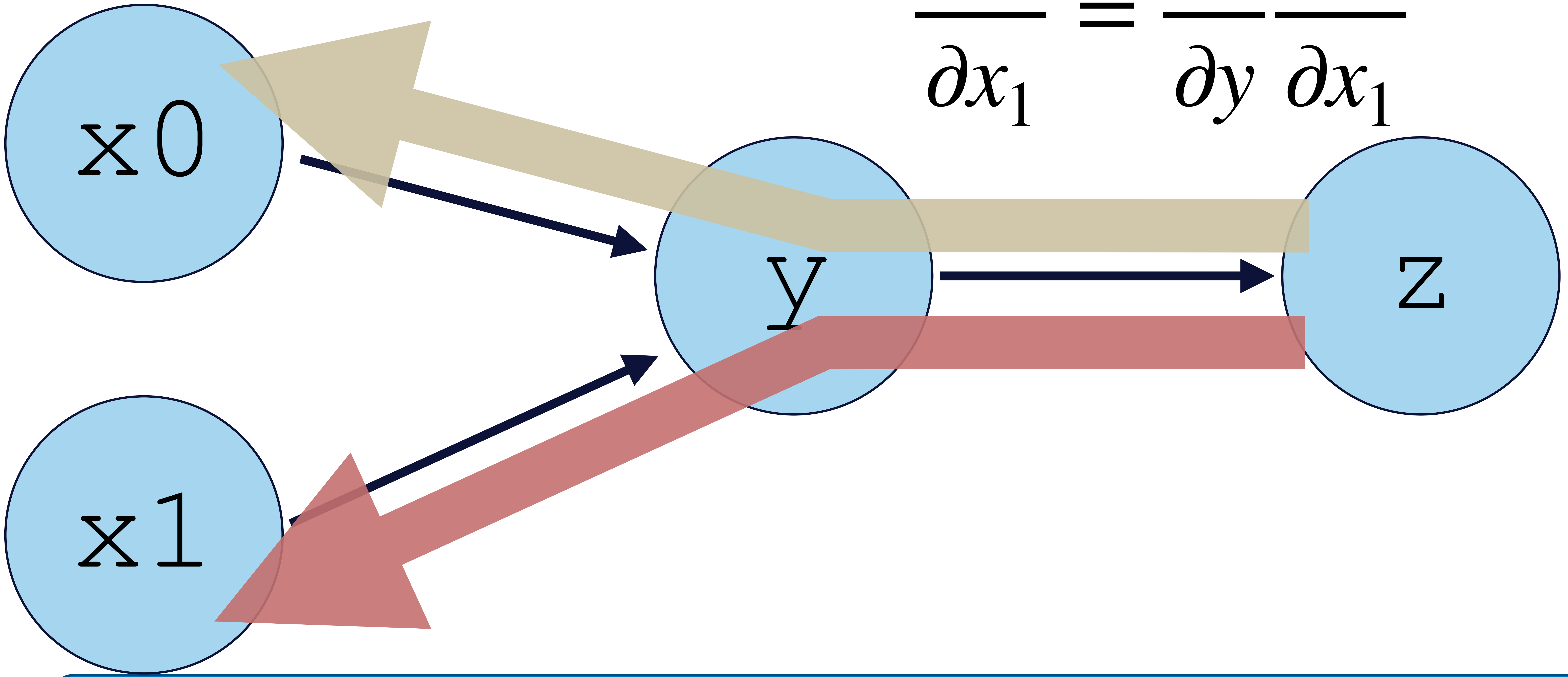


MULTIVARIATE EXAMPLE

```
y = f(x0, x1)
z = g(y)
```

$$\frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_0}$$

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_1}$$

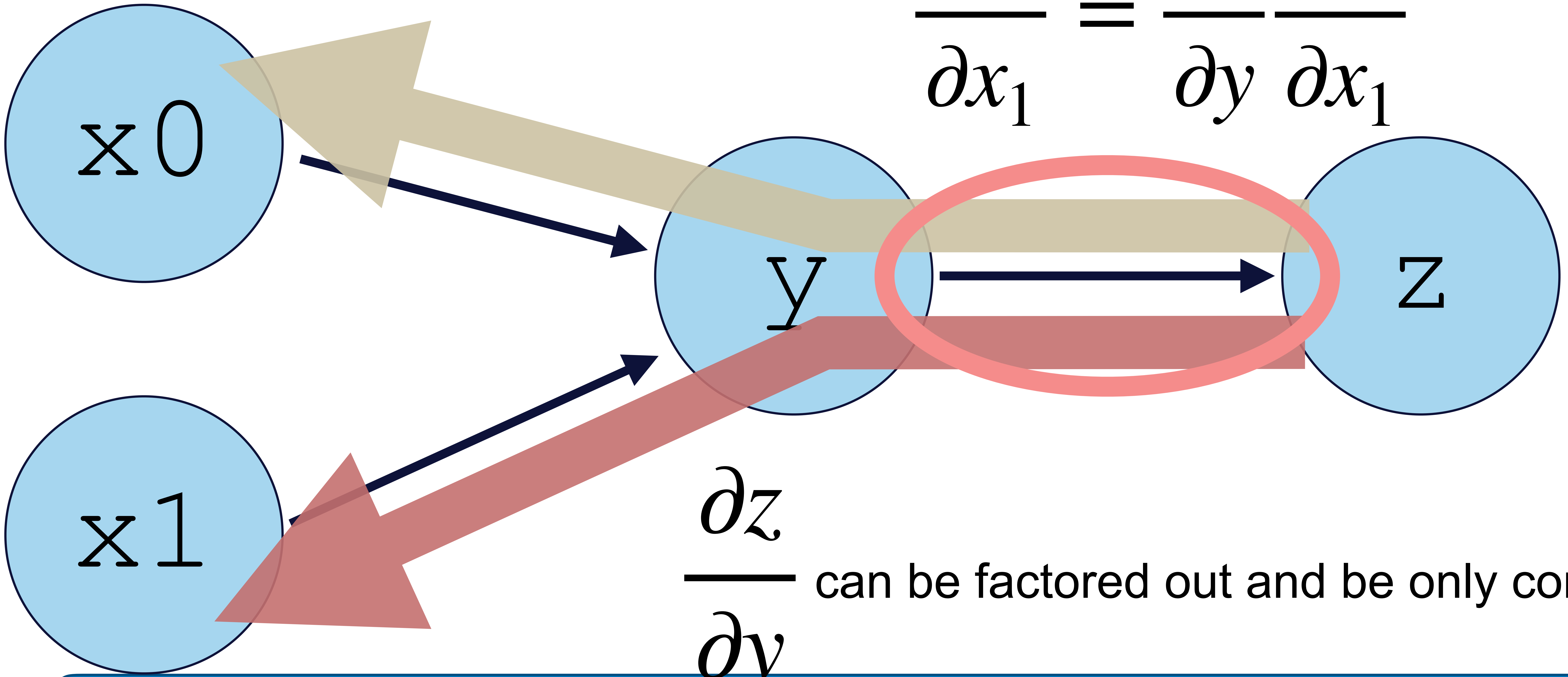


MULTIVARIATE EXAMPLE

```
y = f(x0, x1)
z = g(y)
```

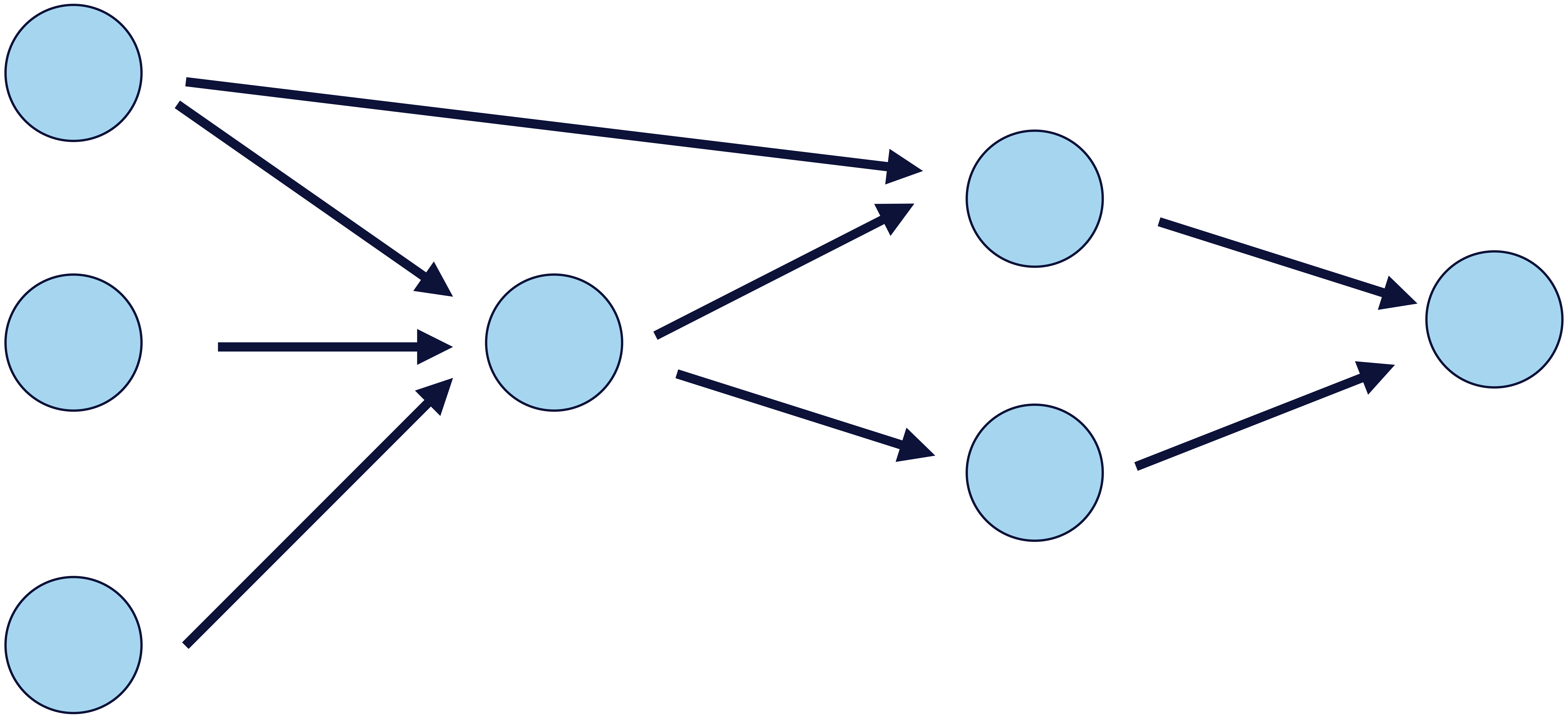
$$\frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_0}$$

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x_1}$$

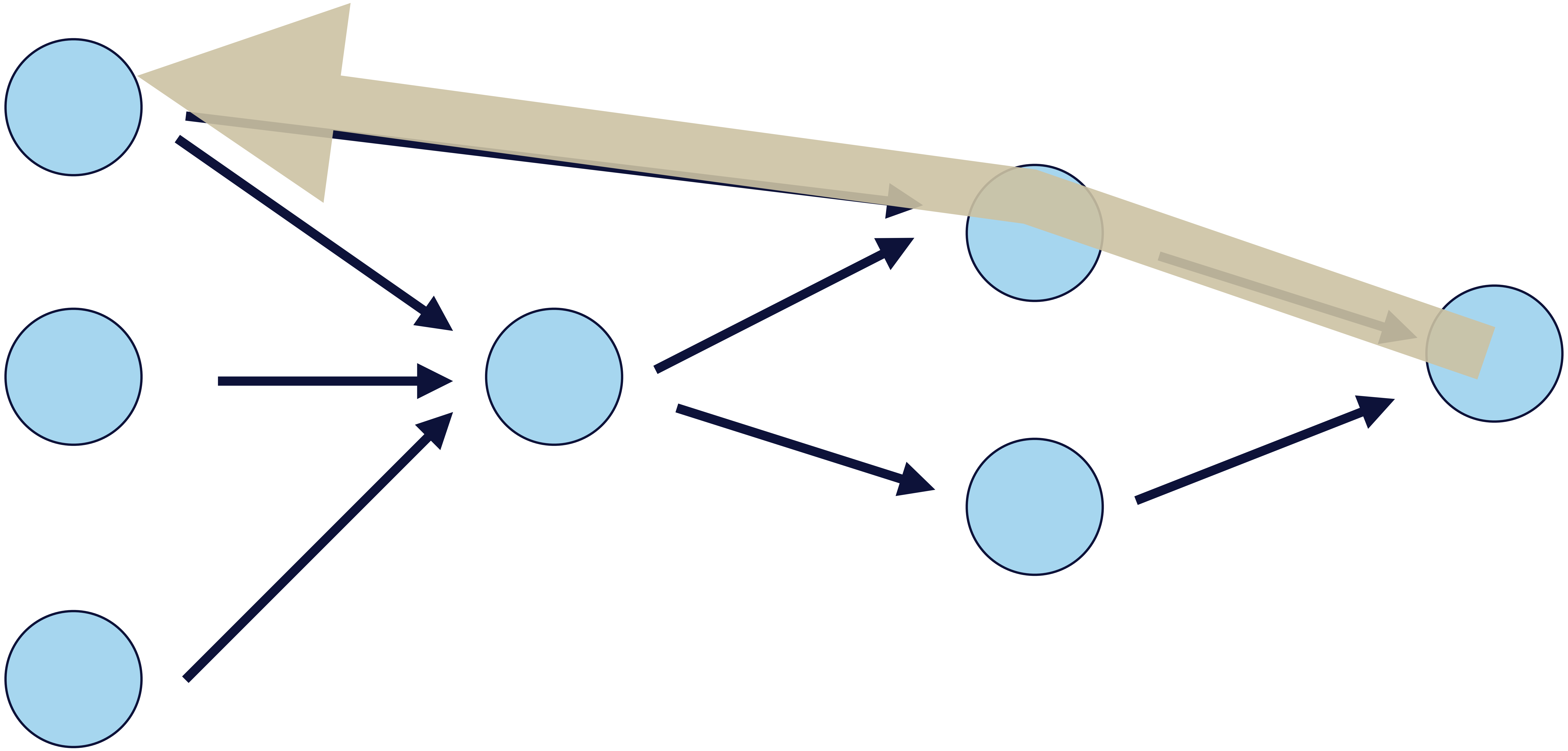


$\frac{\partial z}{\partial y}$ can be factored out and be only computed once!

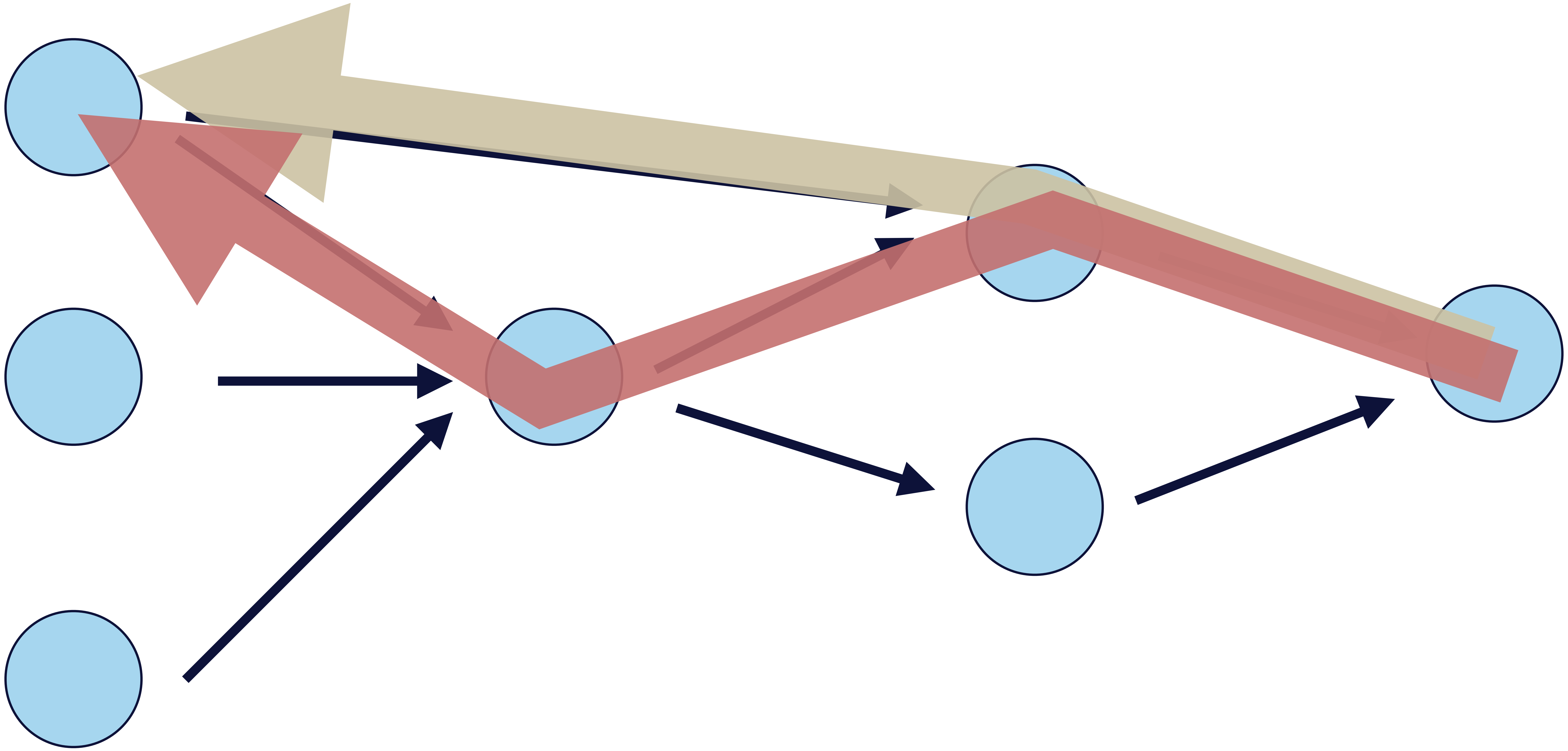
AUTODIFF = A PATH FINDING PROBLEM



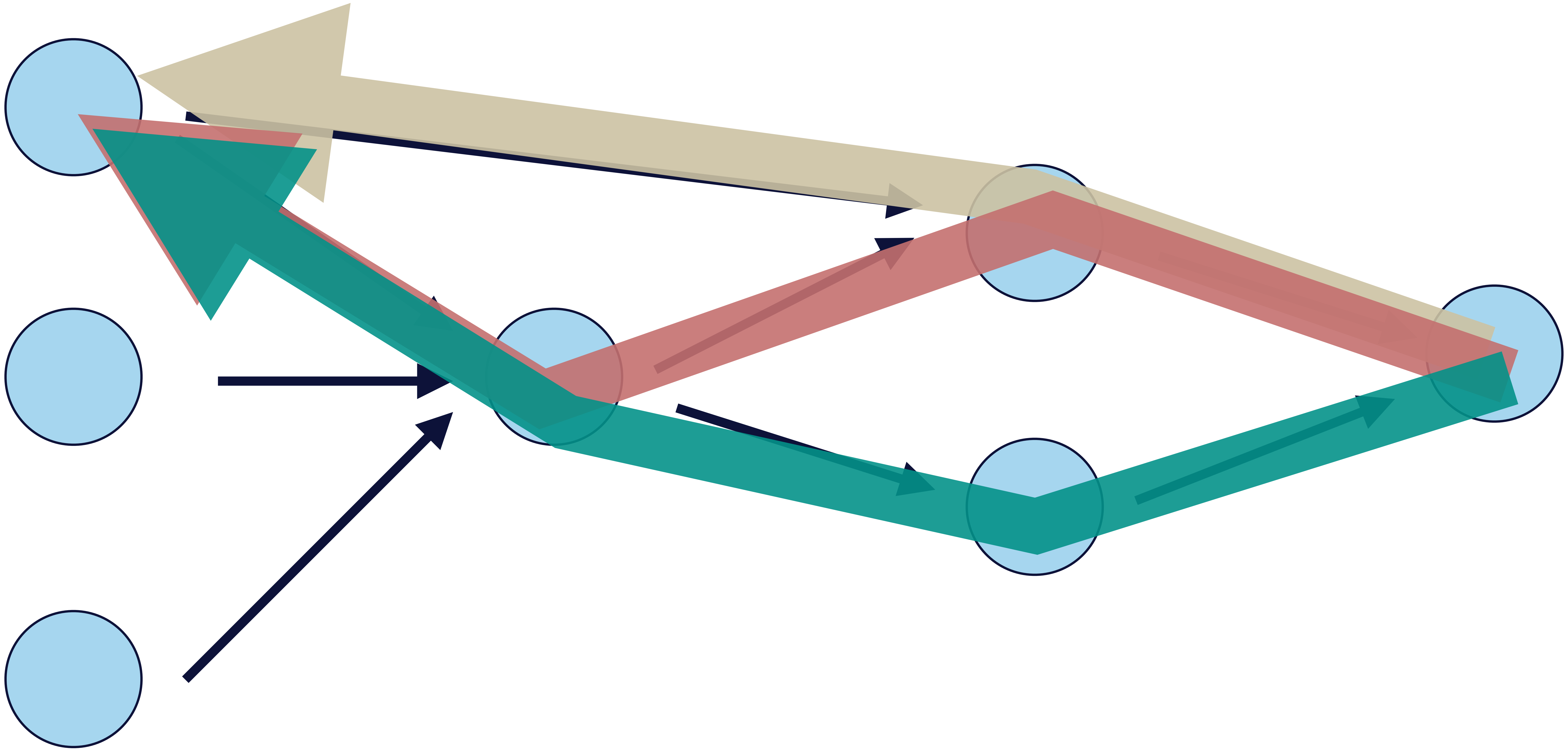
AUTODIFF = A PATH FINDING PROBLEM



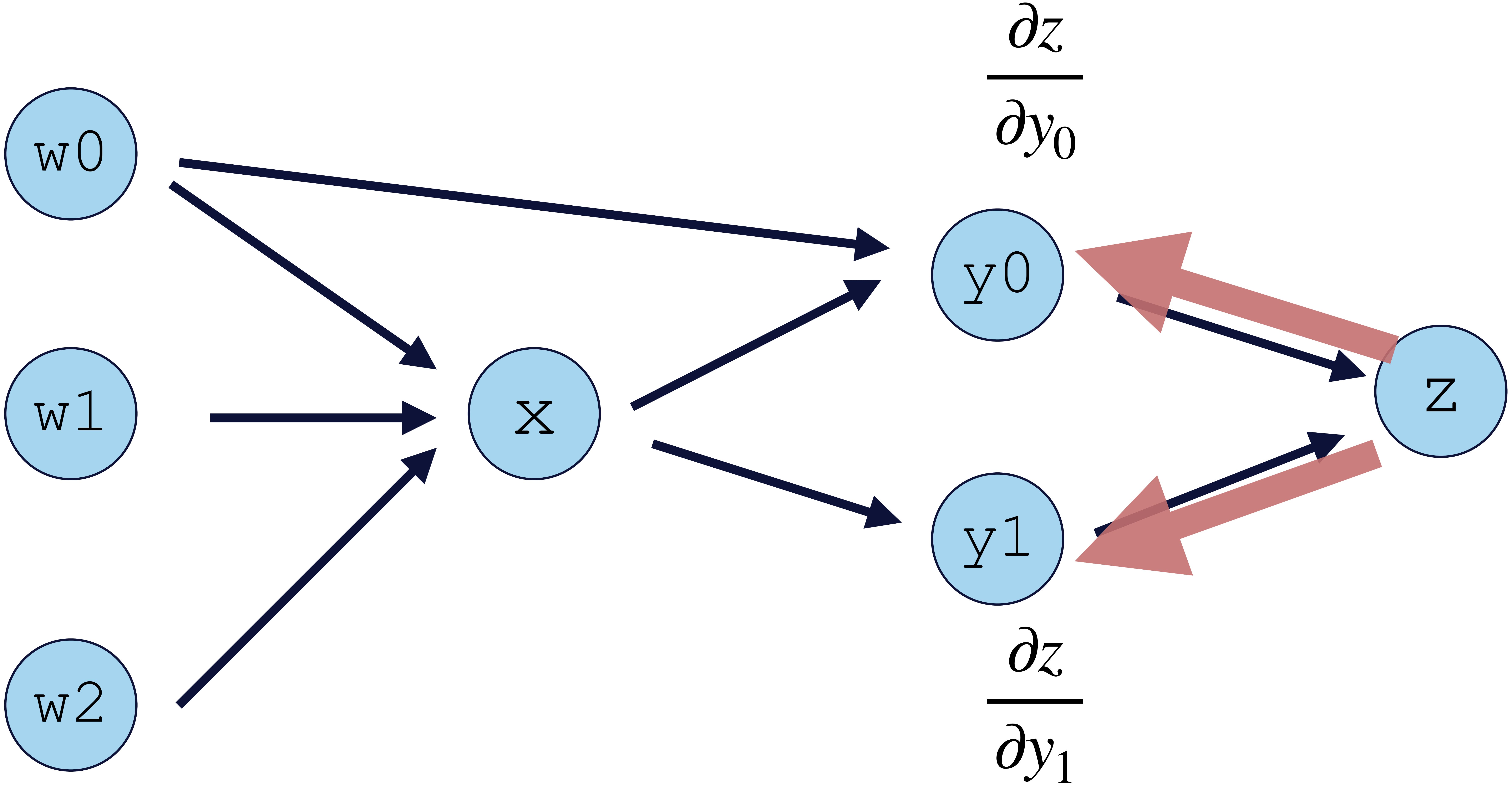
AUTODIFF = A PATH FINDING PROBLEM



AUTODIFF = A PATH FINDING PROBLEM



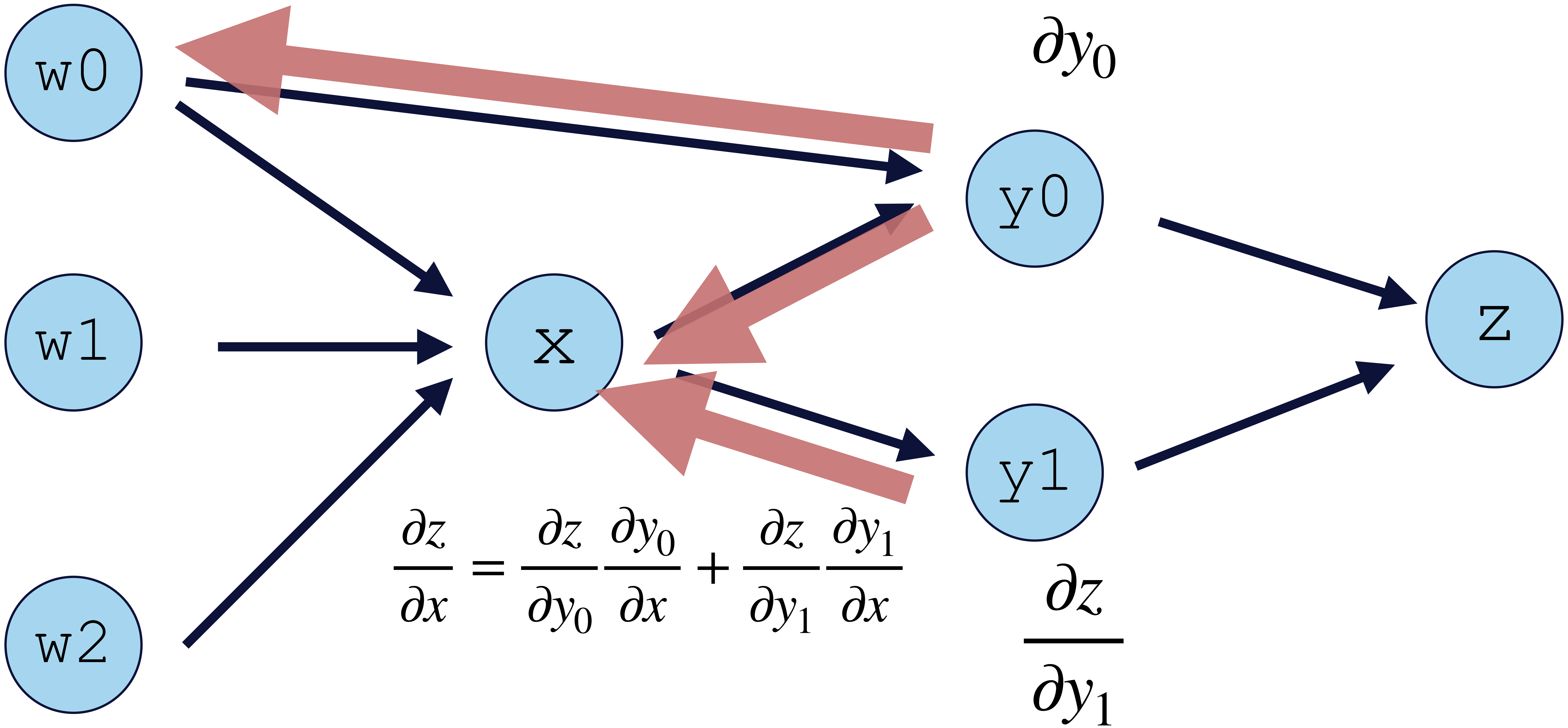
REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM



REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

$$\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0}\frac{\partial y_0}{\partial w_0}$$

$$\frac{\partial z}{\partial y_0}$$

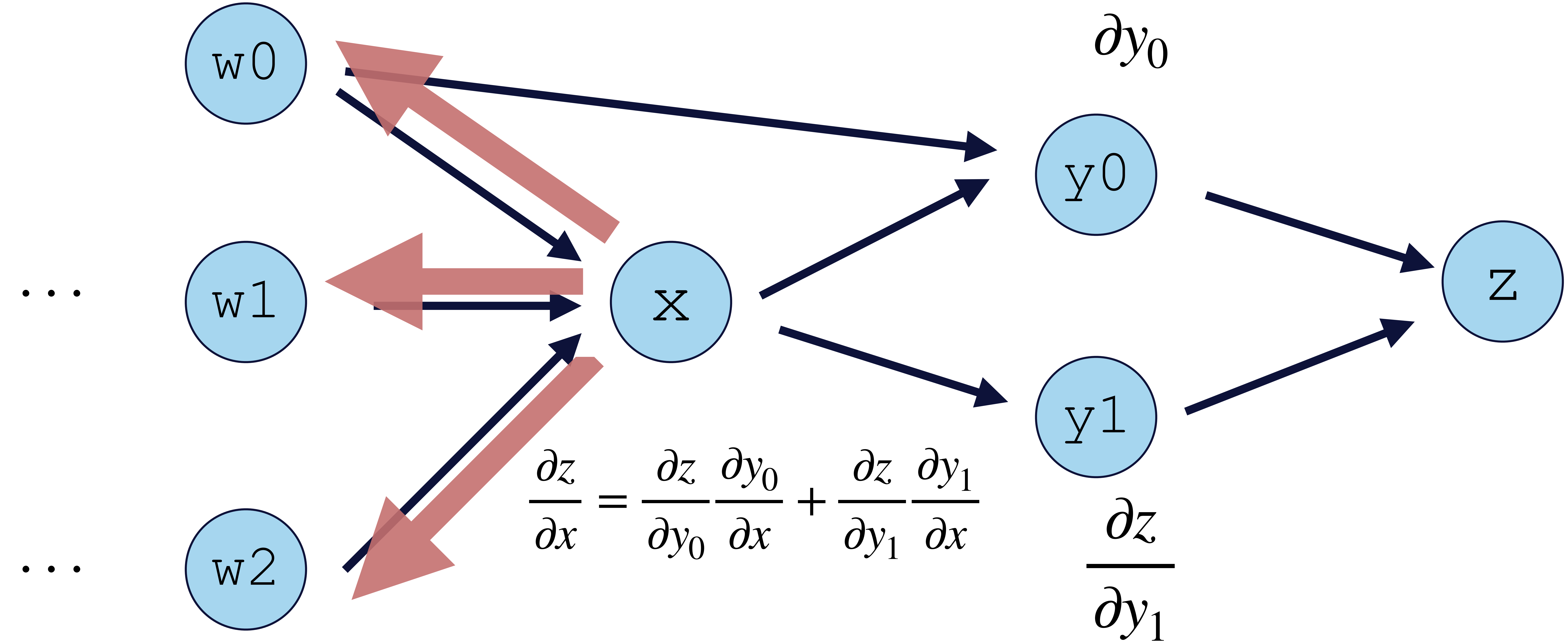


$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_0}\frac{\partial y_0}{\partial x} + \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}$$

$$\frac{\partial z}{\partial y_1}$$

REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

$$\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0}\frac{\partial y_0}{\partial w_0} + \frac{\partial z}{\partial x}\frac{\partial x}{\partial w_0}$$



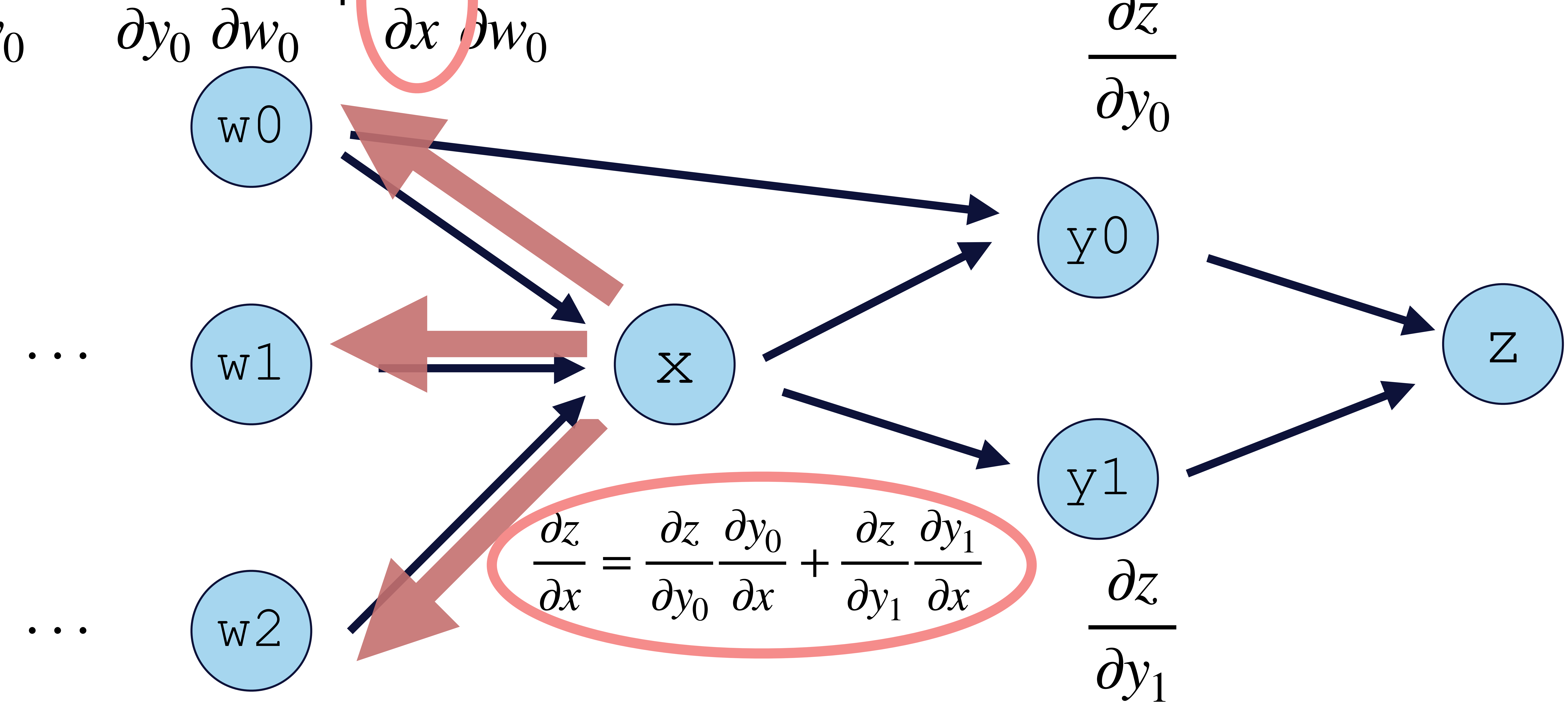
$$\frac{\partial z}{\partial y_0}$$

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_0}\frac{\partial y_0}{\partial x} + \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x}$$

$$\frac{\partial z}{\partial y_1}$$

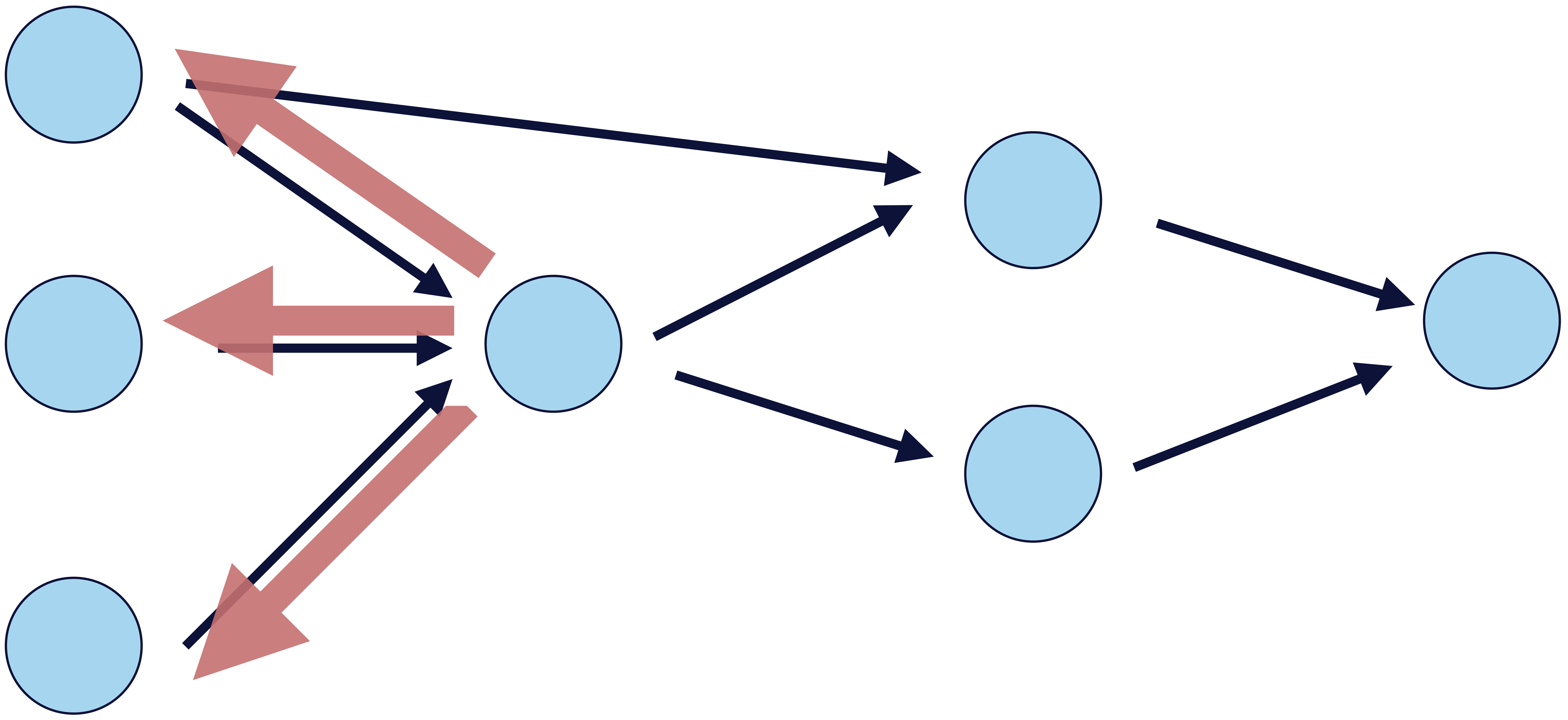
REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

$$\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0}\frac{\partial y_0}{\partial w_0} + \frac{\partial z}{\partial x}\frac{\partial x}{\partial w_0}$$



REVERSE-MODE AUTOMATIC DIFFERENTIATION PRODUCES EFFICIENT GRADIENTS

- gradient complexity: number of edges * constant
  - same as directly computing the function ("*cheap gradient principle*")



TRANSFORMING LOOPS WITH REVERSE MODE

- remember every intermediate values in the forward pass, then run the loop backward
  - also works for recursion
  - unbounded memory usage

```
function f(x):
  result = x
  for i = 1 to 8:
    result = exp(result)
  return result
```

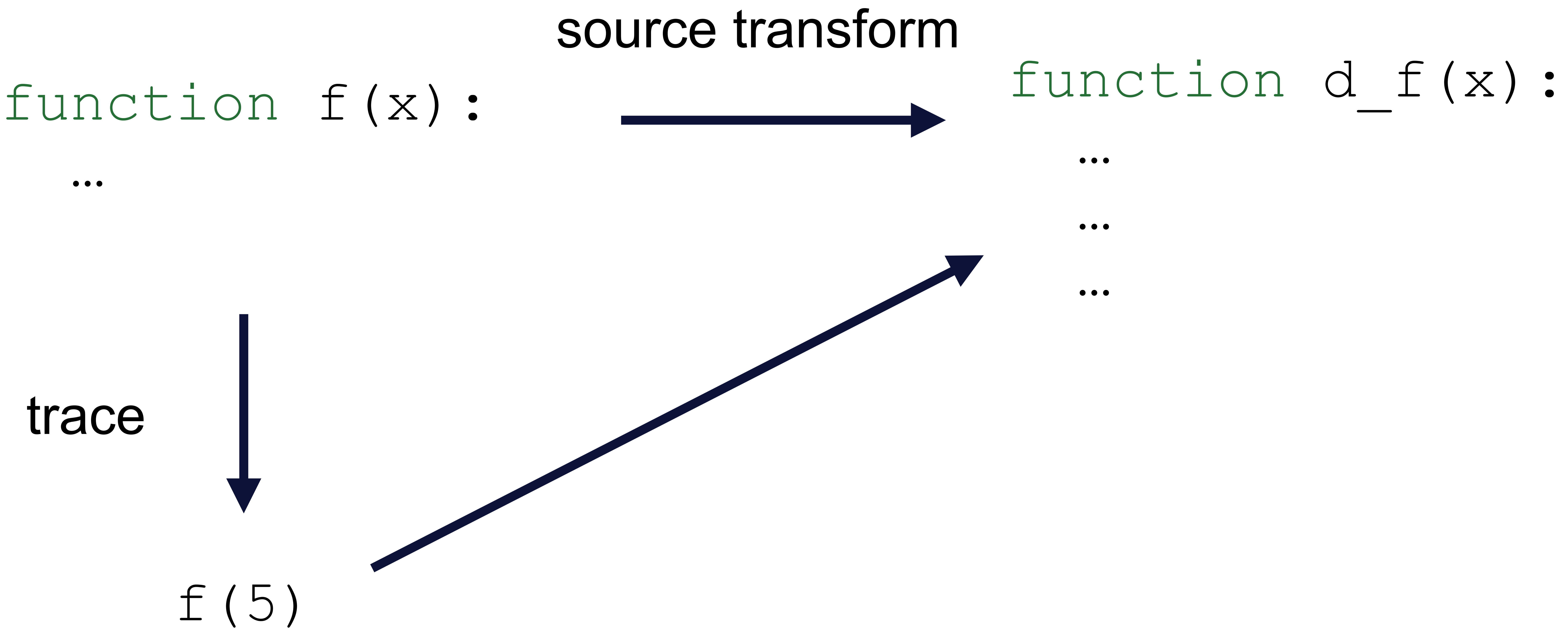
→

```
function d_f(x):
  result = x
  results = []
  for i = 1 to 8:
    results.push(result)
    result = exp(result)

  for i = 8 to 1:
    d_results = d_result *
      exp(results[i])
  return result
```

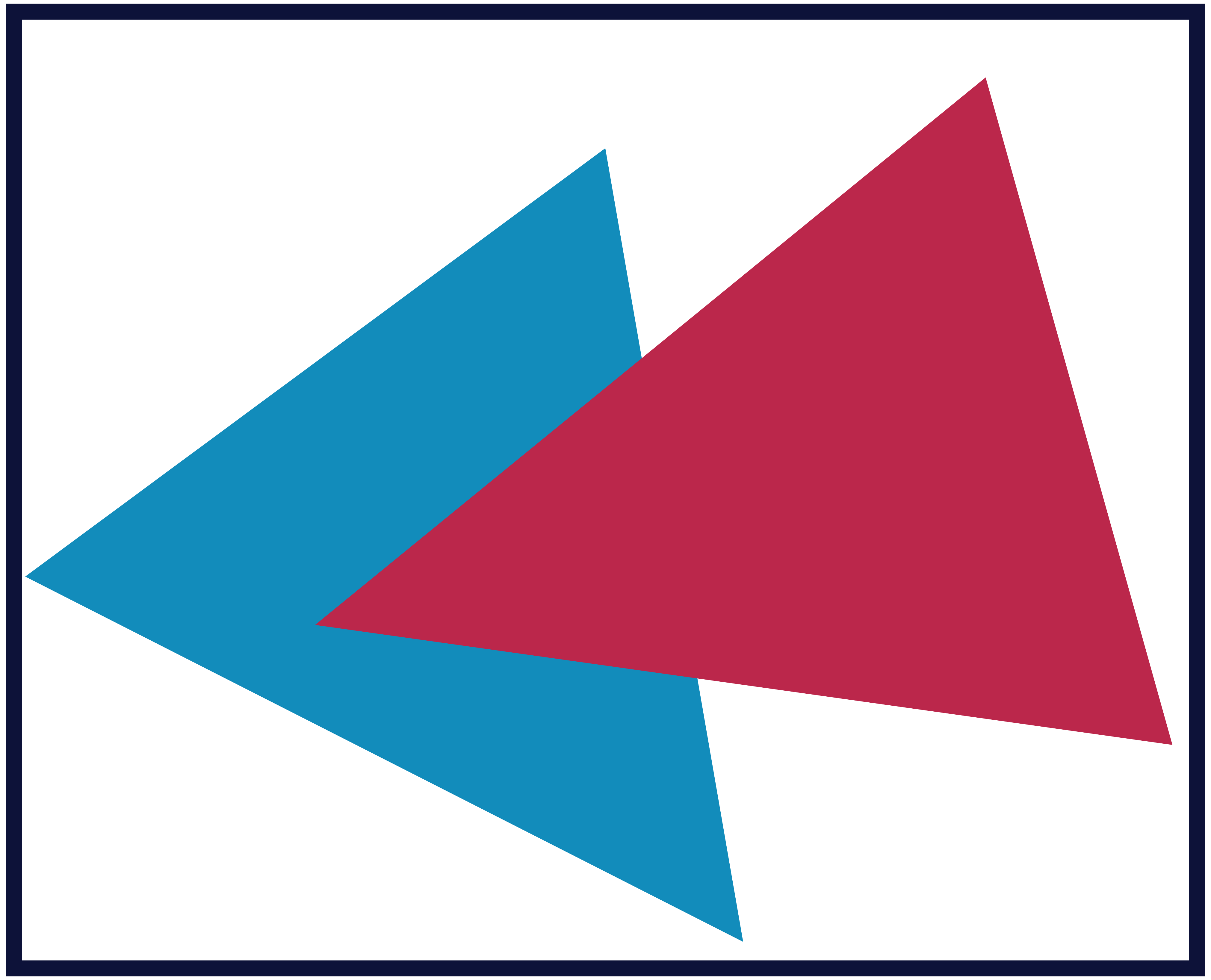
SOURCE TRANSFORM V.S. TAPING

- a spectrum: how much is done at compile time
  - similar to (tracing) JIT v.s. static compile



DIFFERENTIATING CONDITIONALS

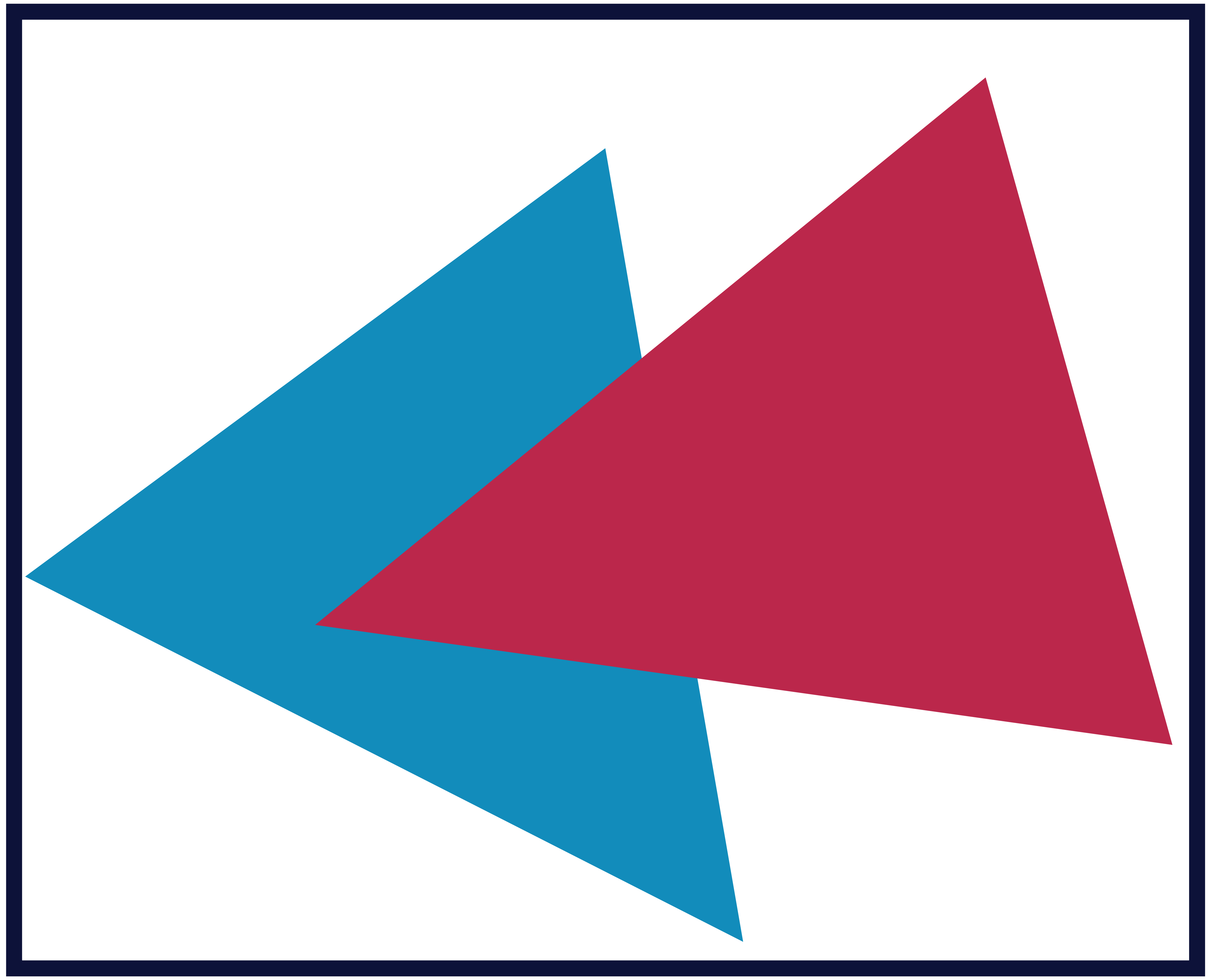
```
if (hit the red triangle)
  return red
elif (hit the blue triangle)
  return blue
else
  return white
```



DIFFERENTIATING CONDITIONALS

```
if (hit the red triangle)
  return red
elif (hit the blue triangle)
  return blue
else
  return white
```

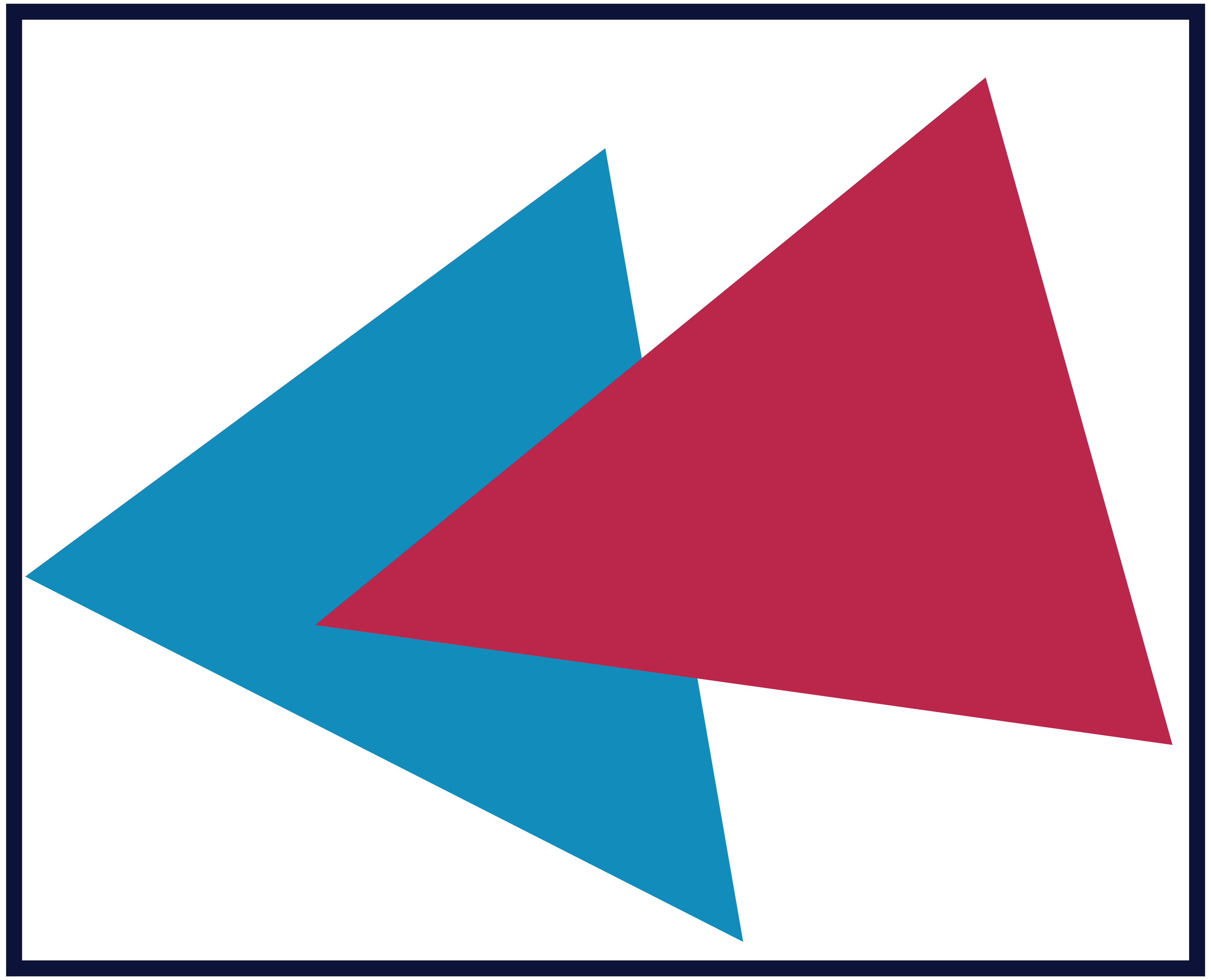
- derivative of color w.r.t. triangle vertex is 0



DIFFERENTIATING CONDITIONALS

```
if (hit the red triangle)
  return red
elif (hit the blue triangle)
  return blue
else
  return white
```

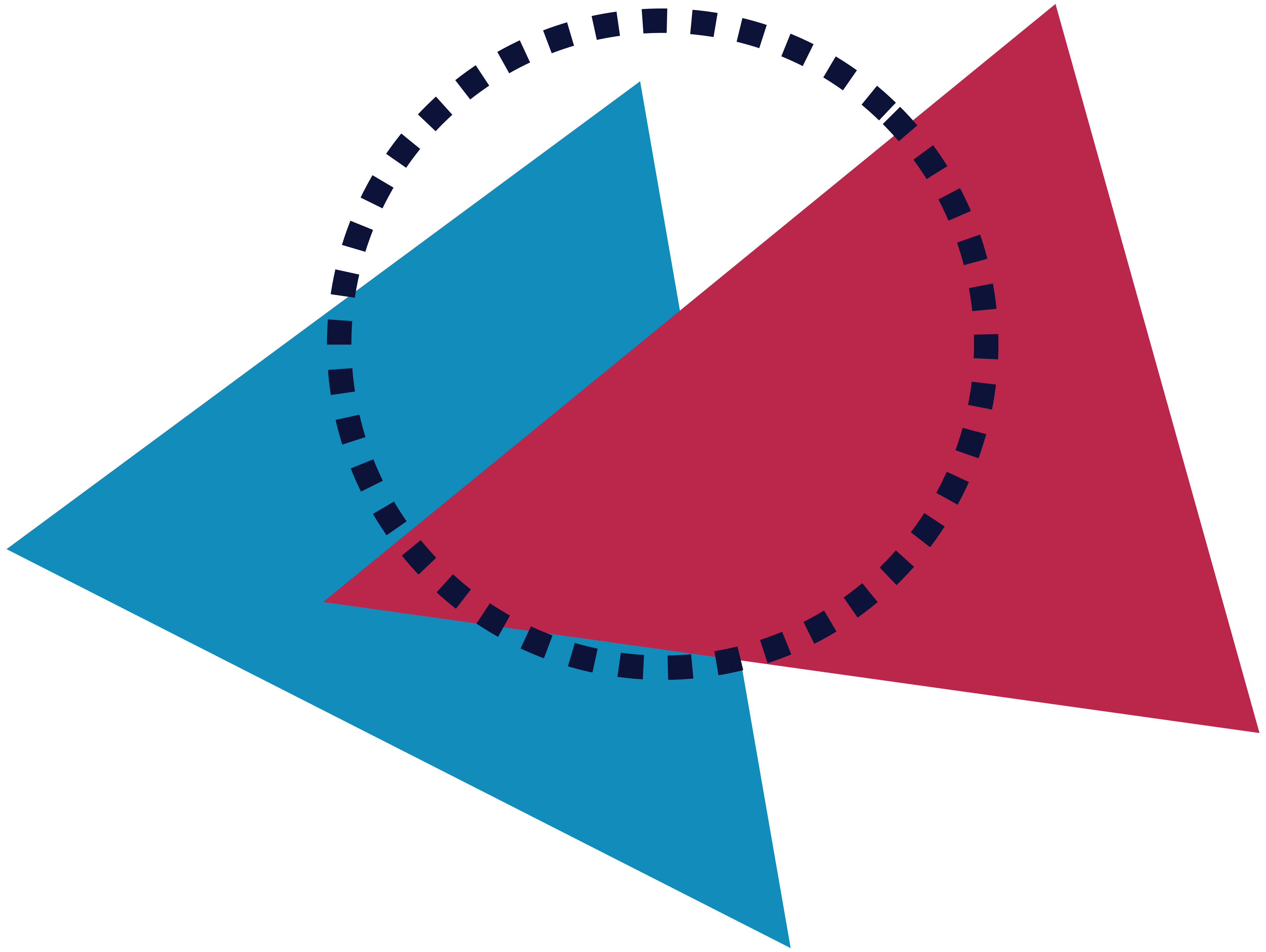
- derivative of color w.r.t. triangle vertex is 0
  - or is it?



RENDERING = COMPUTING INTEGRALS

- pixel color is defined by the average color over an area
  - aka anti-aliasing

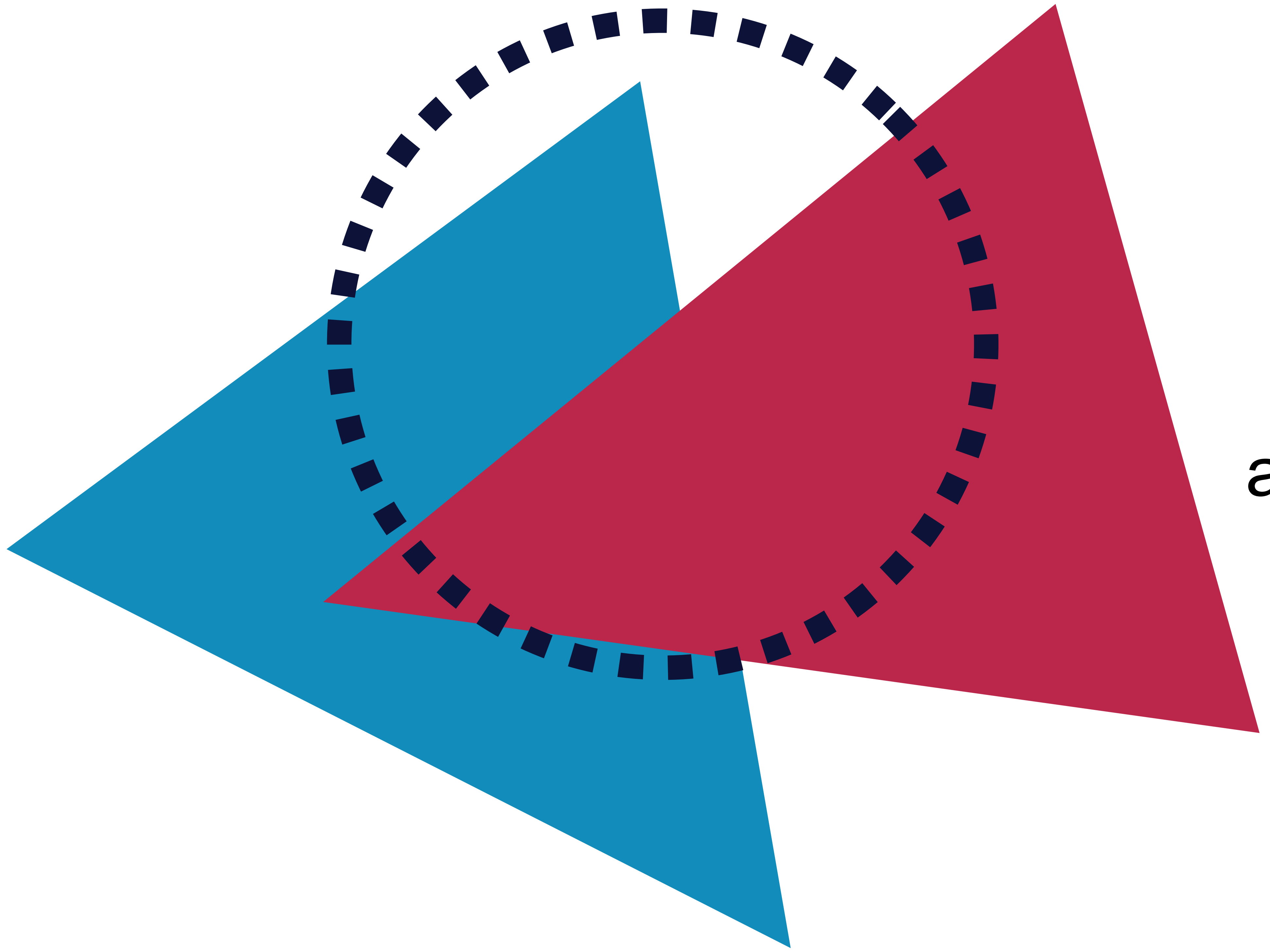
pixel filter support



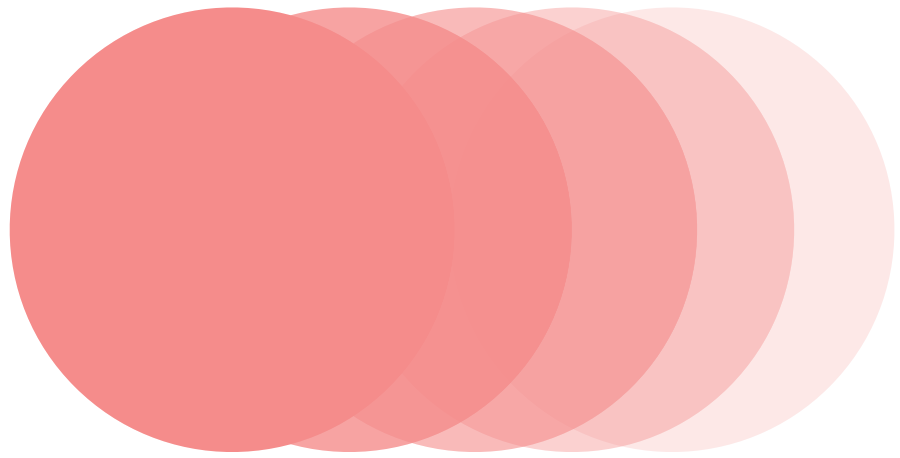
RENDERING = COMPUTING INTEGRALS

- pixel color is defined by the average color over an area
  - aka anti-aliasing

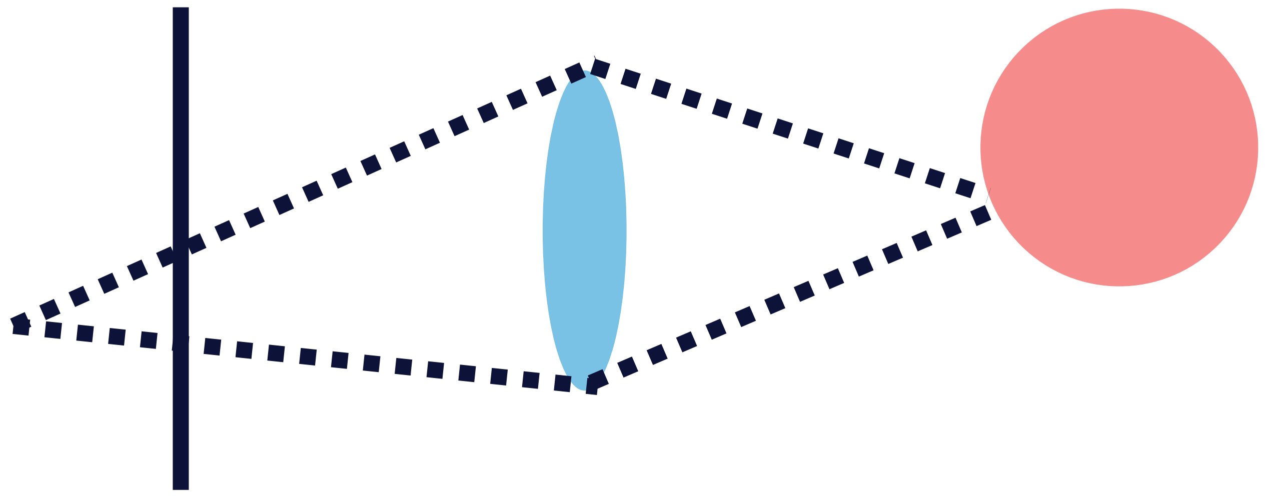
pixel filter support



shutter time
(motion blur)

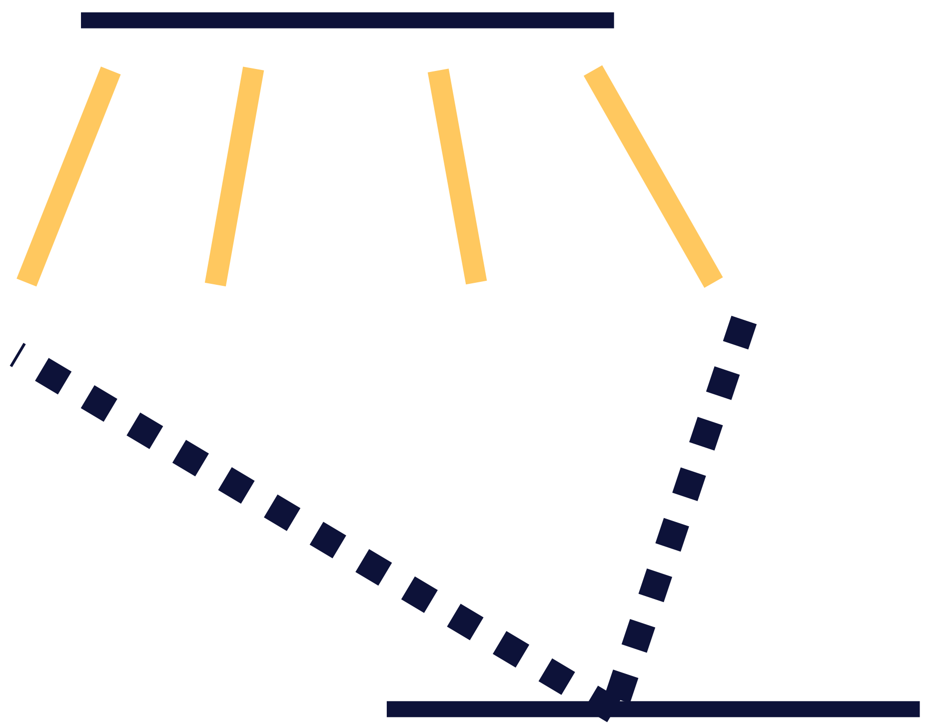


camera aperture
(defocus blur)

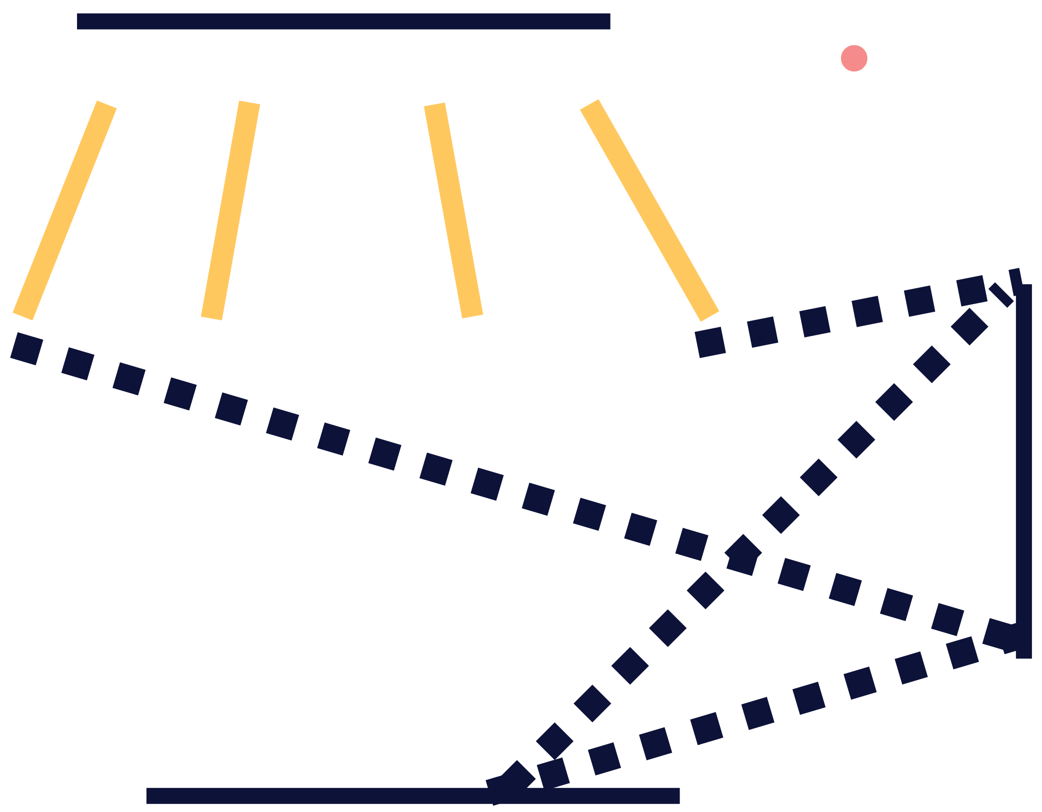


- wavelength
- participating media
- …
- and more!

area light

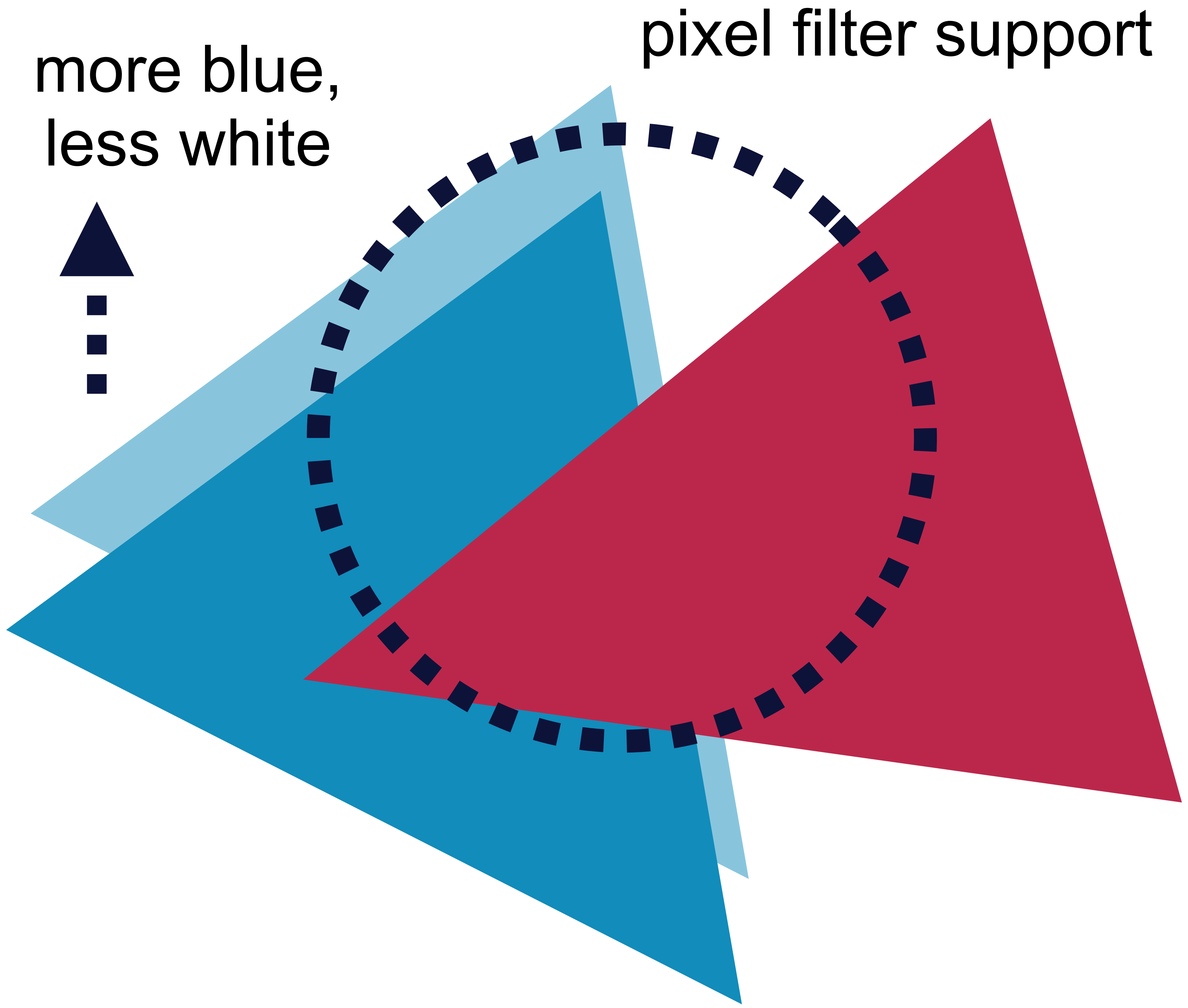


global illumination



THE RENDERING INTEGRALS ARE DIFFERENTIABLE!

- **While the *integrand* is discontinuous, the *integral* is differentiable!**
  - the average color changes continuously as triangles move

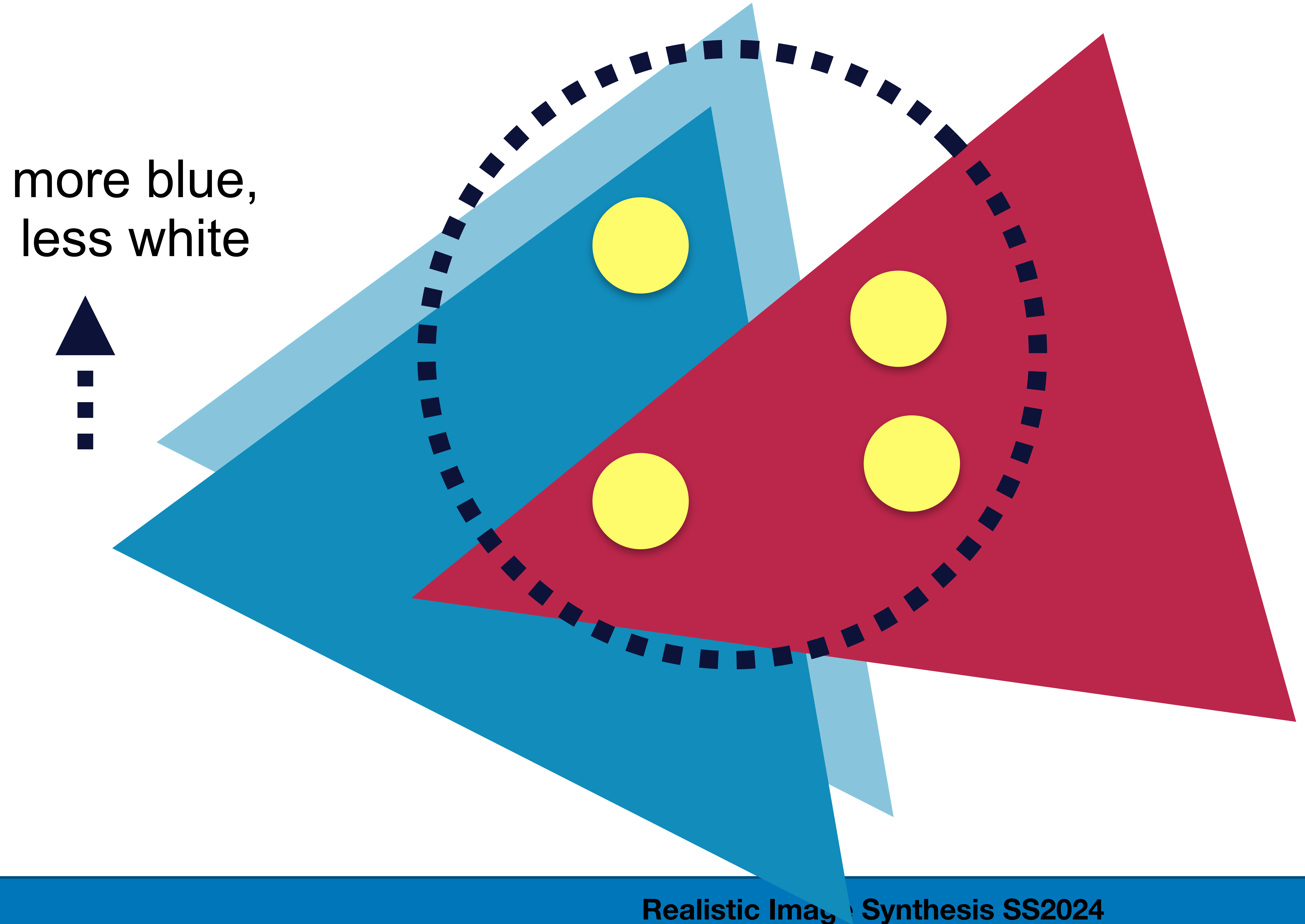


$$\int$$

```
if (hit the red triangle)
  return red
elif (hit the blue triangle)
  return blue
else
  return white
```

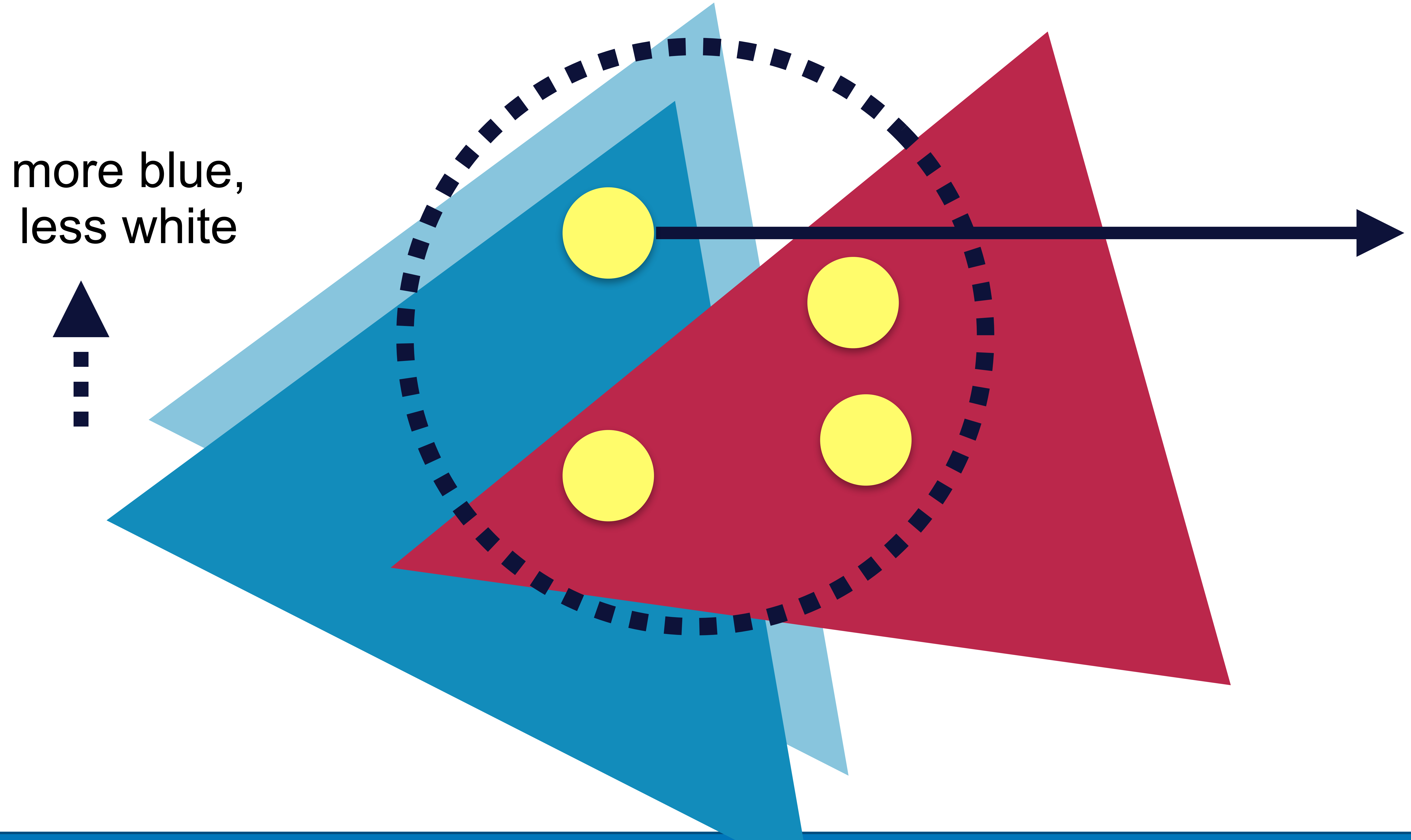
RENDERING = SAMPLING INTEGRALS

- We evaluate these integrals by sampling them



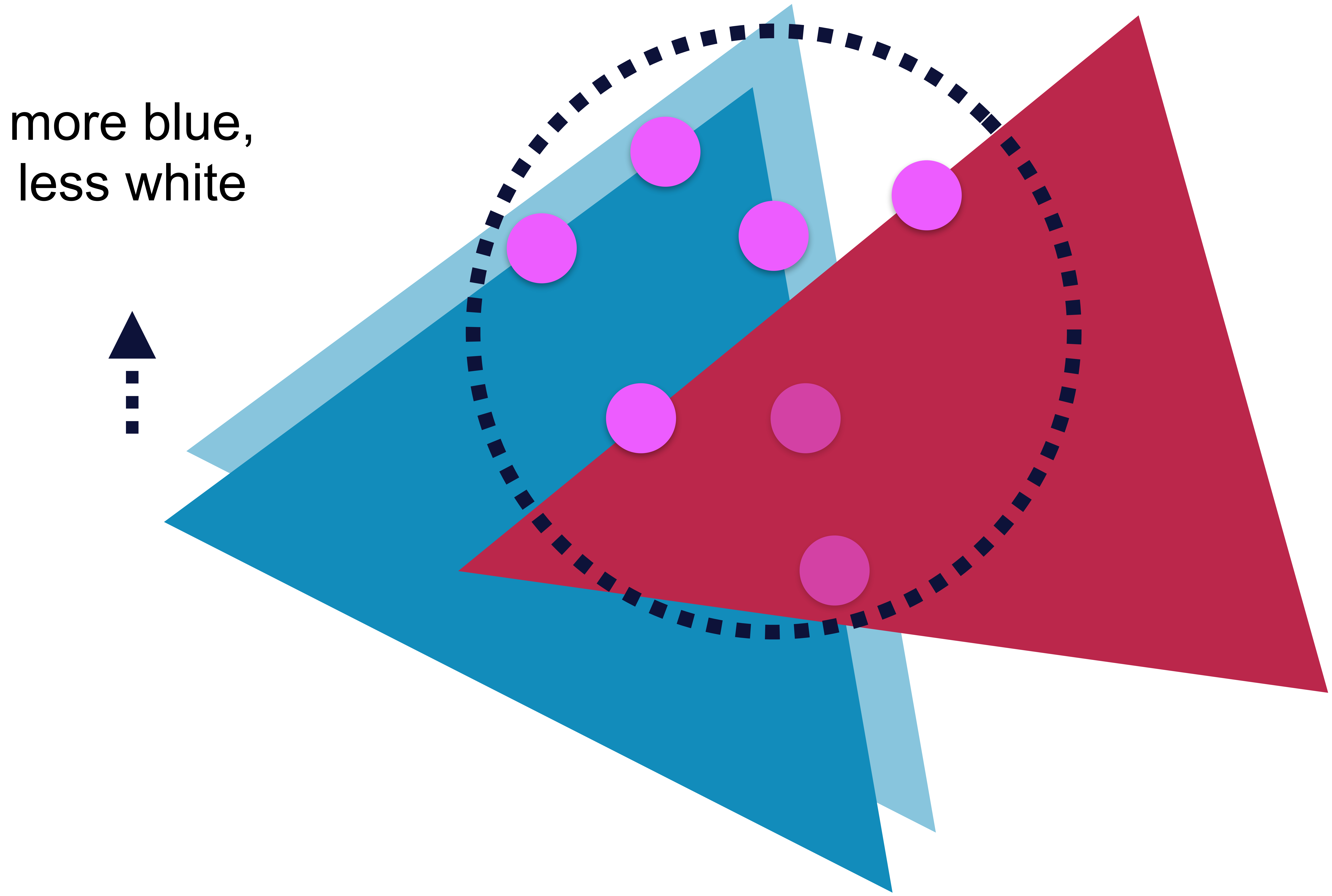
$$\int \ldots \approx \sum \ldots$$

DIFFERENTIATING INTEGRAL SAMPLES GIVES WRONG DERIVATIVES



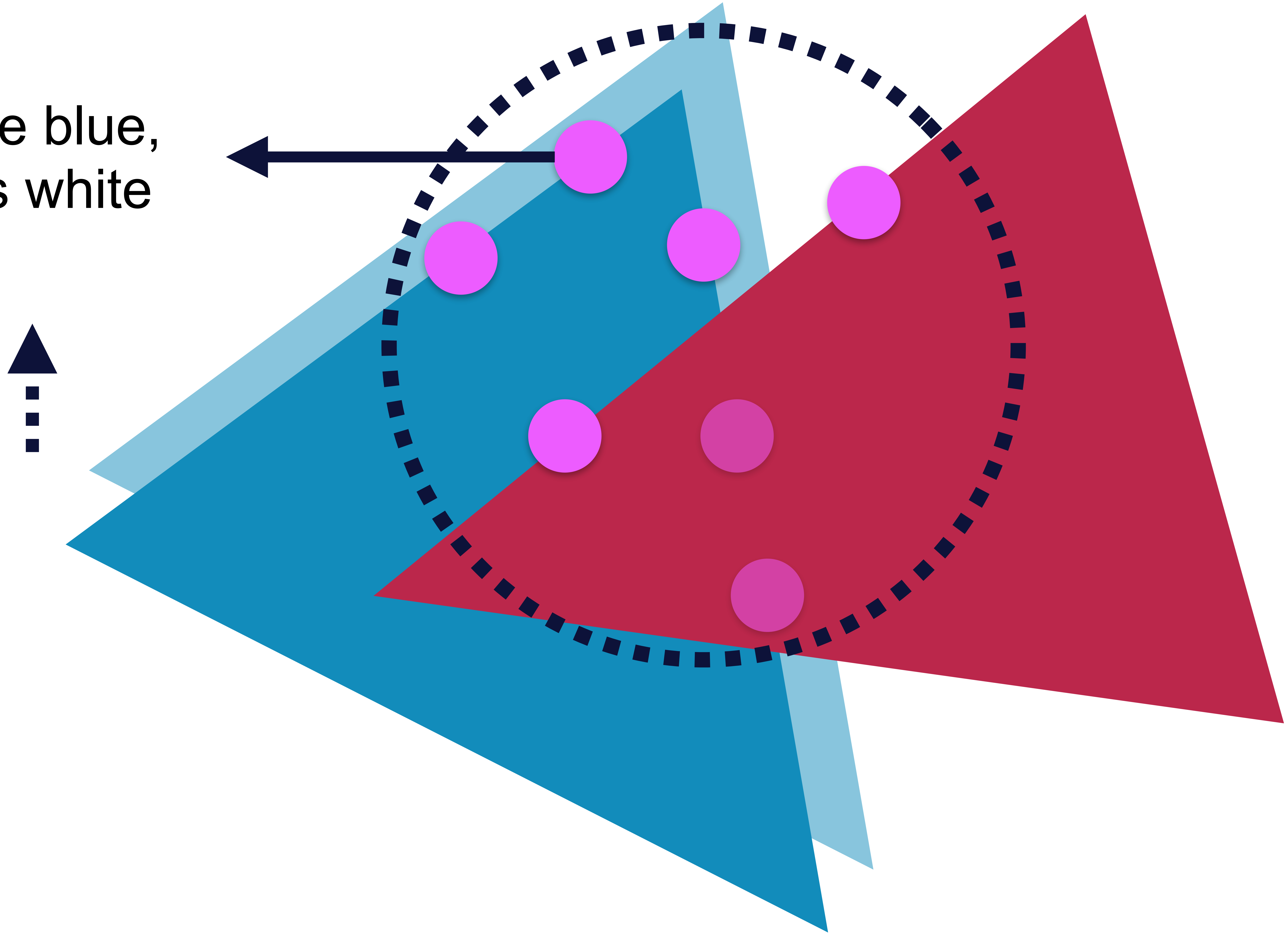
$$\frac{\partial}{\partial p} = 0$$

KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES



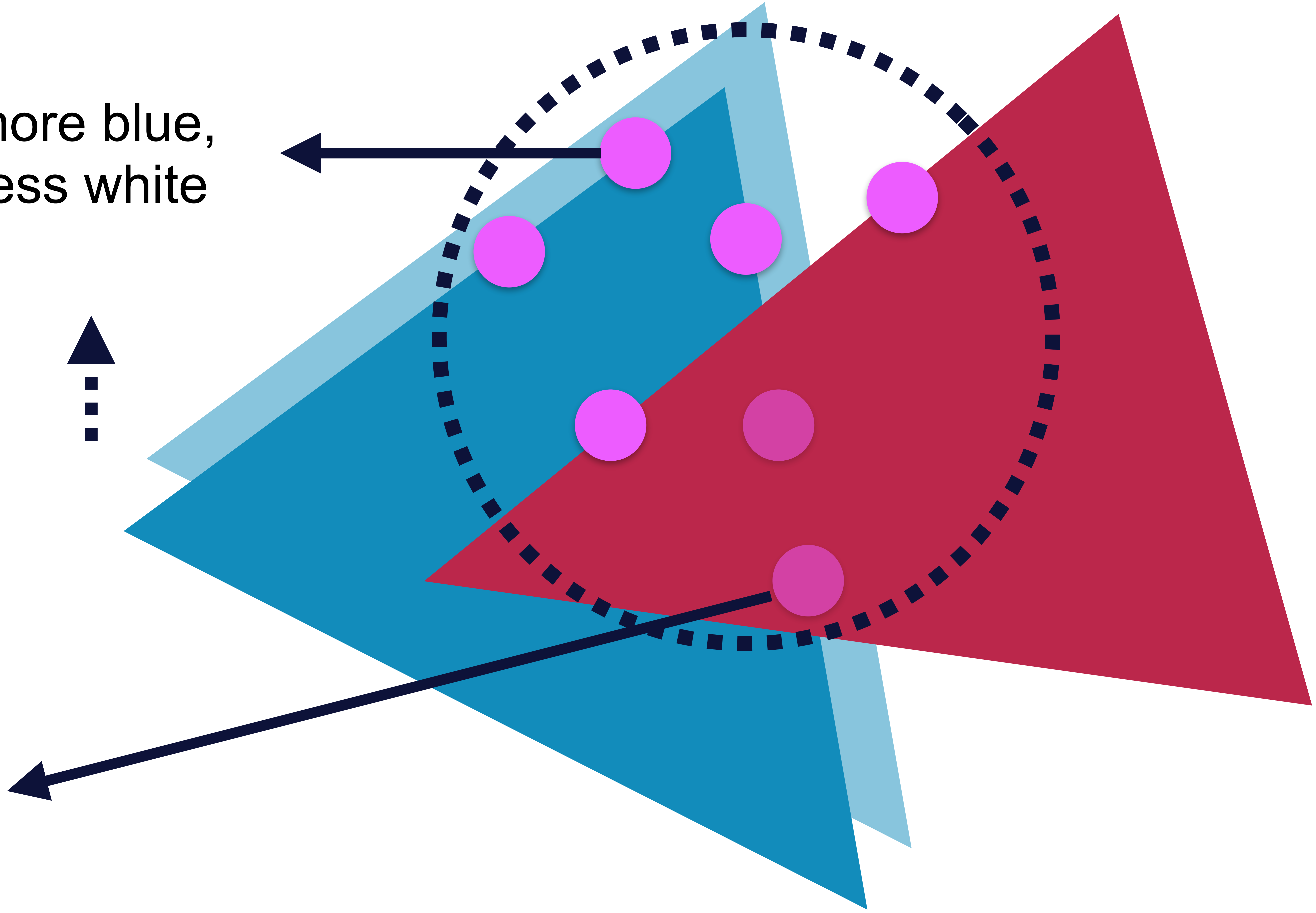
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

$$\frac{\partial}{\partial p} = \text{more blue, less white}$$



KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

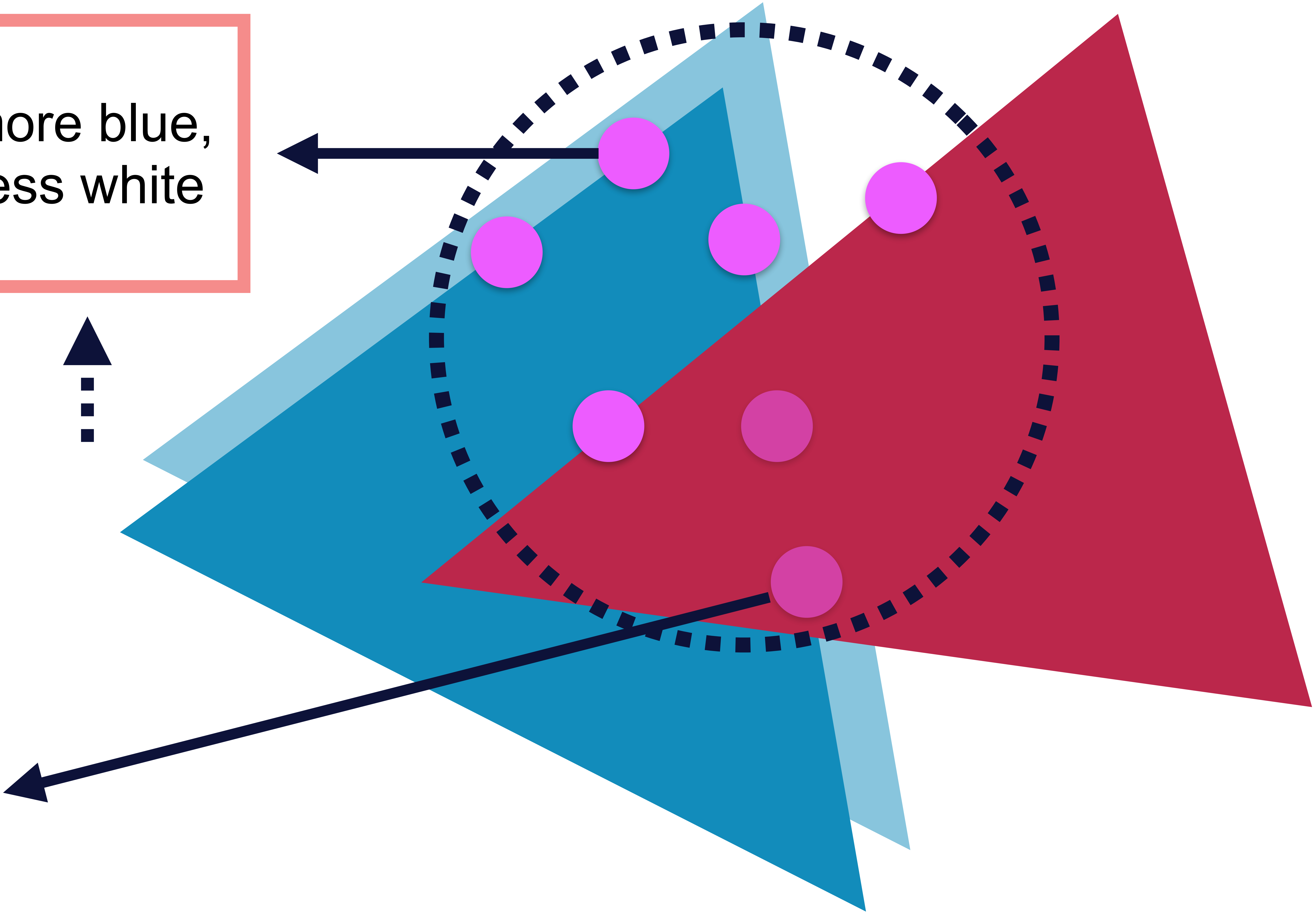
$$\frac{\partial}{\partial p} = \text{more blue, less white}$$



$$\frac{\partial}{\partial p} = 0$$

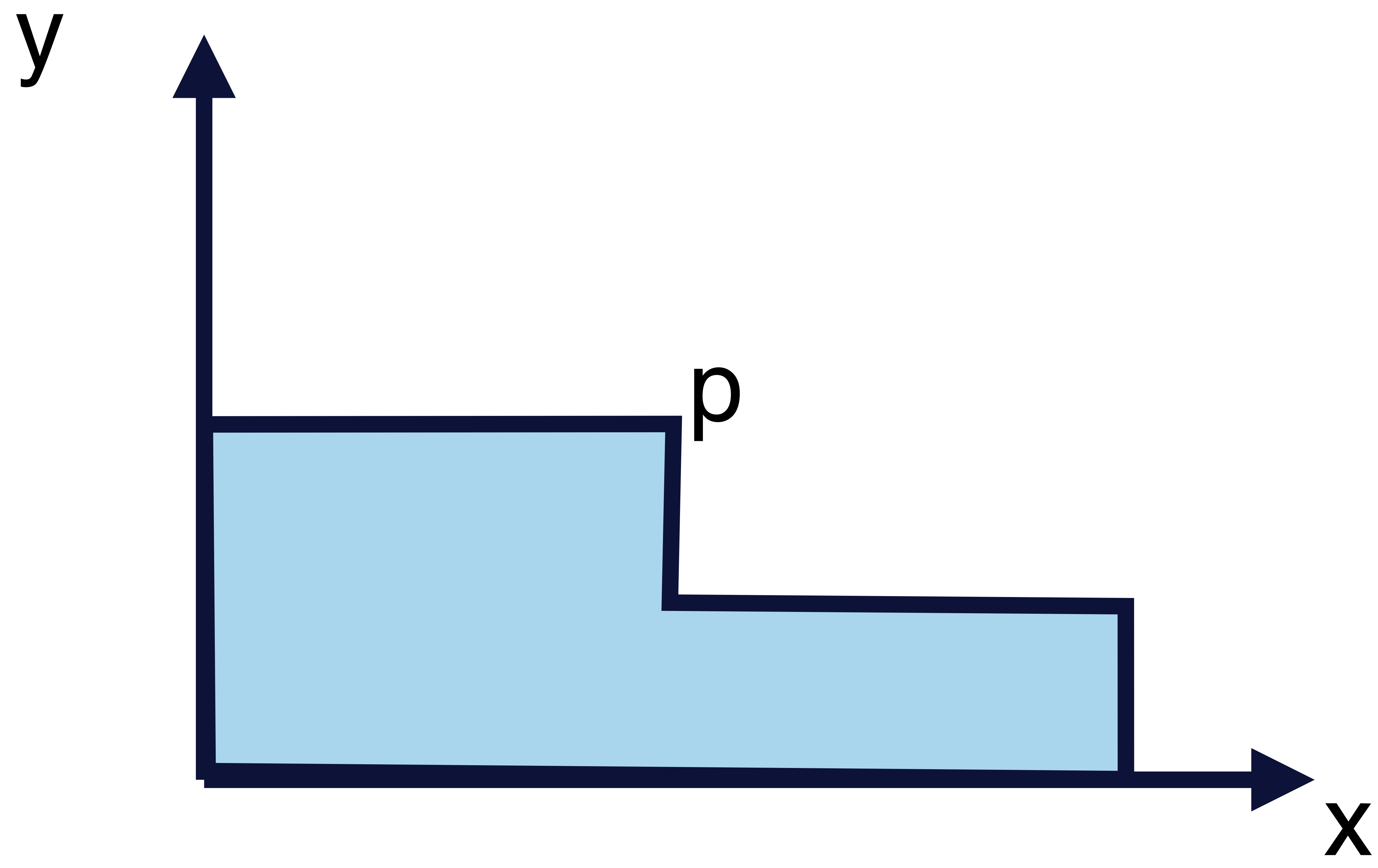
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

$$\frac{\partial}{\partial p} = $$ more blue, less white



$$\frac{\partial}{\partial p} = 0$$

LET'S DERIVE THE DERIVATIVES IN 1D

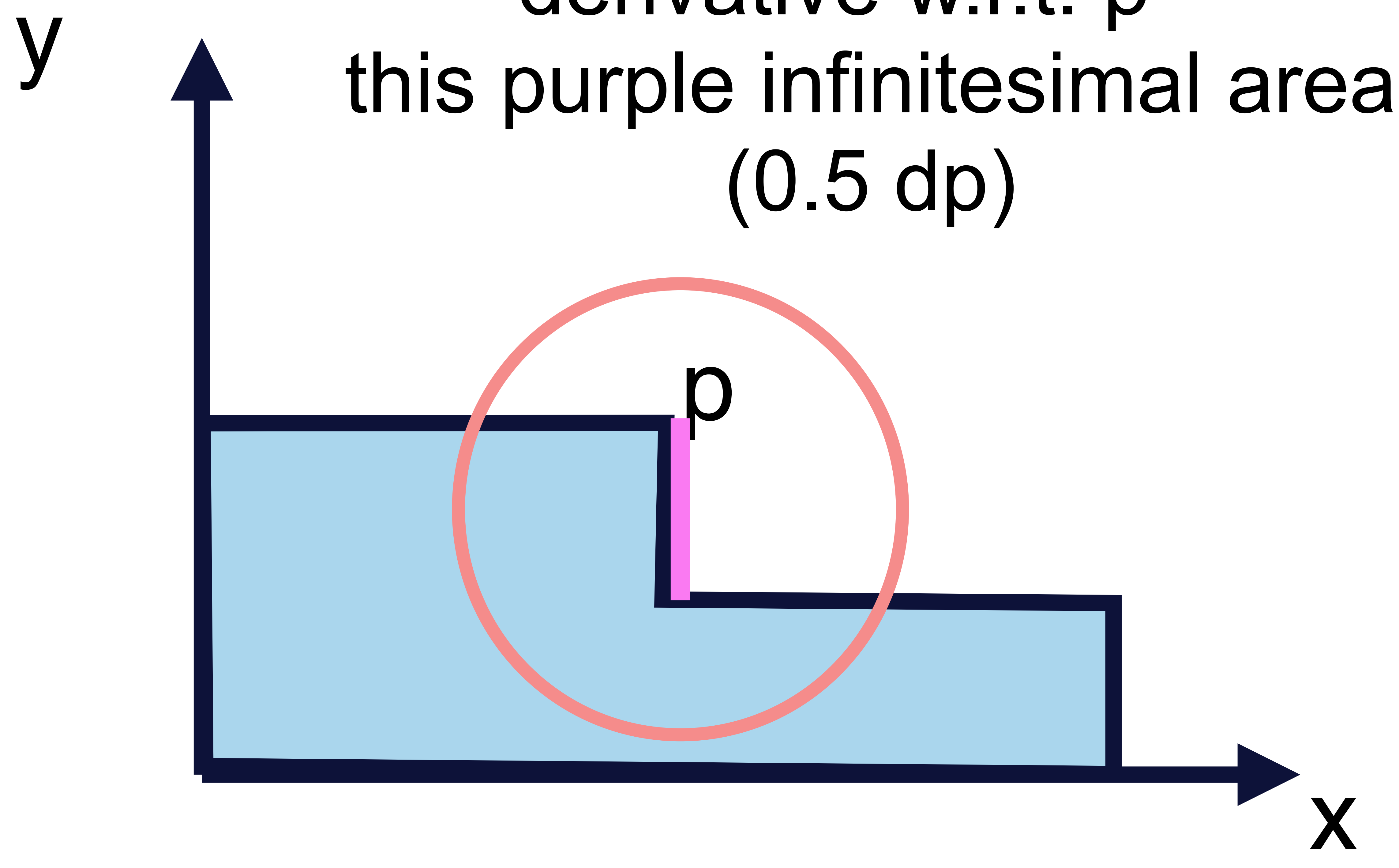


(the blue area)

$$\int_{x=0}^{x=1} x < p \; ? \; 1 : 0.5$$

LET'S DERIVE THE DERIVATIVES IN 1D

derivative w.r.t. p =
this purple infinitesimal area
(0.5 dp)

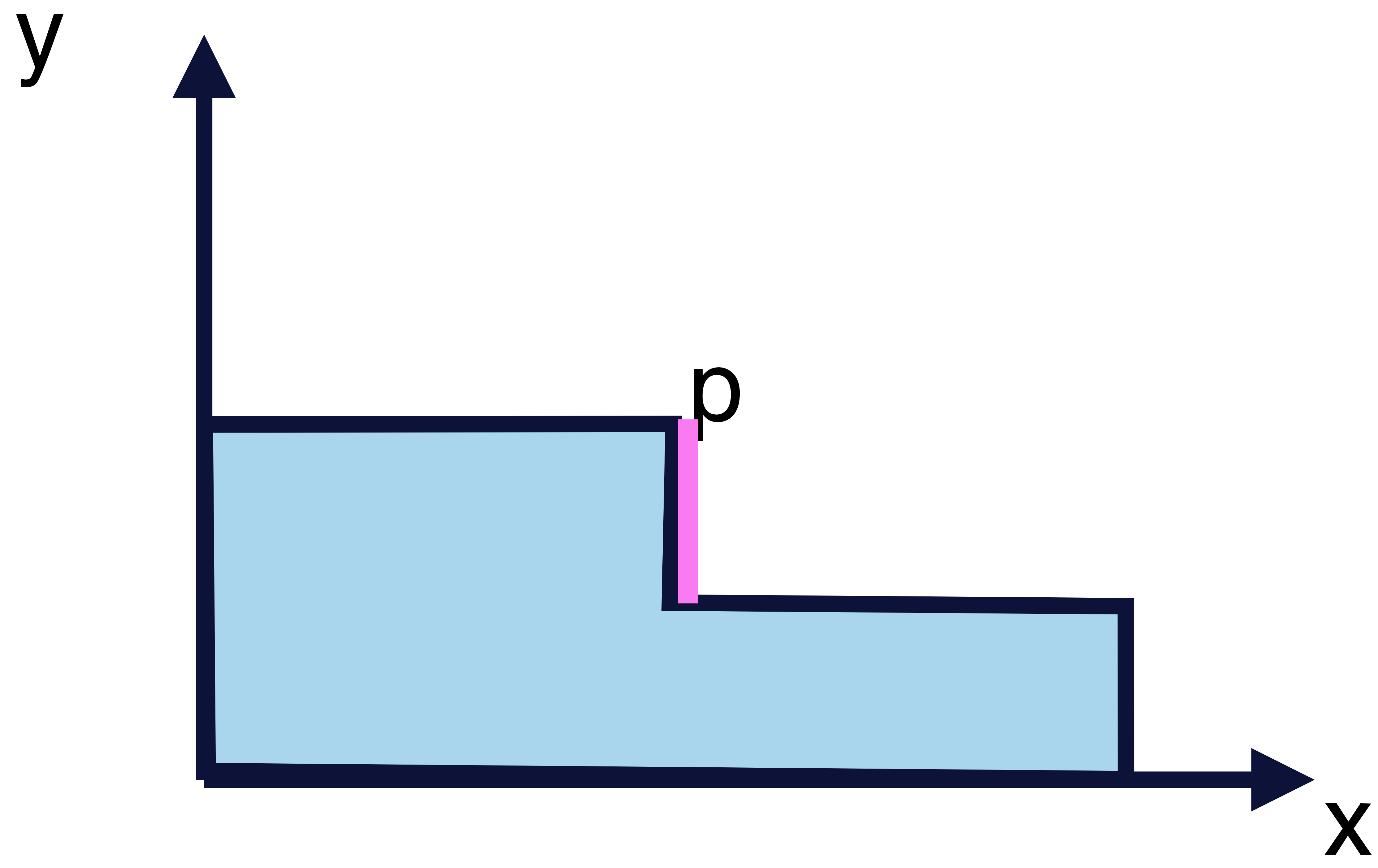


(the blue area)

$$\int_{x=0}^{x=1} \quad x < p \; ? \; 1 : 0.5$$

LET'S DERIVE THE DERIVATIVES IN 1D

- Trick: move the discontinuities to the integral boundaries



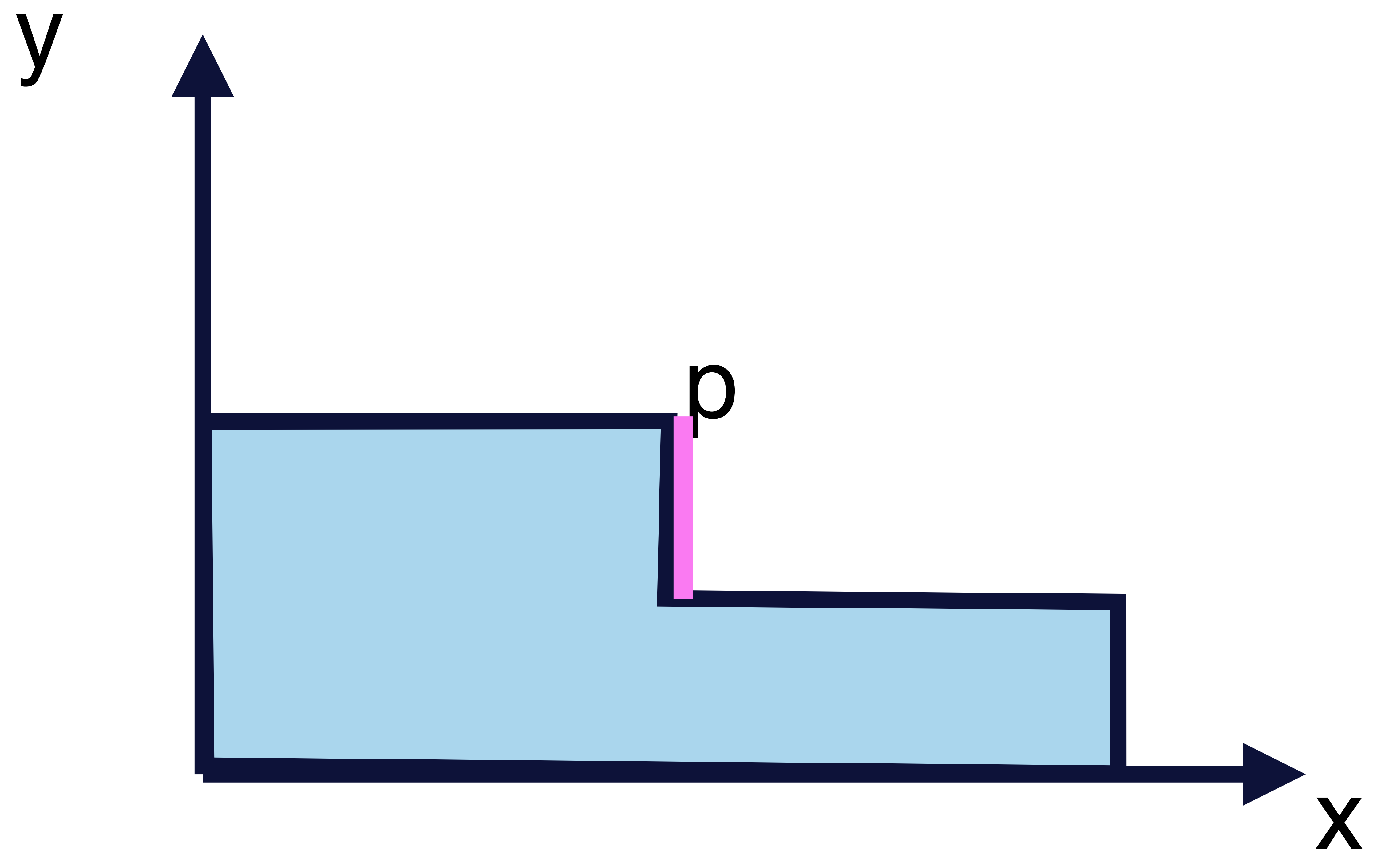
(the blue area)

$$\int_{x=0}^{x=1} x < p \ ? \ 1 : 0.5$$

$$= \int_{x=0}^{x=p} 1 + \int_{x=p}^{x=1} 0.5$$

LET'S DERIVE THE DERIVATIVES IN 1D

- Trick: move the discontinuities to the integral boundaries



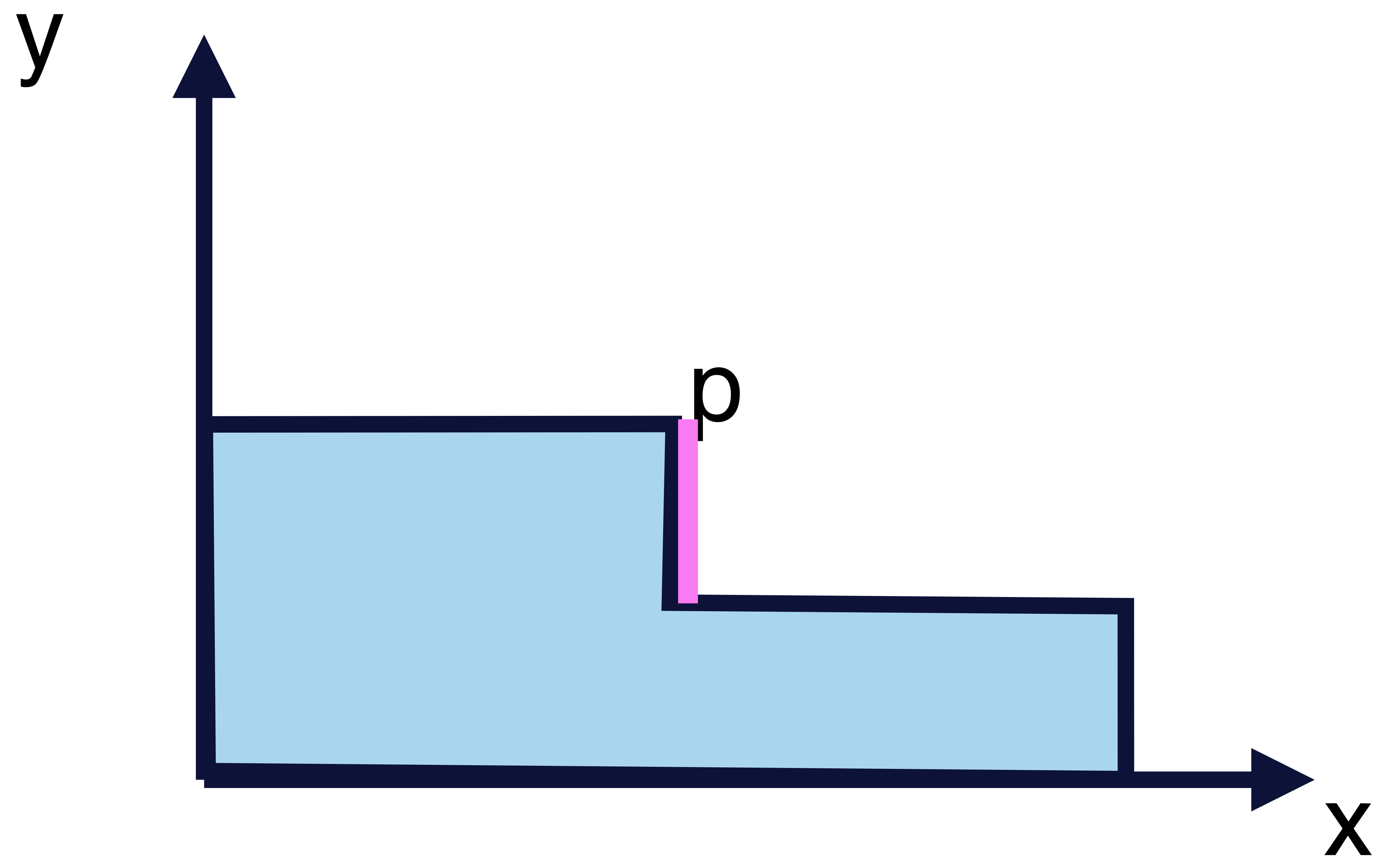
(the blue area)

$$\int_{x=0}^{x=1} x < p \; ? \; 1 : 0.5$$

$$= \int_{x=0}^{x=p} 1 + \int_{x=p}^{x=1} 0.5$$

DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

$$\int_{x=0}^{x=1} \quad x < p \; ? \; 1 : 0.5$$

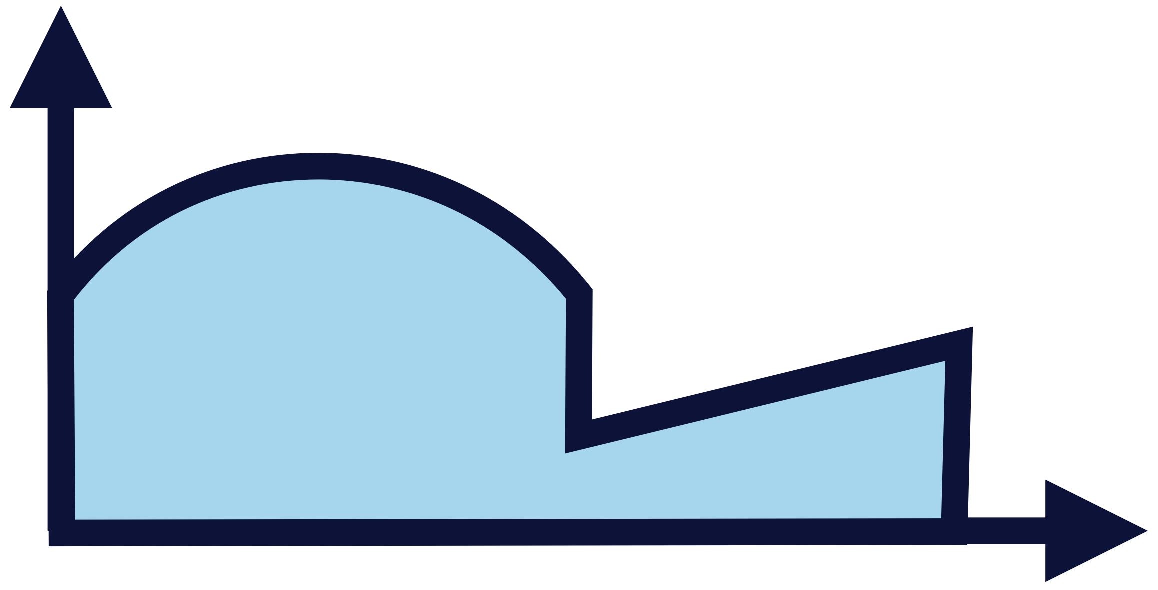
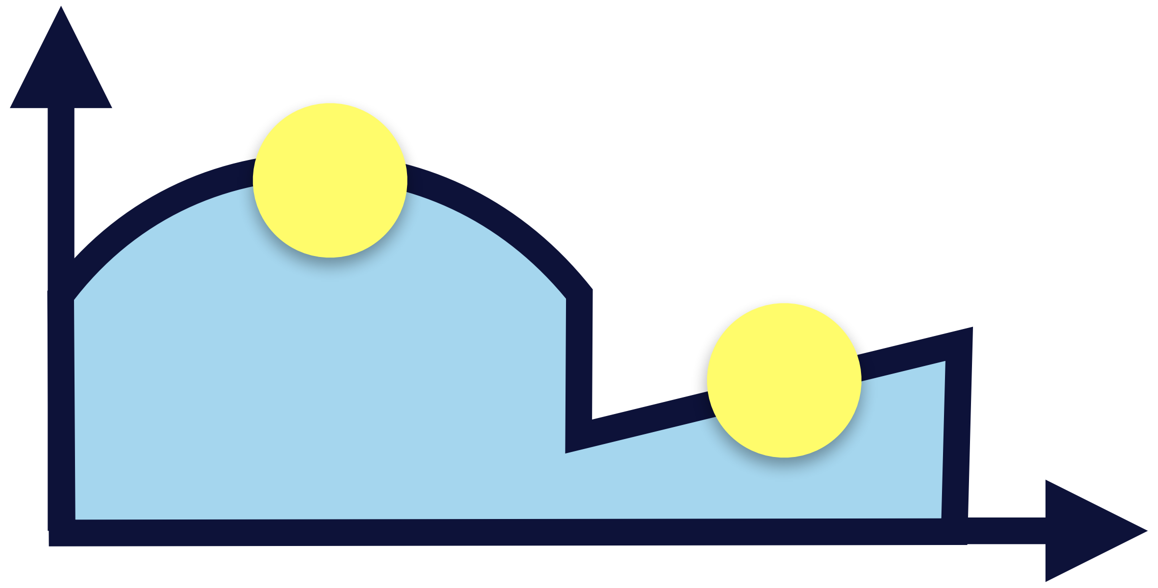


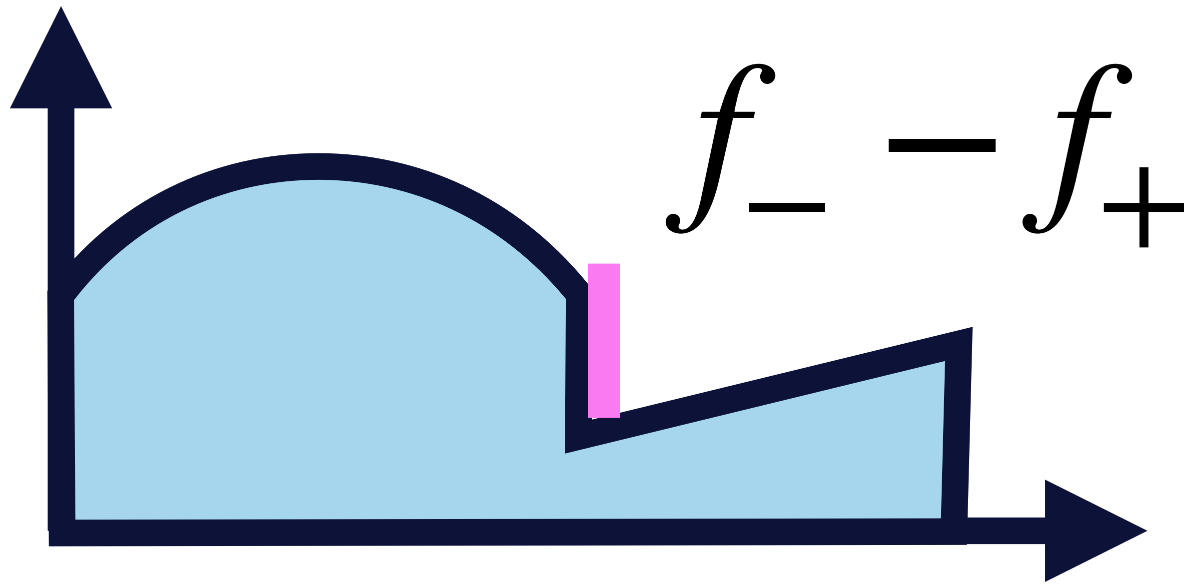
(derivative of blue area w.r.t. p)

$$\frac{\partial}{\partial p}\left(\int_{x=0}^{x=p} 1 + \int_{x=p}^{x=1} 0.5\right)$$

$$= 1 - 0.5$$

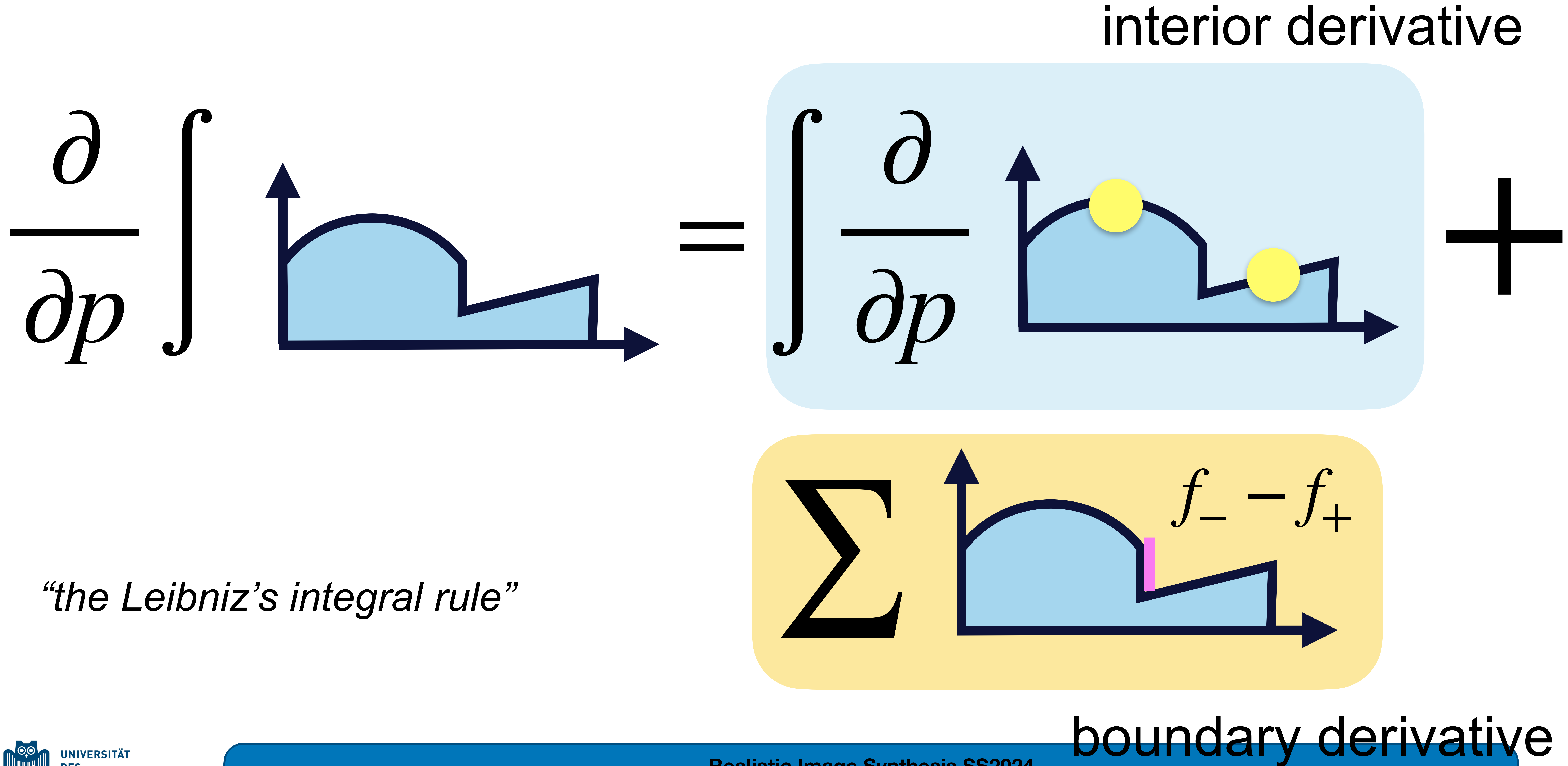
DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

$$\frac{\partial}{\partial p}\int$$  $$= \int \frac{\partial}{\partial p}$$  $$+$$

$$\sum$$  $$f_- - f_+$$

“the Leibniz’s integral rule”

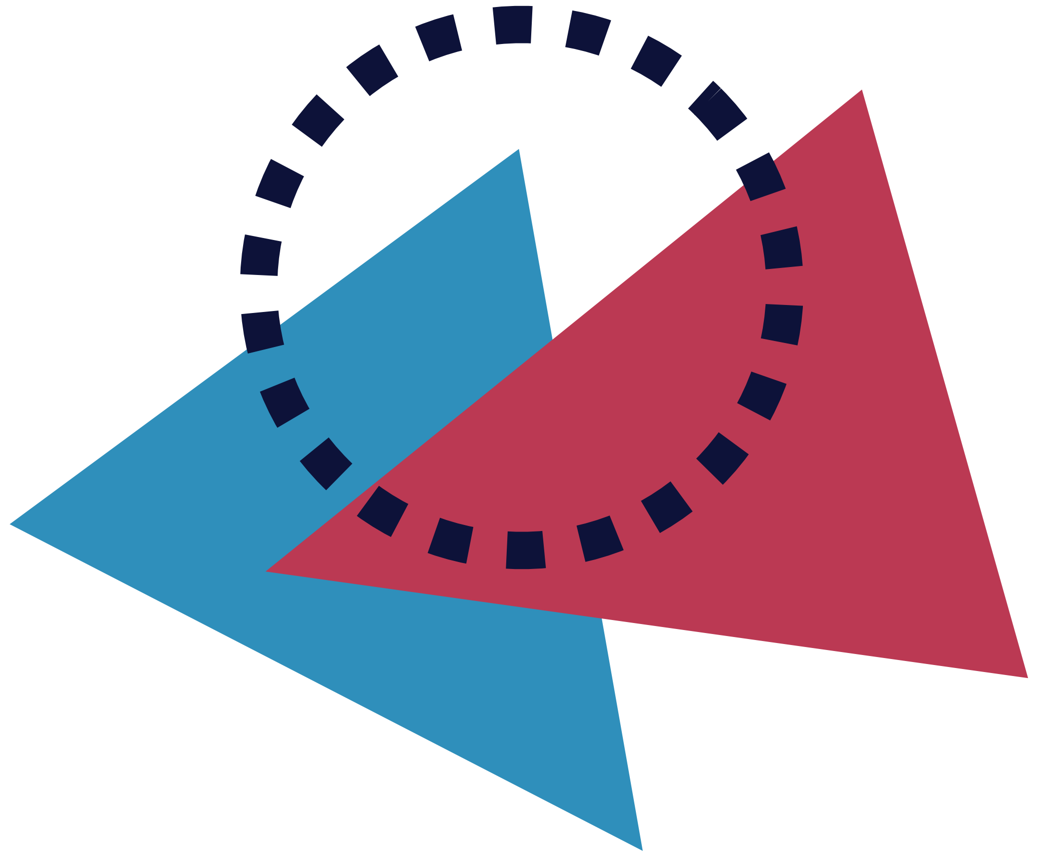
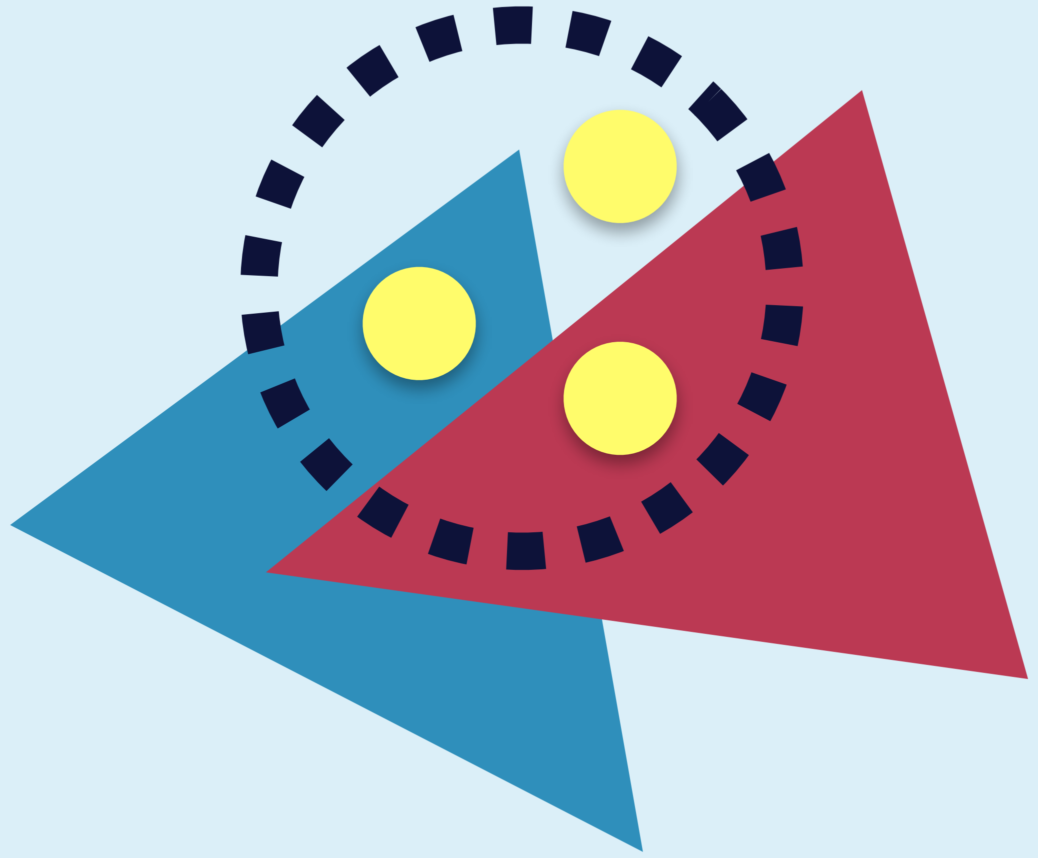
DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

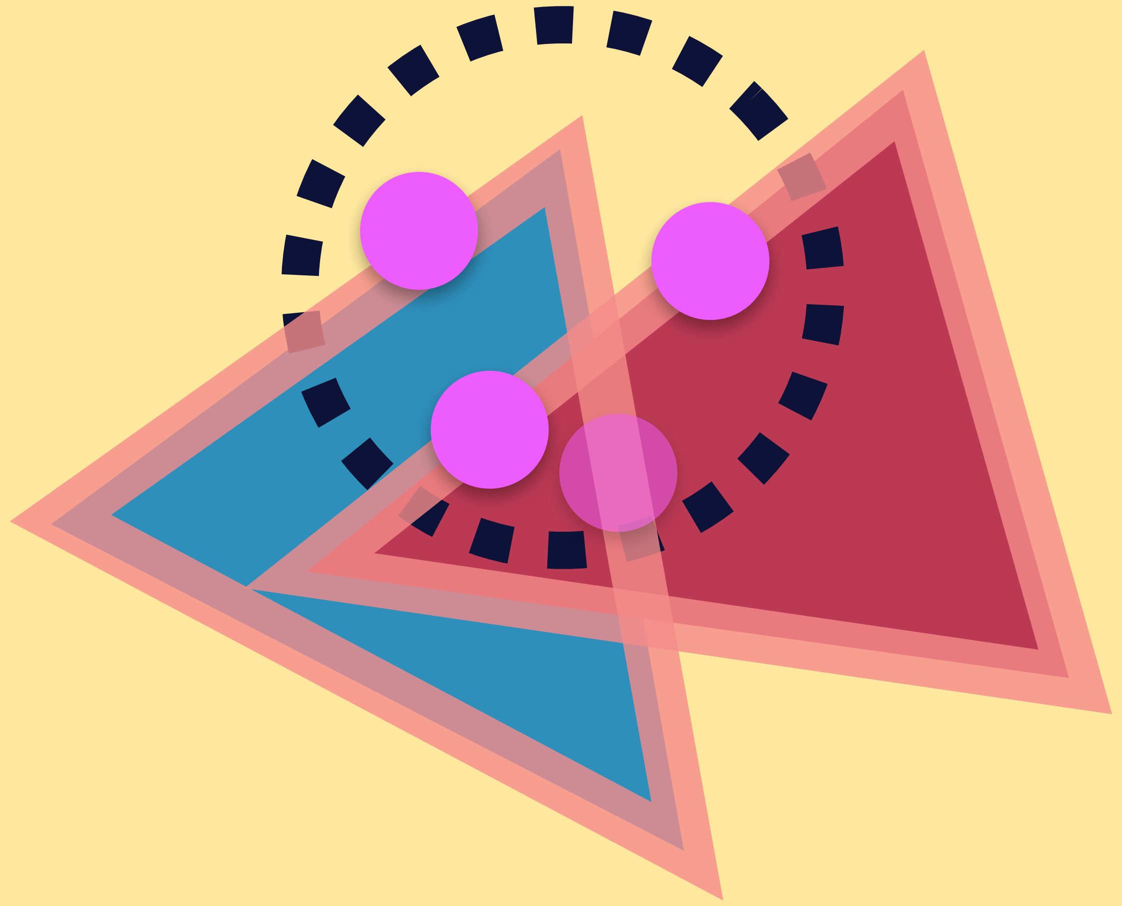


“the Leibniz’s integral rule”

GENERALIZE TO 2D

interior derivative

$$\frac{\partial}{\partial p}\iint$$  $$= \iint \frac{\partial}{\partial p}$$ 

$$+\int$$ 

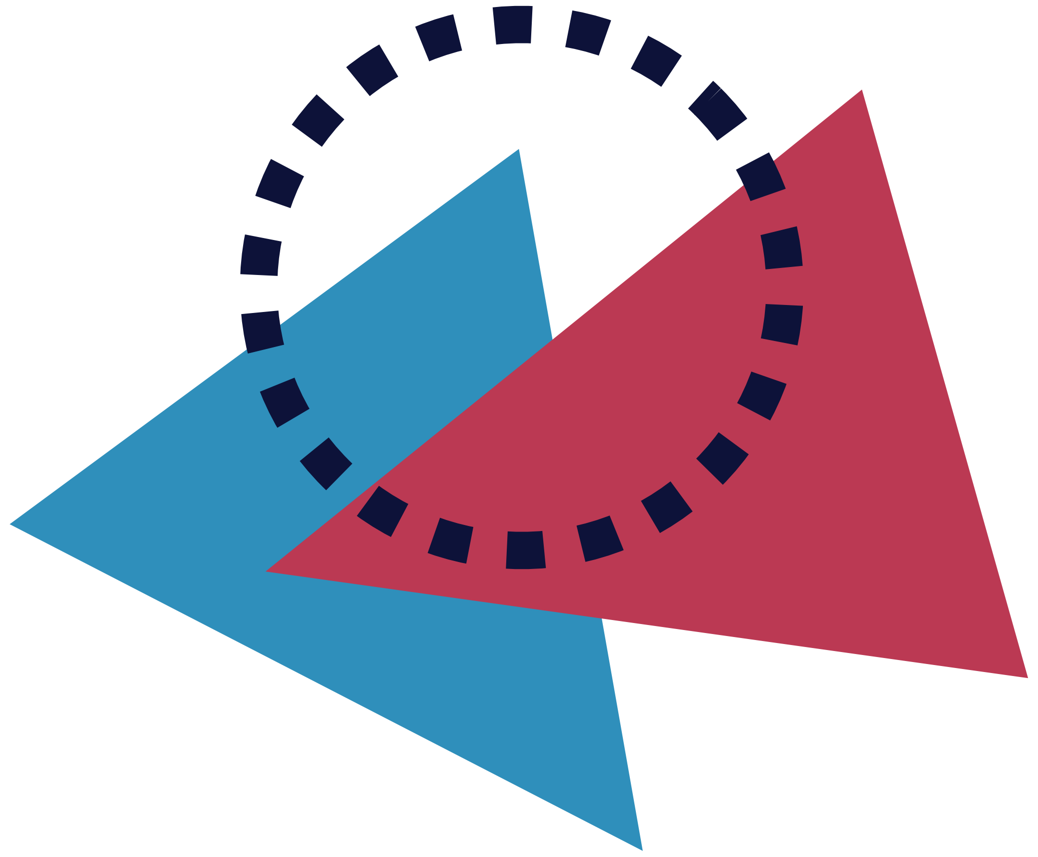
boundary derivative

Reynolds transport theorem
[Reynolds 1903]

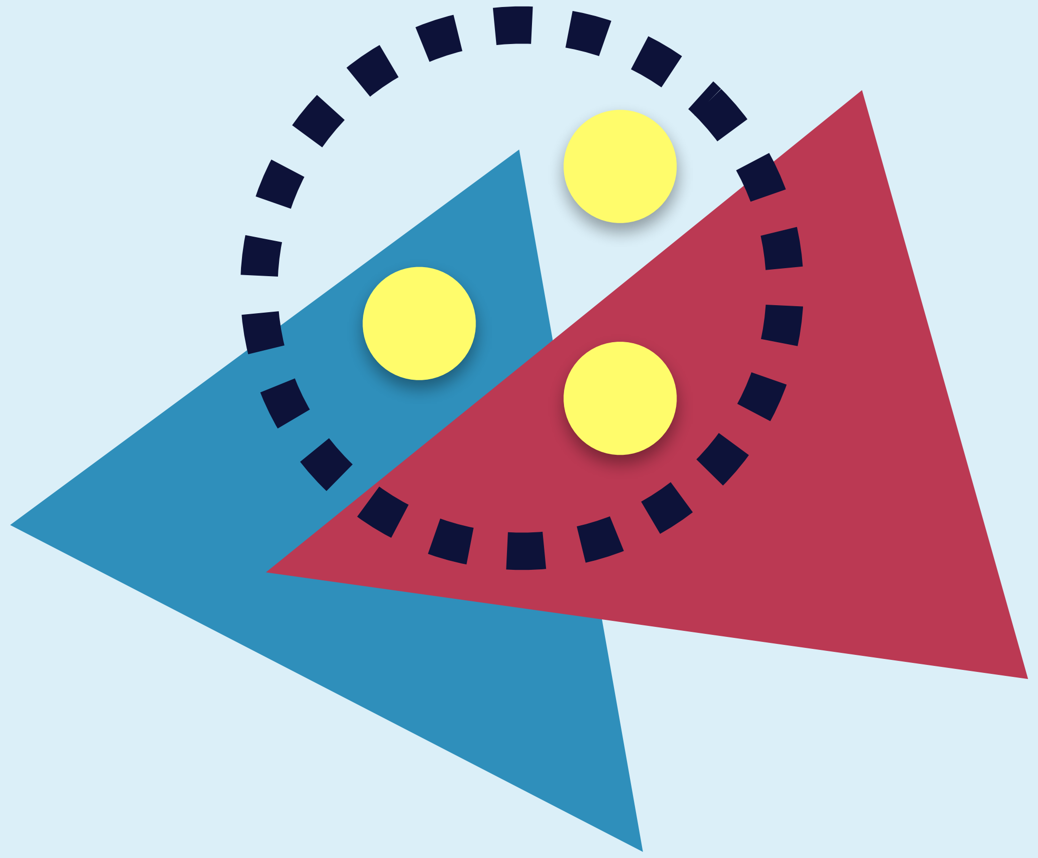
GENERALIZE TO 2D

interior derivative

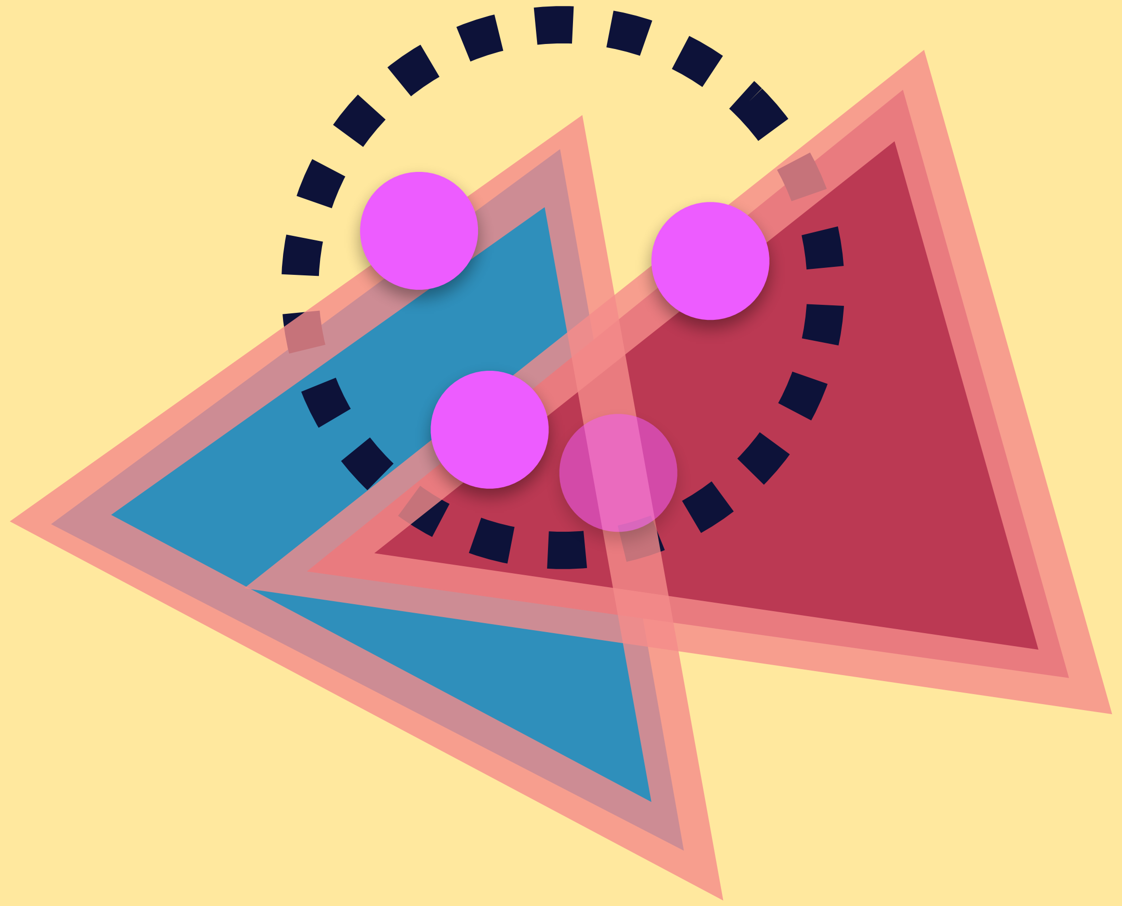
$$\frac{\partial}{\partial p} \iint$$



$$= \iint \frac{\partial}{\partial p}$$



$$+ \int$$

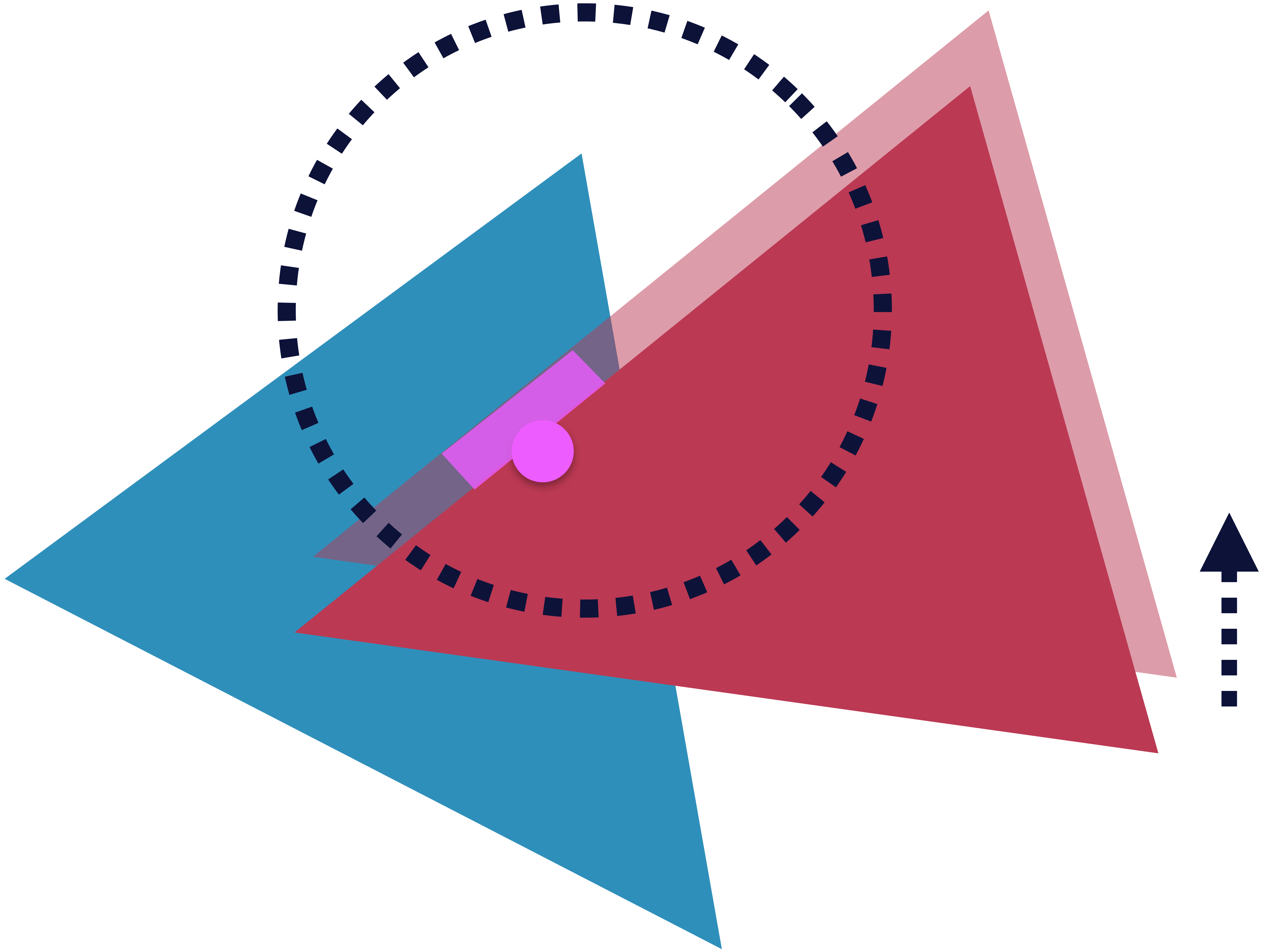


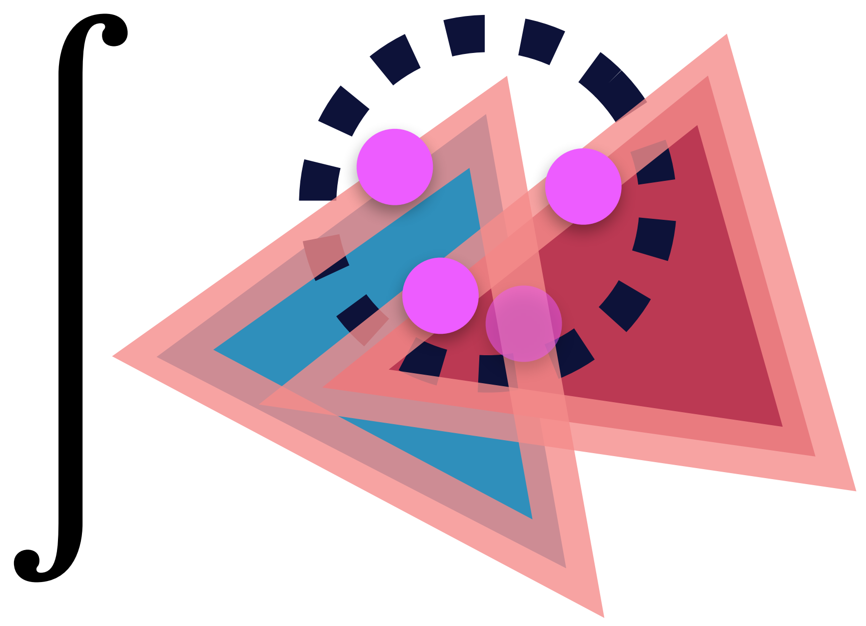
boundary derivative

Reynolds transport theorem
[Reynolds 1903]

DERIVING THE 2D BOUNDARY DERIVATIVE

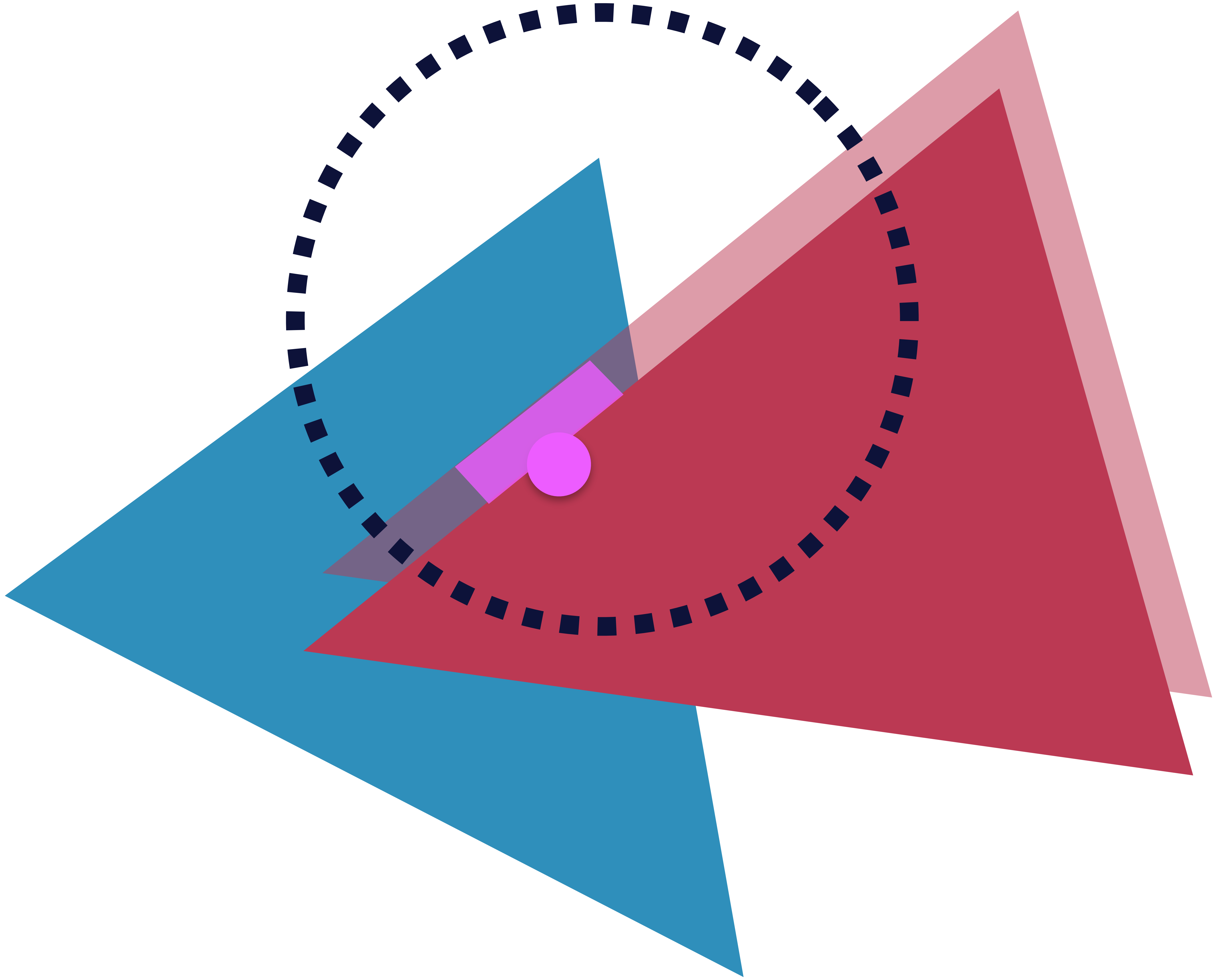
- boundary derivative = infinitesimal volume change w.r.t. parameter

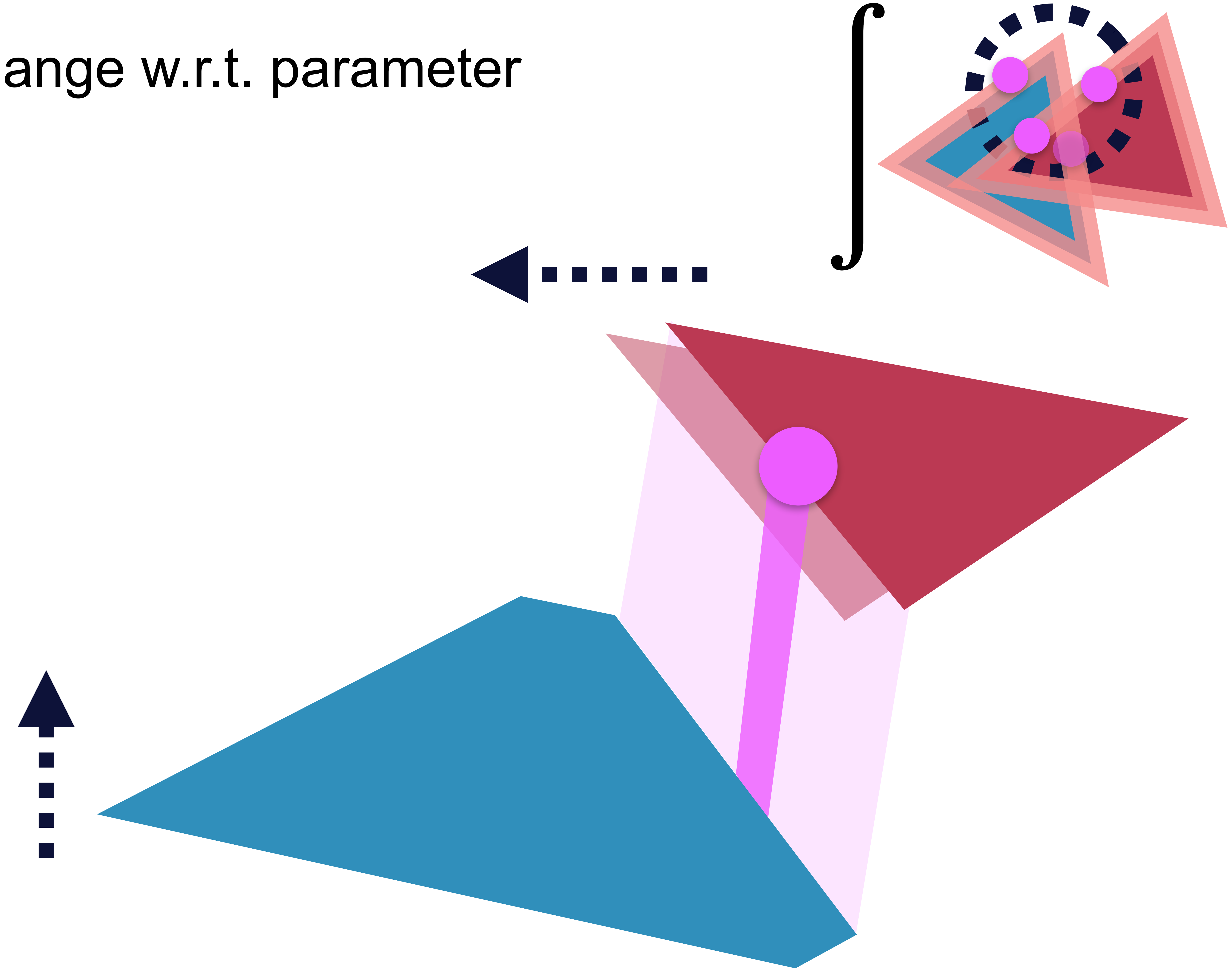




DERIVING THE 2D BOUNDARY DERIVATIVE

- boundary derivative = infinitesimal volume change w.r.t. parameter





3D view around the purple sample

DERIVING THE 2D BOUNDARY DERIVATIVE

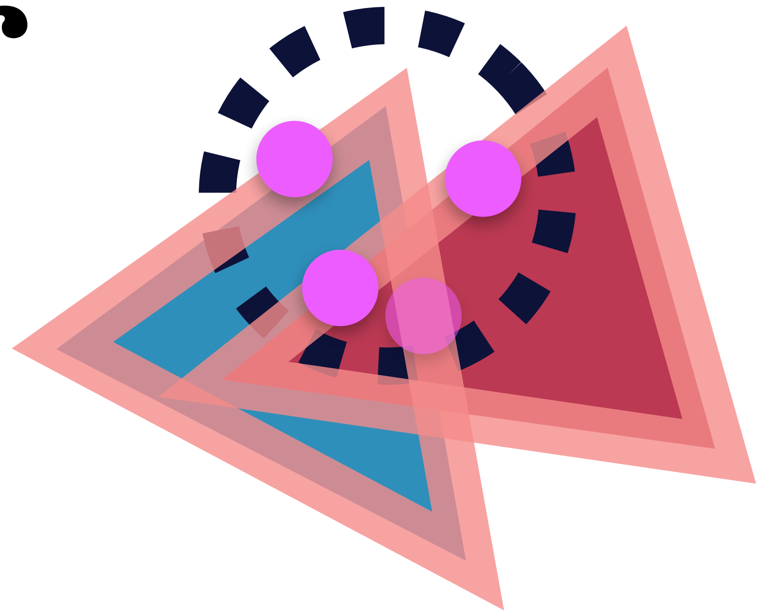
- boundary derivative = infinitesimal volume change w.r.t. parameter

DERIVING THE 2D BOUNDARY DERIVATIVE

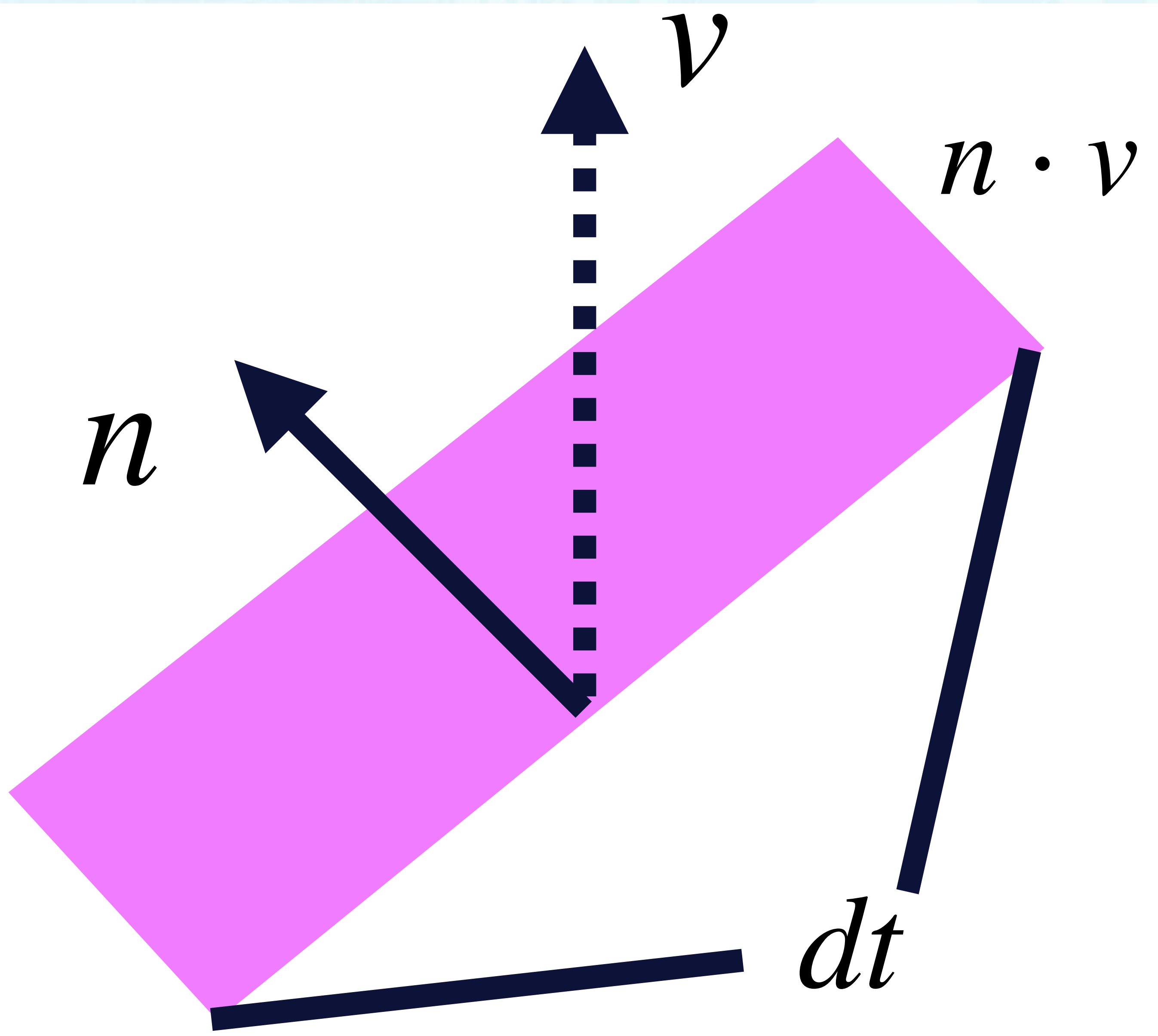
$v =$ boundary movement w.r.t. param

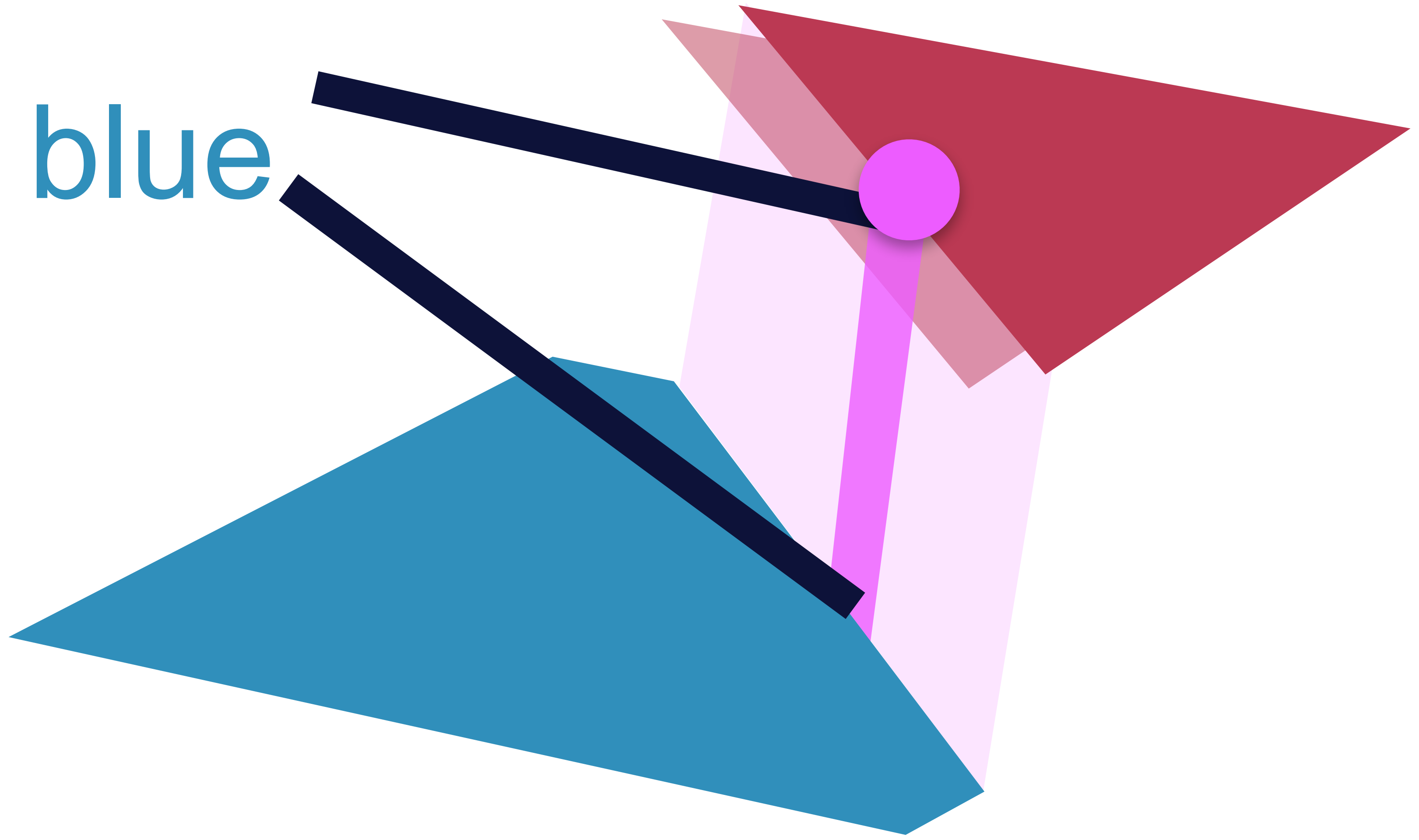
$$= \frac{\partial x}{\partial p}$$

THE INFINITESIMAL BOUNDARY VOLUME

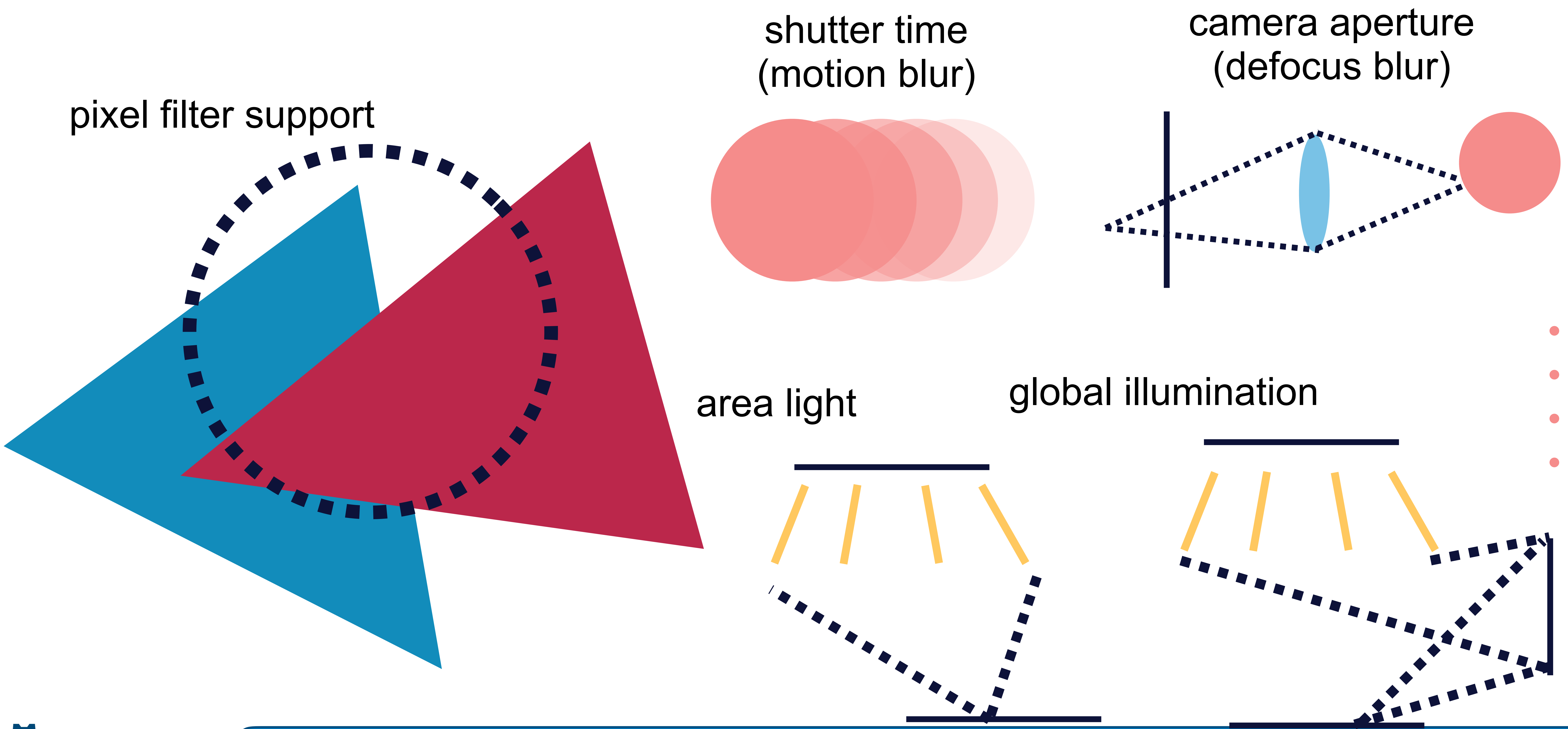
$$\int$$  $$dt = \int \left(f_- - f_+\right)(n \cdot v)\, dt$$

height width length





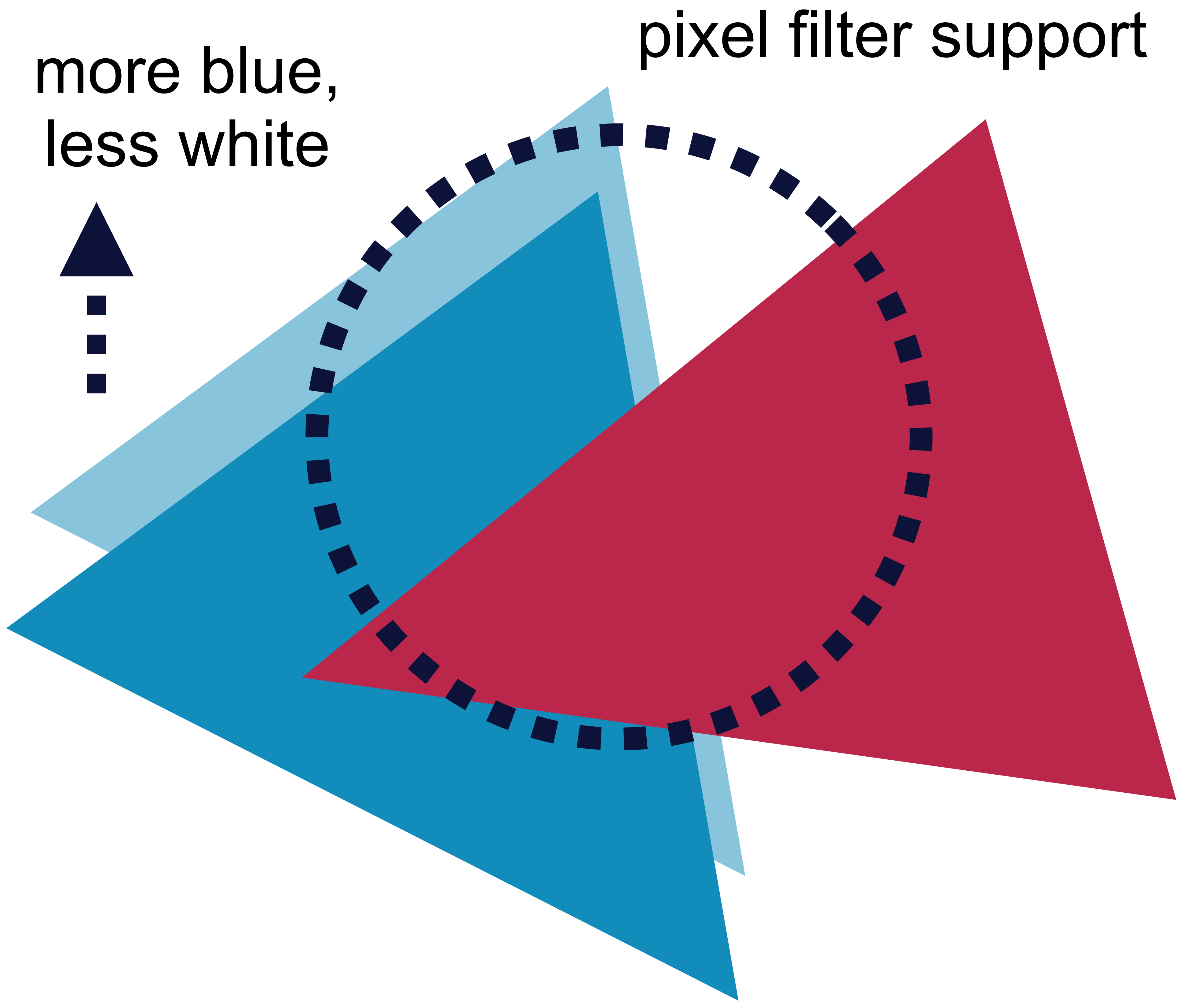
RENDERING = COMPUTING INTEGRALS



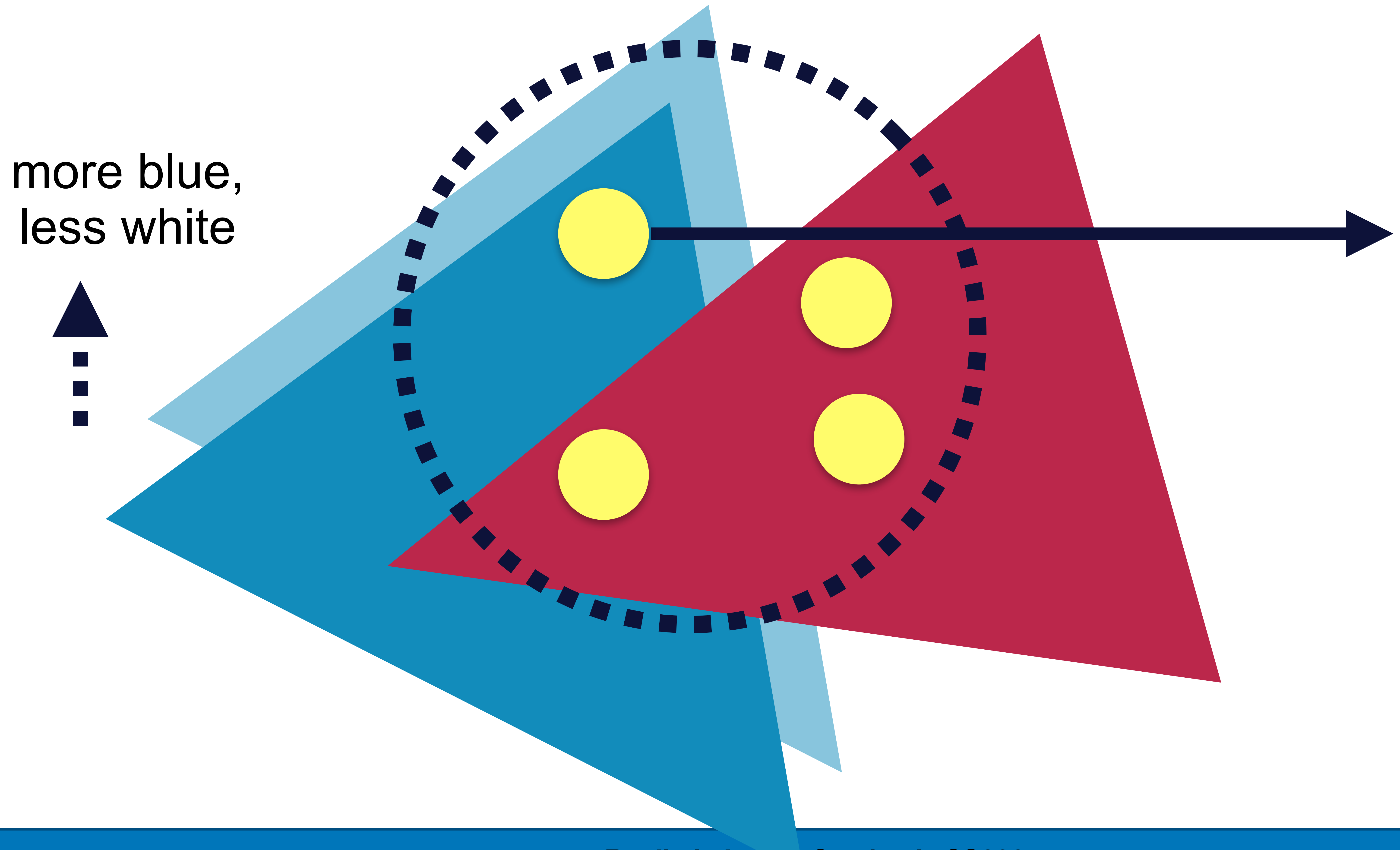
- wavelength
- transmittance
- …
- and more!

RECAP

- **While the *integrand* is discontinuous, the *integral* is differentiable!**
  - the average color changes continuously as triangles move



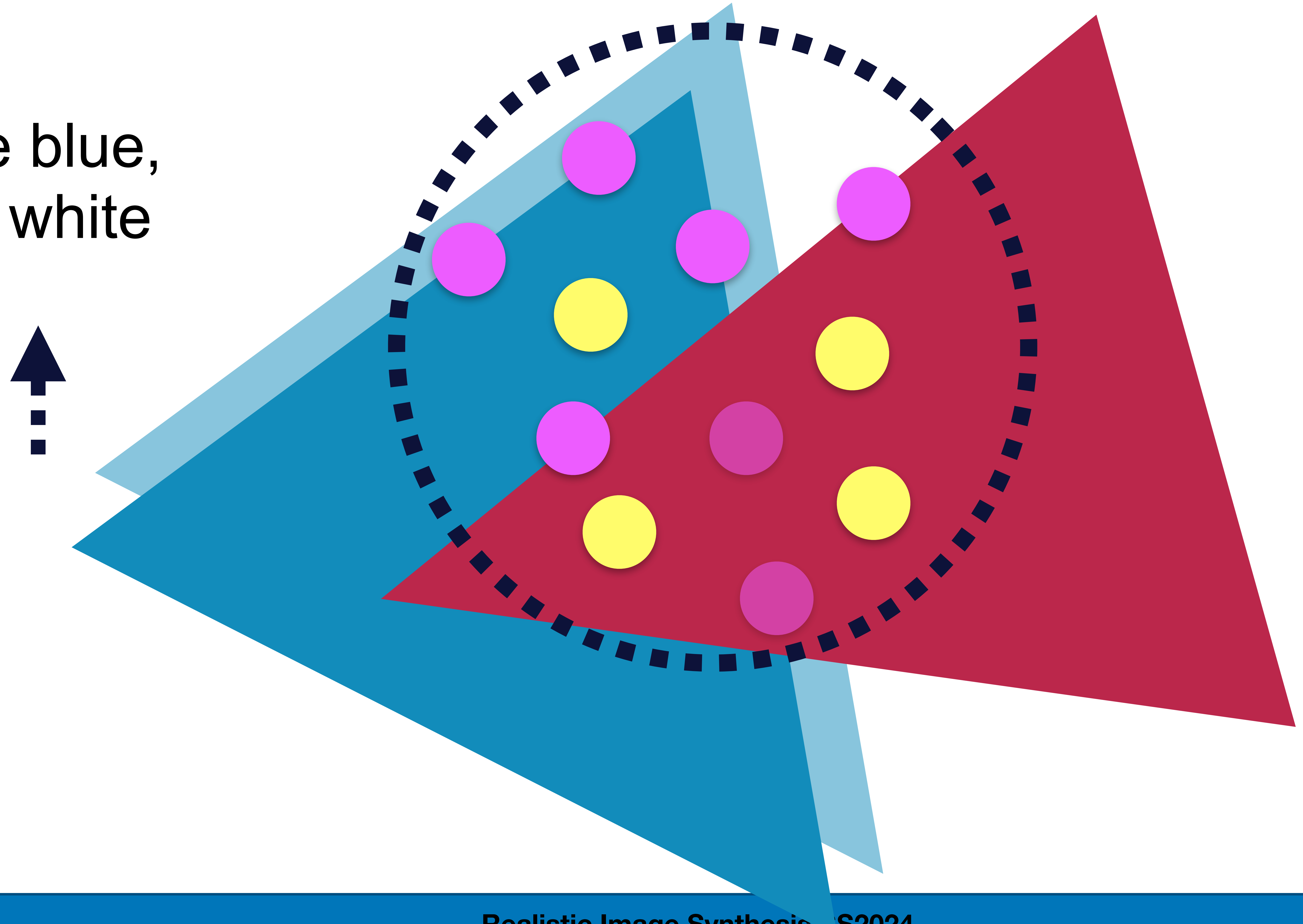
DIFFERENTIATING INTEGRAL SAMPLES GIVES WRONG DERIVATIVES



$$\frac{\partial}{\partial p} = 0$$

KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

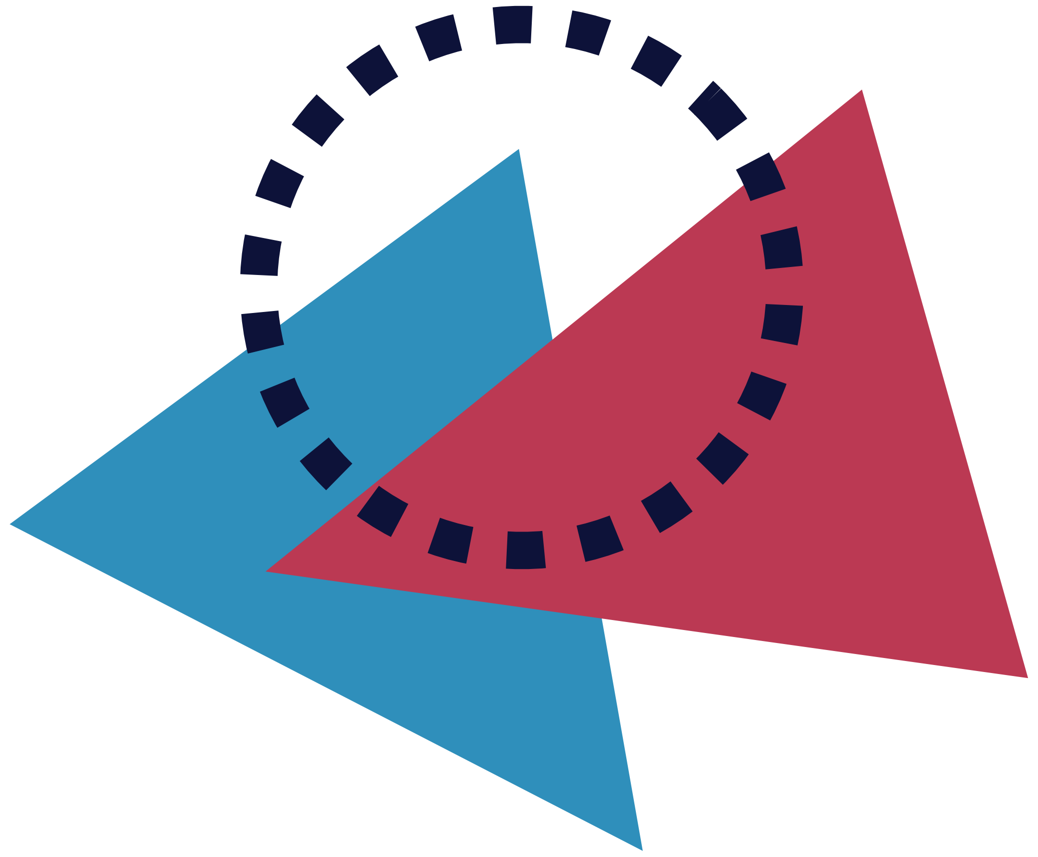
more blue,
less white



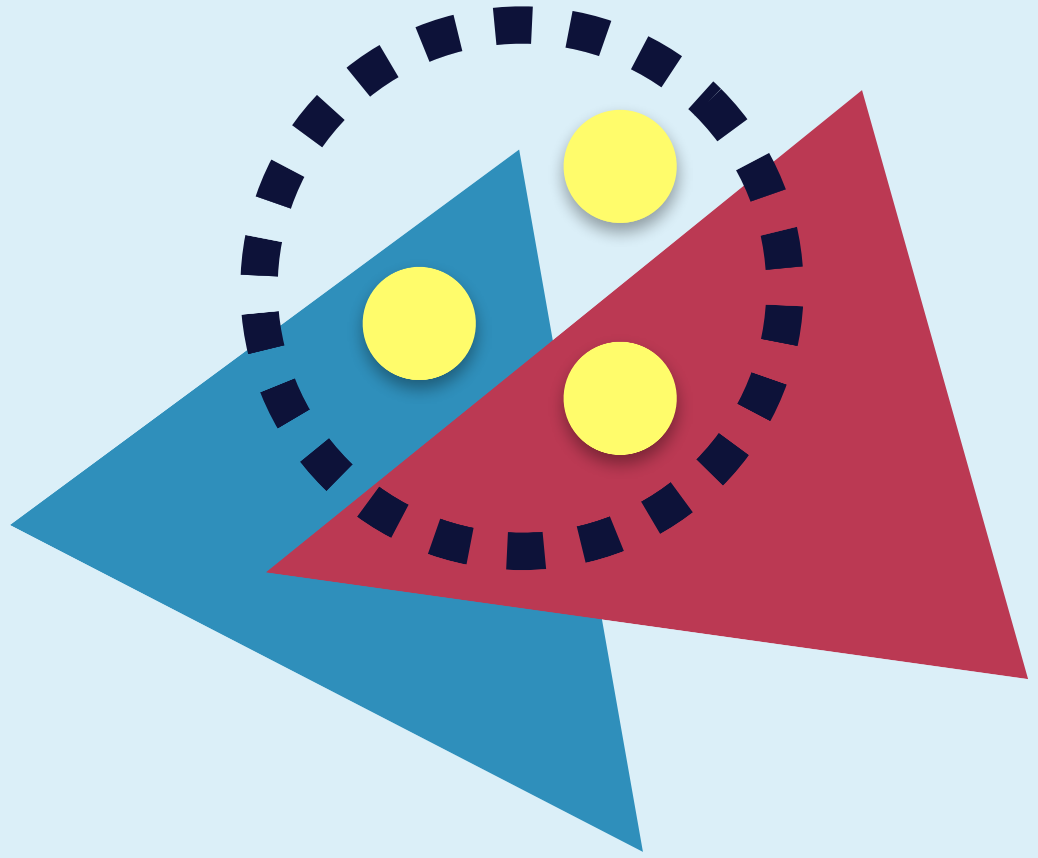
RECAP

interior derivative

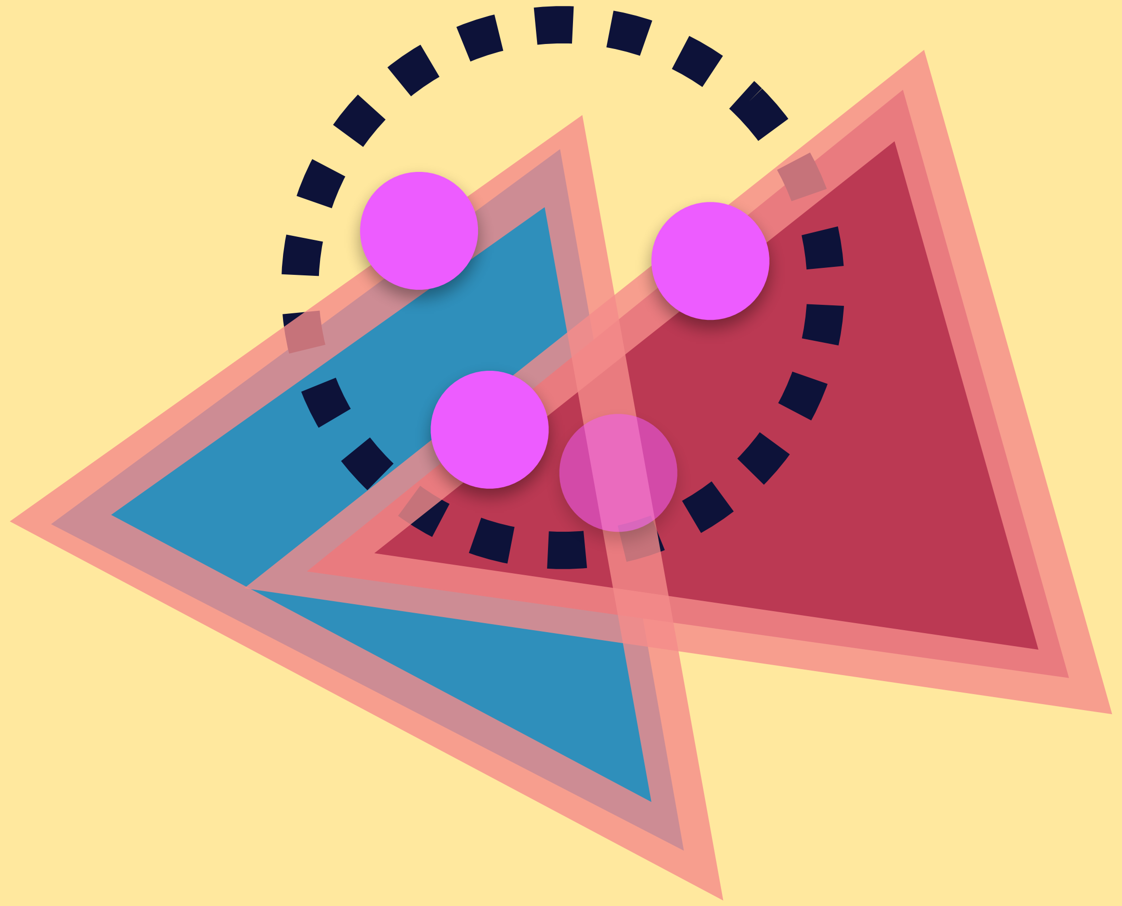
$$\frac{\partial}{\partial p}\iint$$



$$= \iint \frac{\partial}{\partial p}$$



$$+ \int$$



boundary derivative

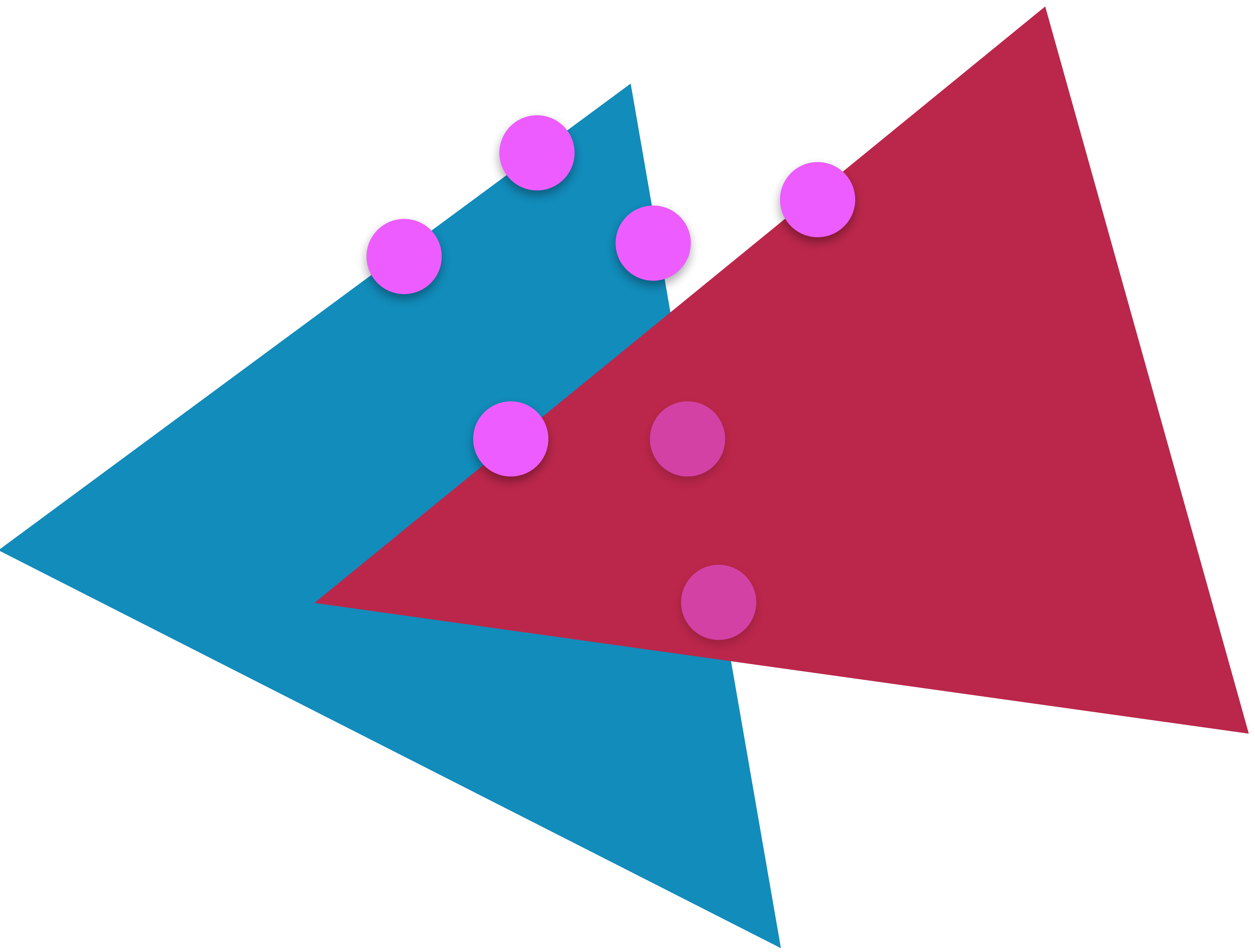
Reynolds transport theorem
[Reynolds 1903]

DISCUSSION

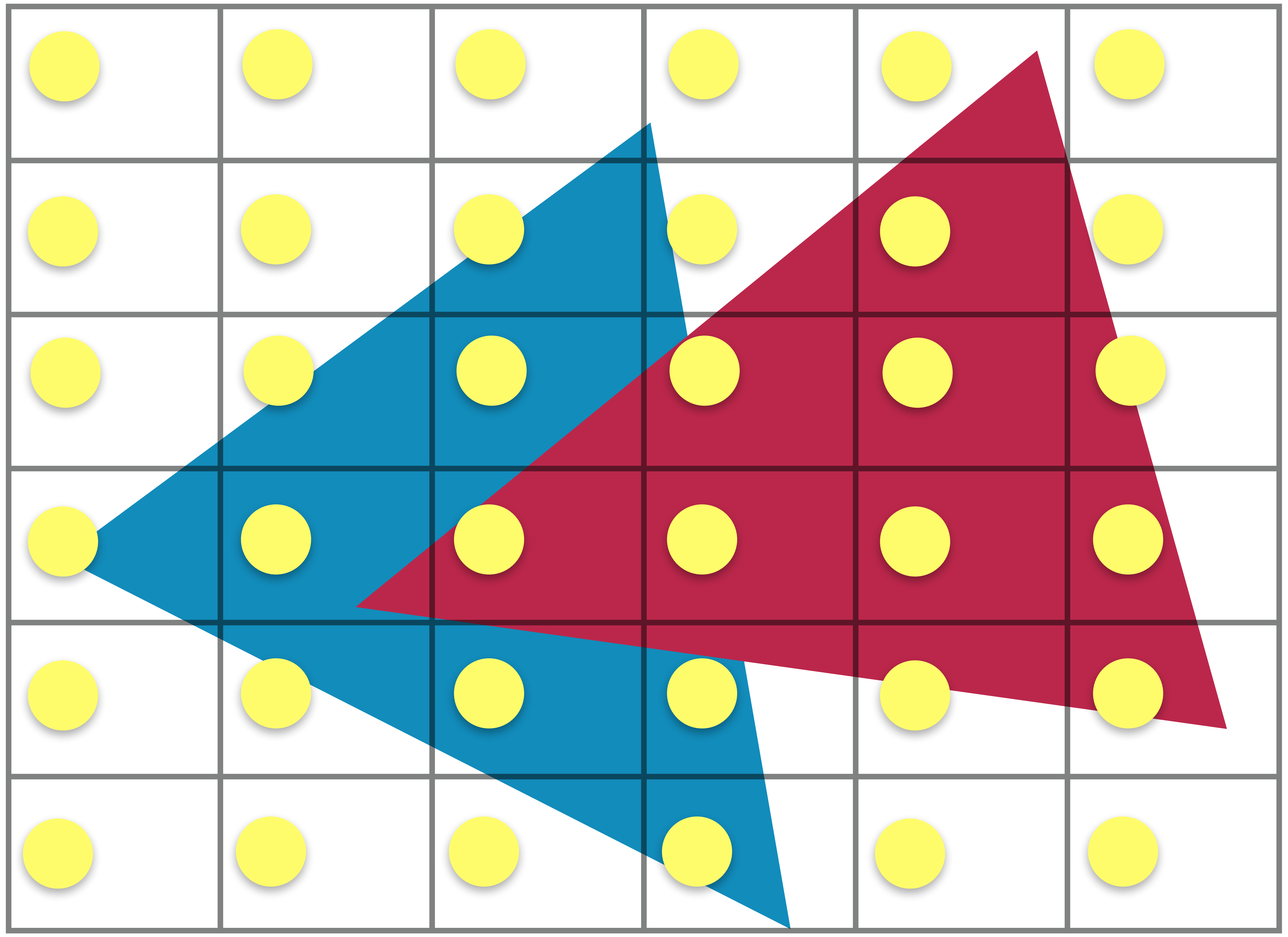
- Ray tracing vs rasterization
- Approximated solutions
- Geometry representation
- Limitations

RAY TRACING VS RASTERIZATION

- The boundary sampling is not very compatible with z-buffer rendering

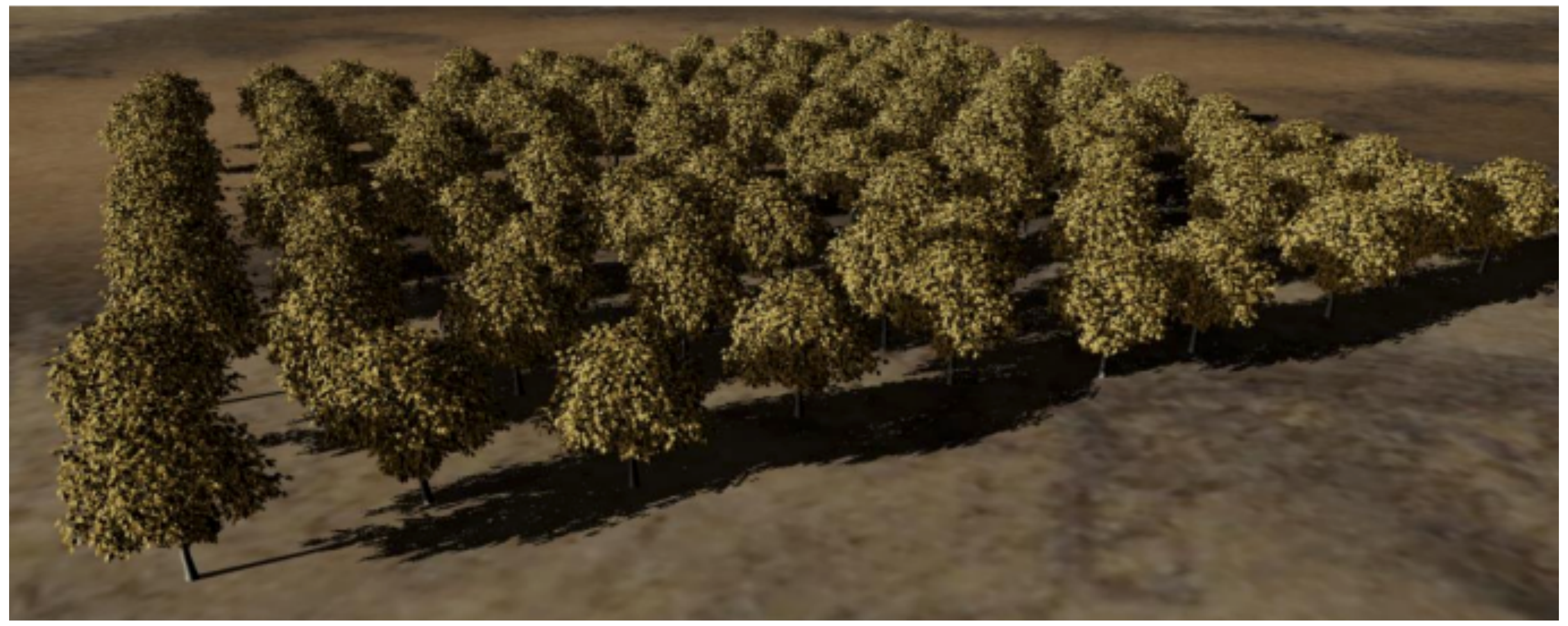


v.s.



RAY TRACING VS RASTERIZATION

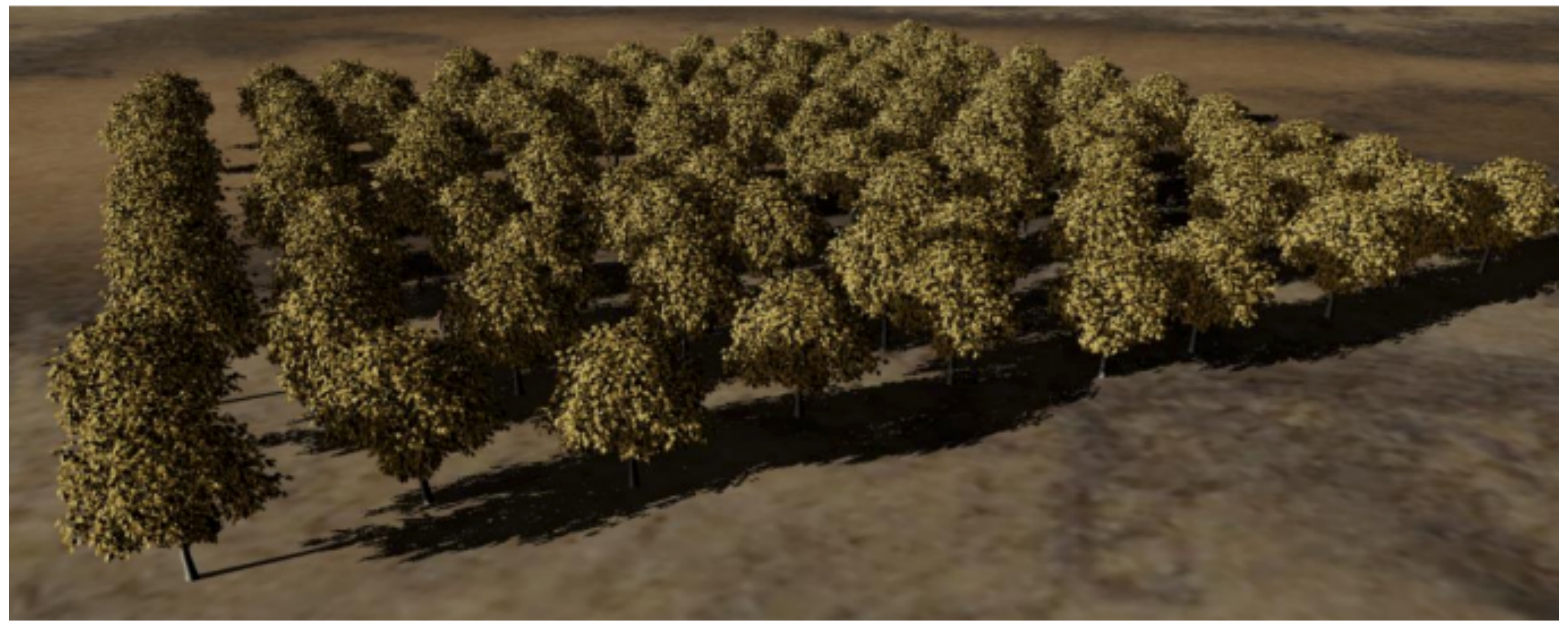
- Ray tracing is not significantly slower than rasterization
- The interior derivatives can be computed using rasterization



from Gruen 2020
1080p, ~19M triangles
raster: 2.7 ms
raytrace: 8.6 ms (2.5 ms for animation)

RAY TRACING VS RASTERIZATION

- Ray tracing is not significantly slower than rasterization
- The interior derivatives can be computed using rasterization
- Visibility queries may not be the main bottleneck

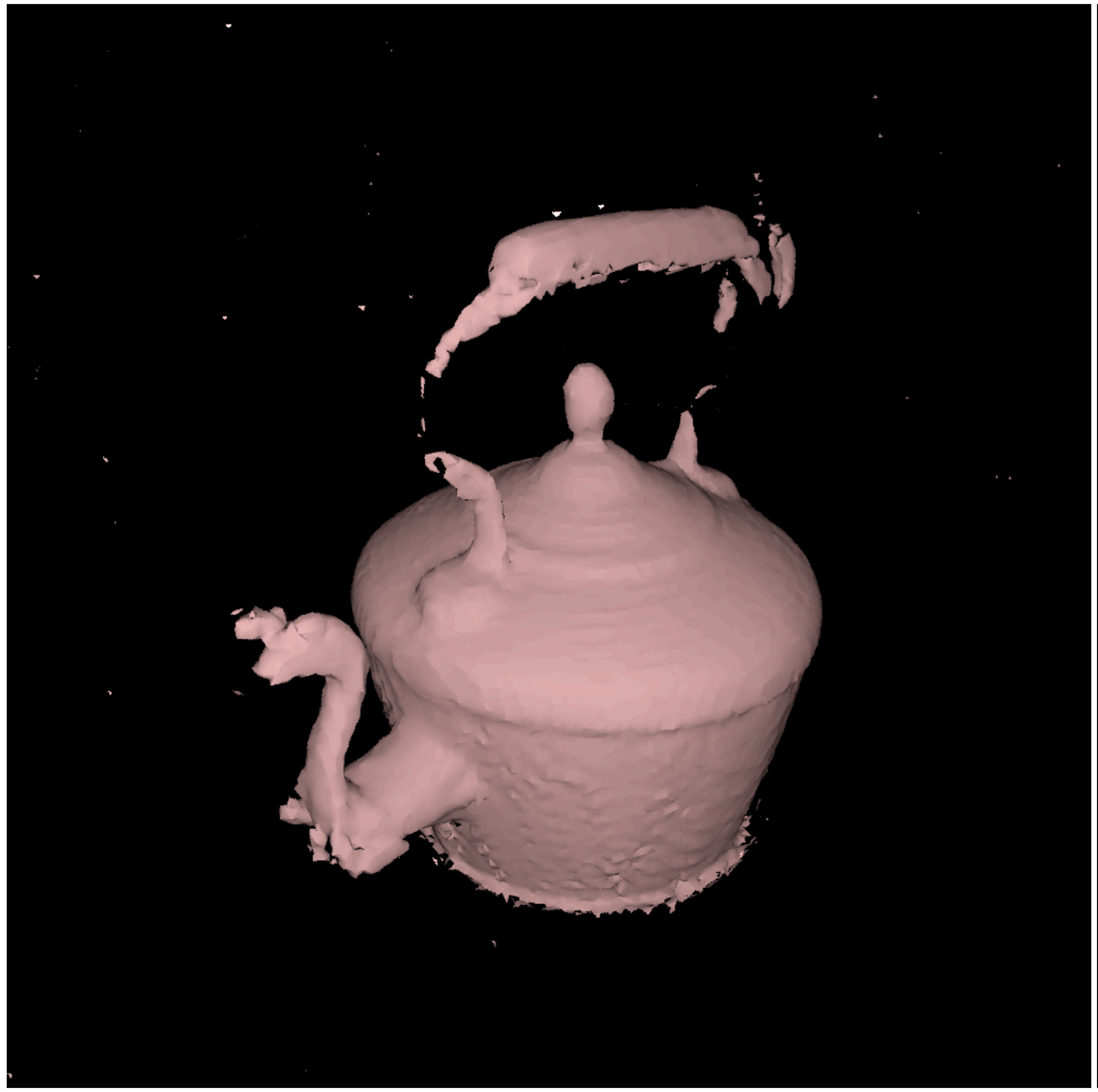


from Gruen 2020
1080p, ~19M triangles
raster: 2.7 ms
raytrace: 8.6 ms (2.5 ms for animation)

~10k faces, 256x256 (Titan Xp)
PyTorch3D (raster) 220ms
redner (raytrace) 60ms
(BVH 20ms, forward 7ms, backward 27ms)

RAY TRACING VS RASTERIZATION

23823 vertices, 44702 faces



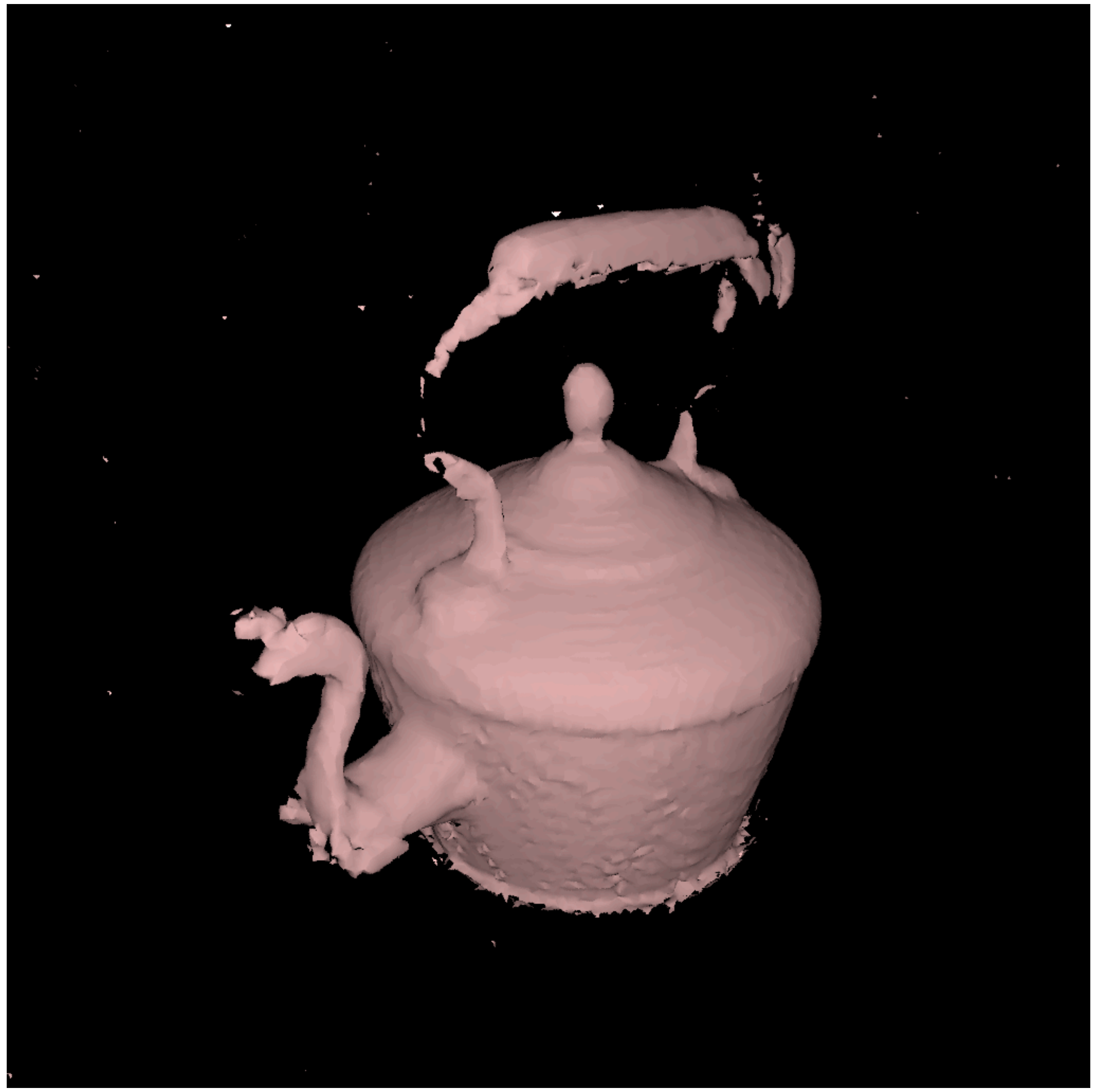
initial

target

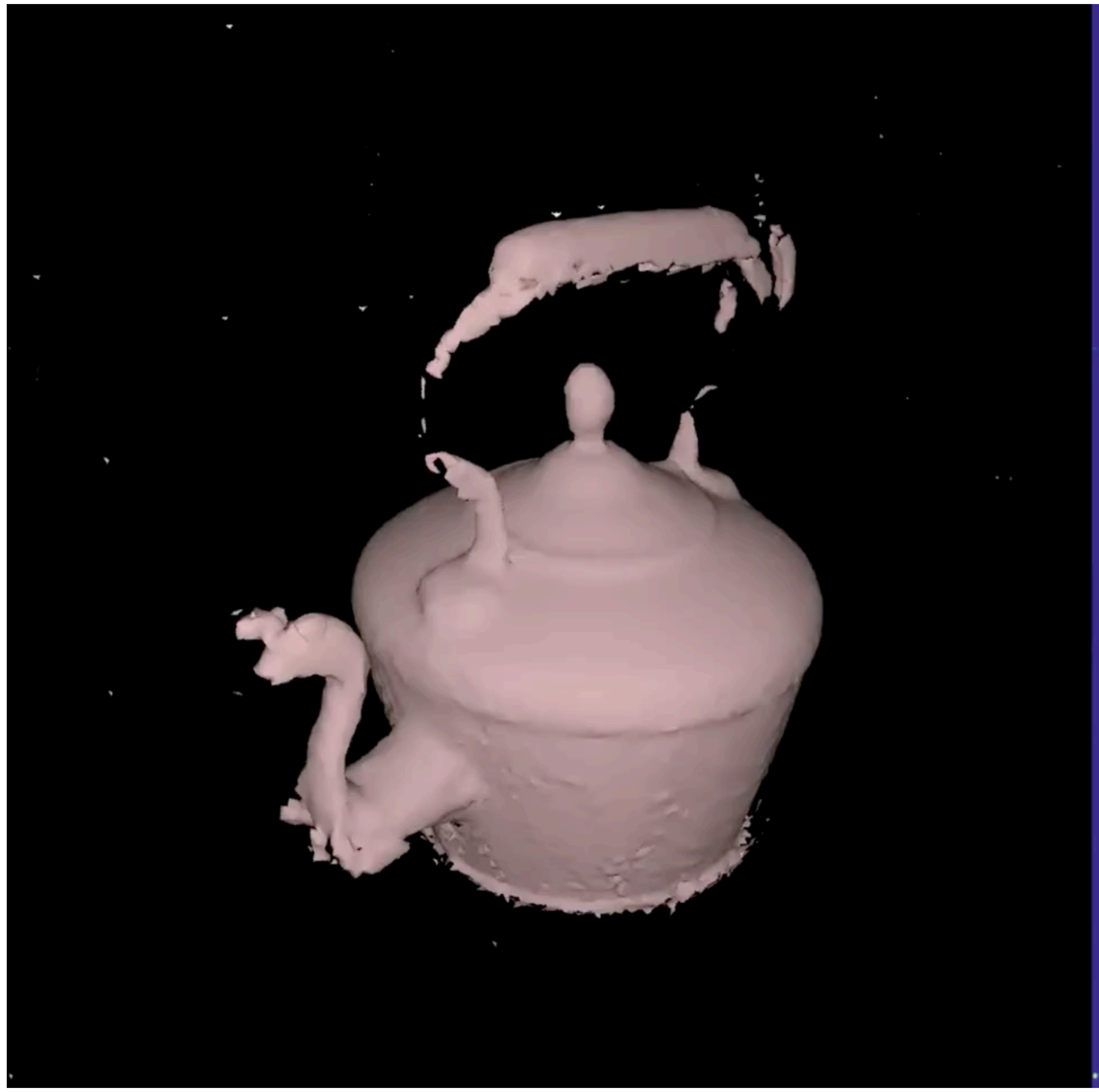
- 1024x1024 at 2 spp (Titan Xp) forward + backward
  - Ray tracing + edge sampling: 0.05—0.1 sec
  - PyTorch3D: 0.15 sec

RAY TRACING VS RASTERIZATION

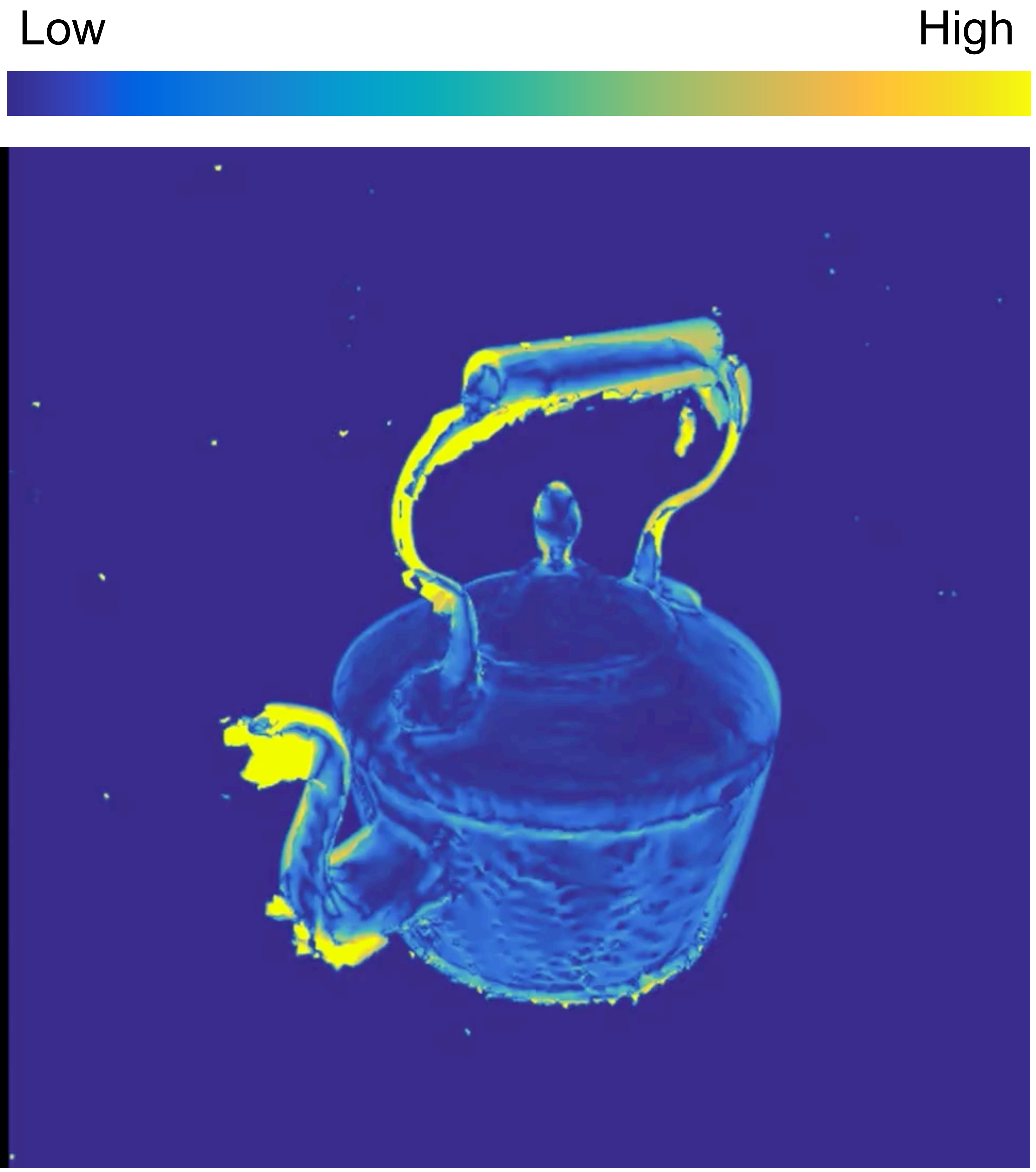
23823 vertices, 44702 faces



initial



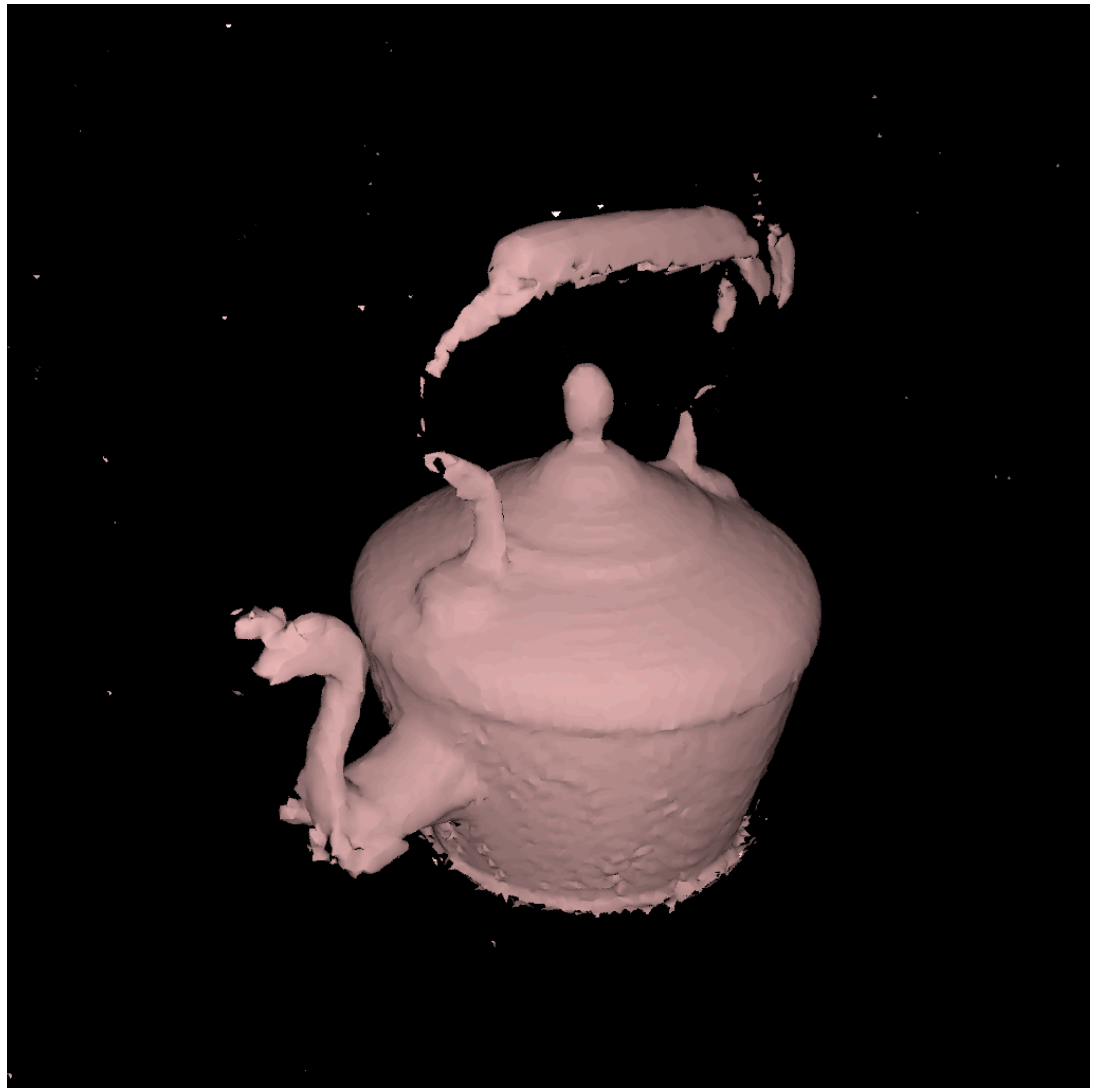
edge sampling
optimization video
(1 view over 20)



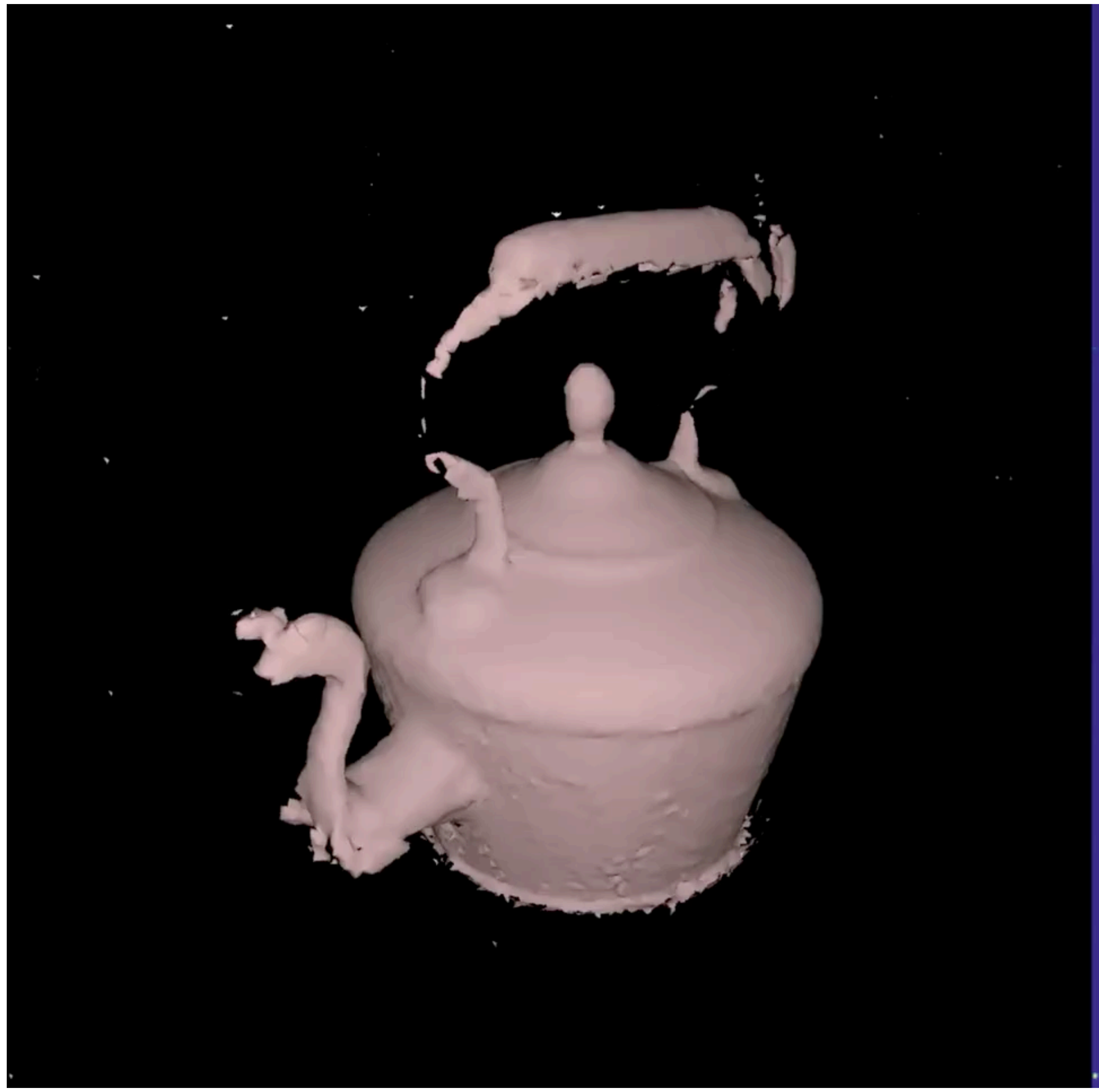
abs. error

RAY TRACING VS RASTERIZATION

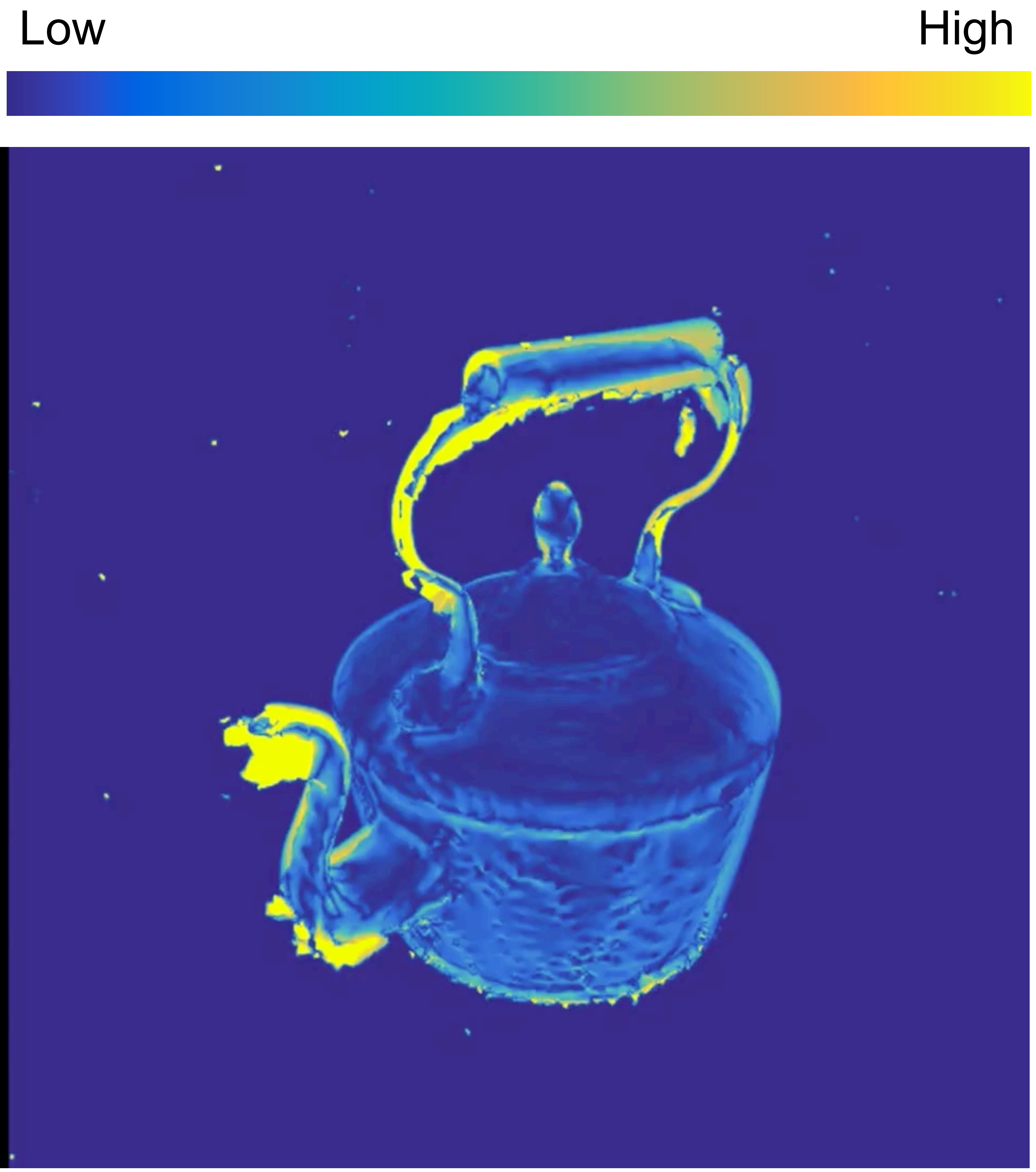
23823 vertices, 44702 faces



initial



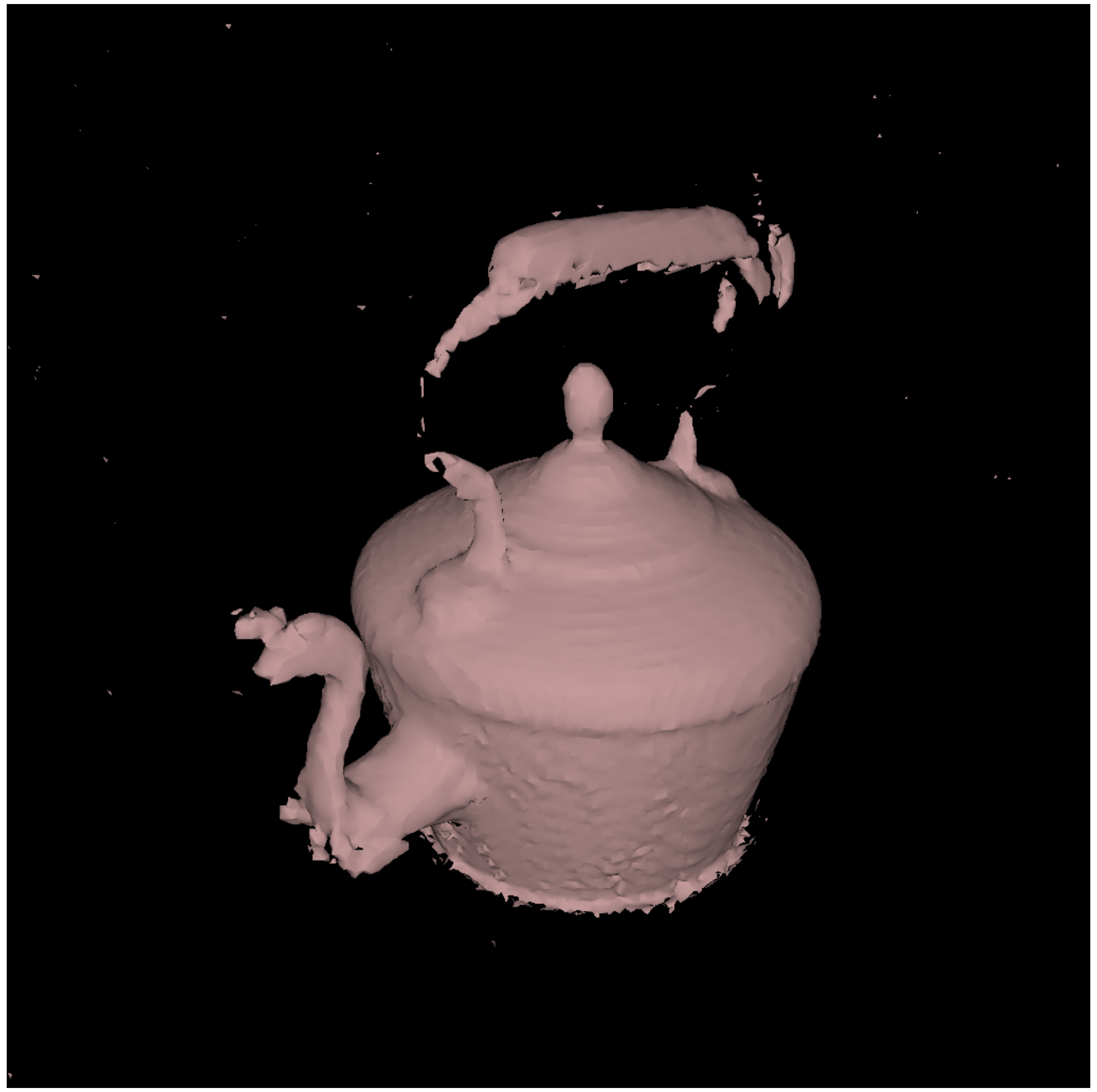
edge sampling optimization video (1 view over 20)



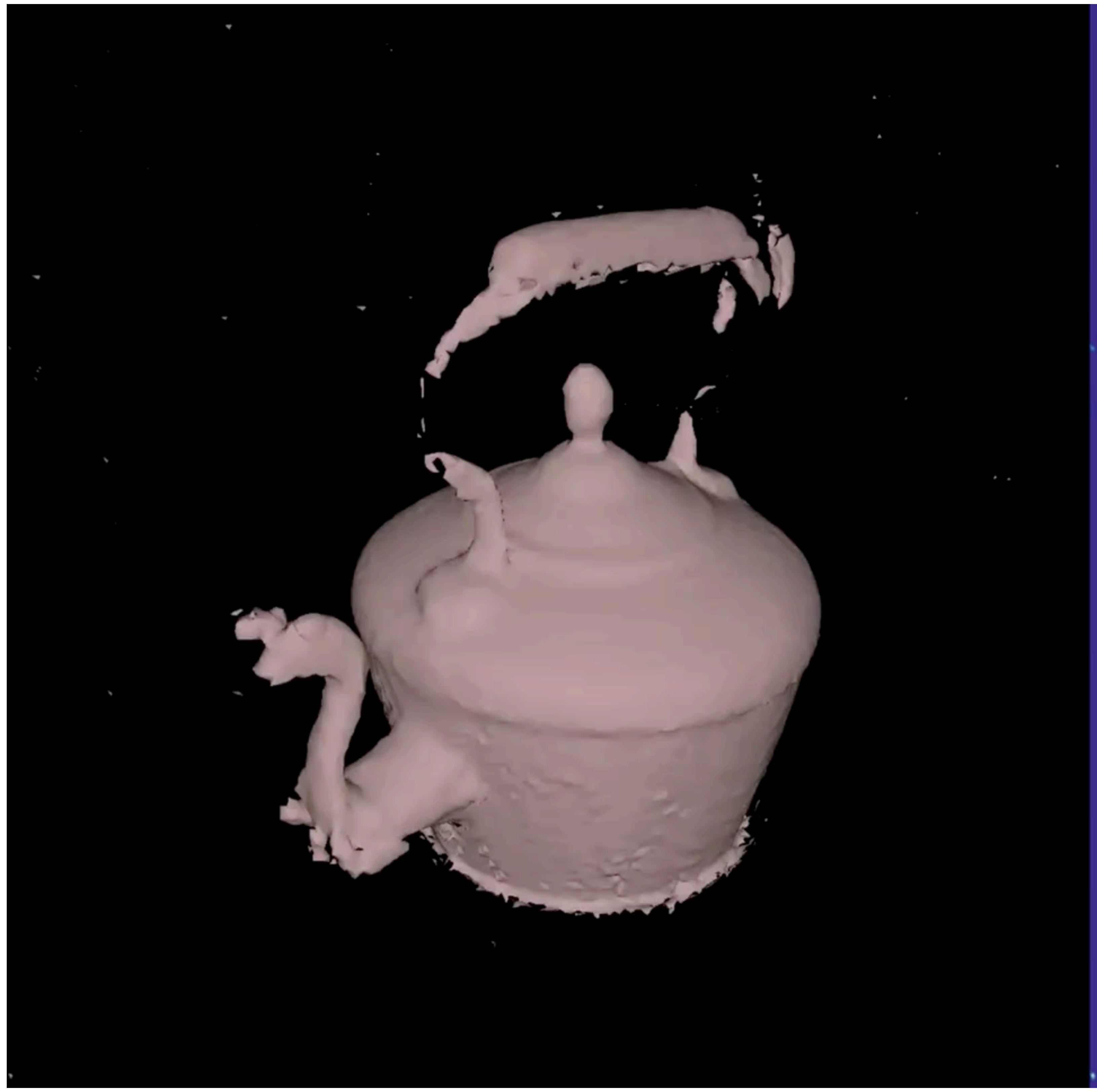
abs. error

RAY TRACING VS RASTERIZATION

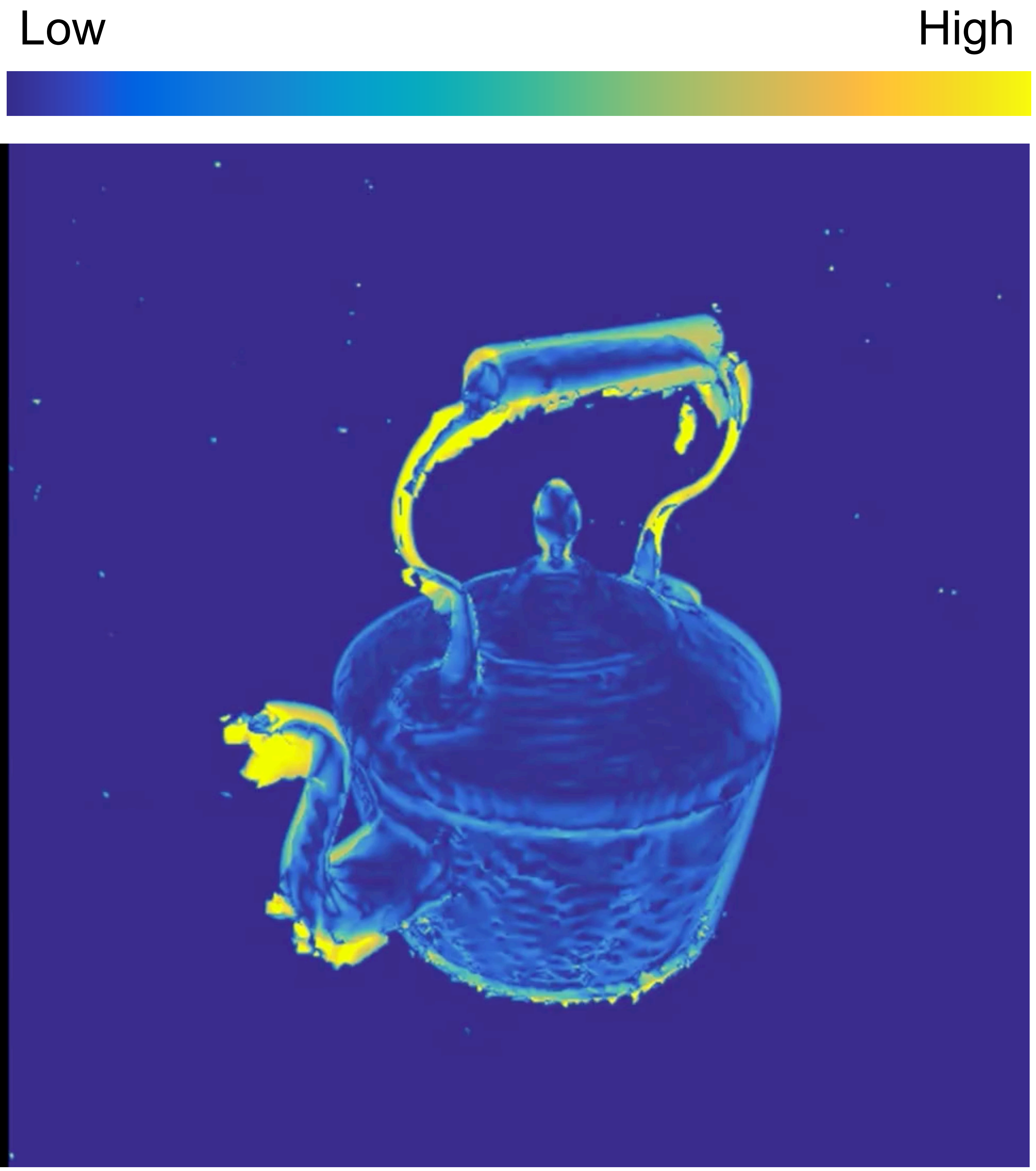
23823 vertices, 44702 faces



initial

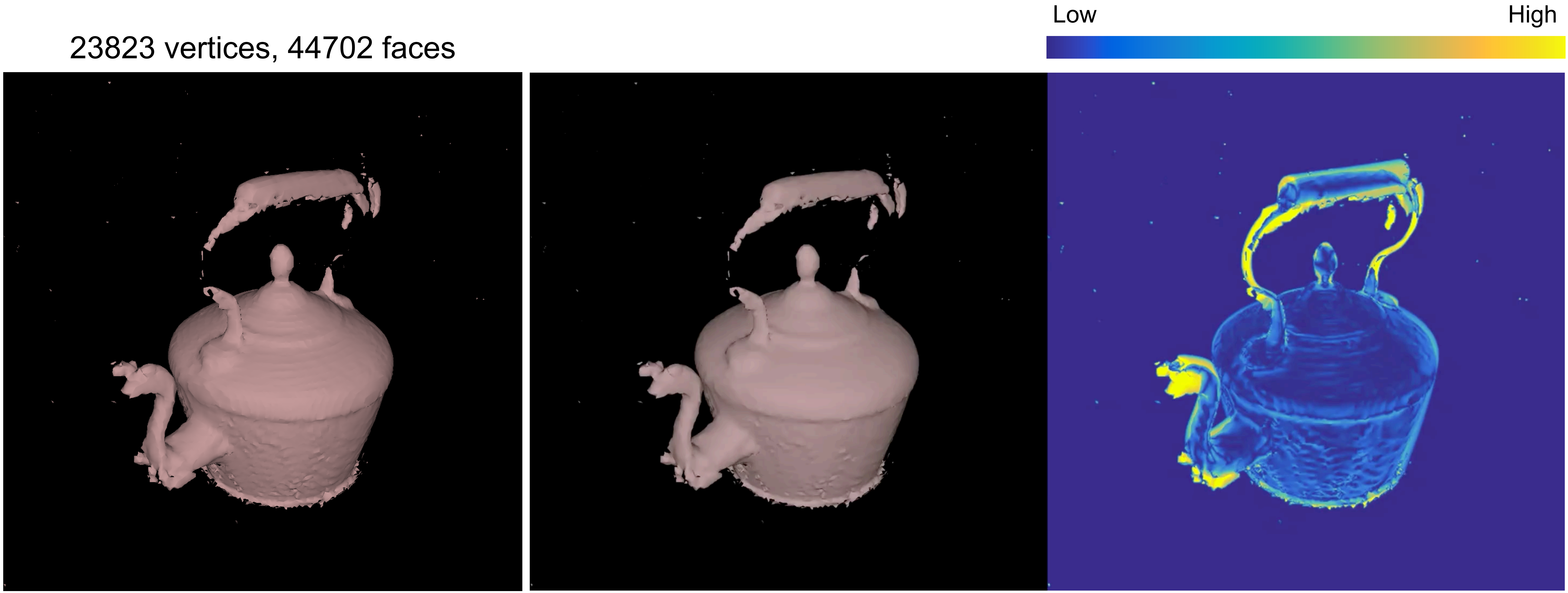


PyTorch3D
optimization video
(1 view over 20)



abs. error

RAY TRACING VS RASTERIZATION



initial

PyTorch3D
optimization video
(1 view over 20)

abs. error

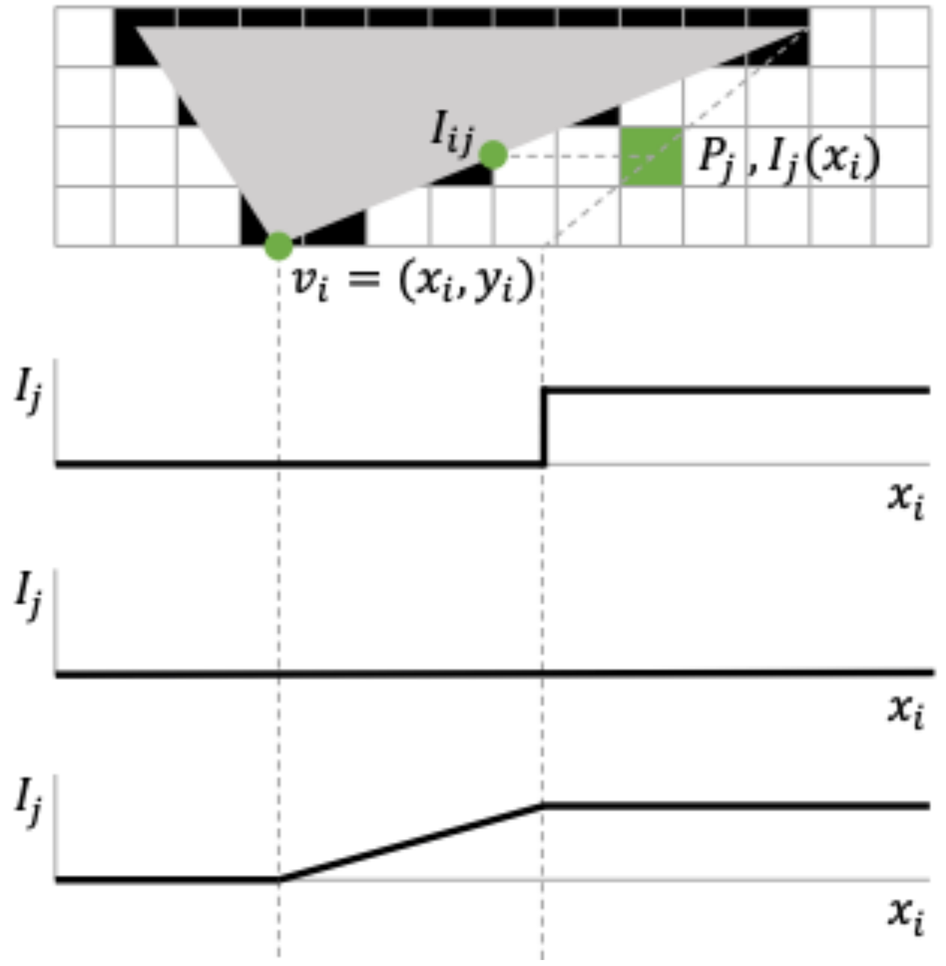
RAY TRACING VS RASTERIZATION

Optimization results after 5000 iterations (with identical settings)

optimized (ray tracing) | target | optimized (PyTorch3D)

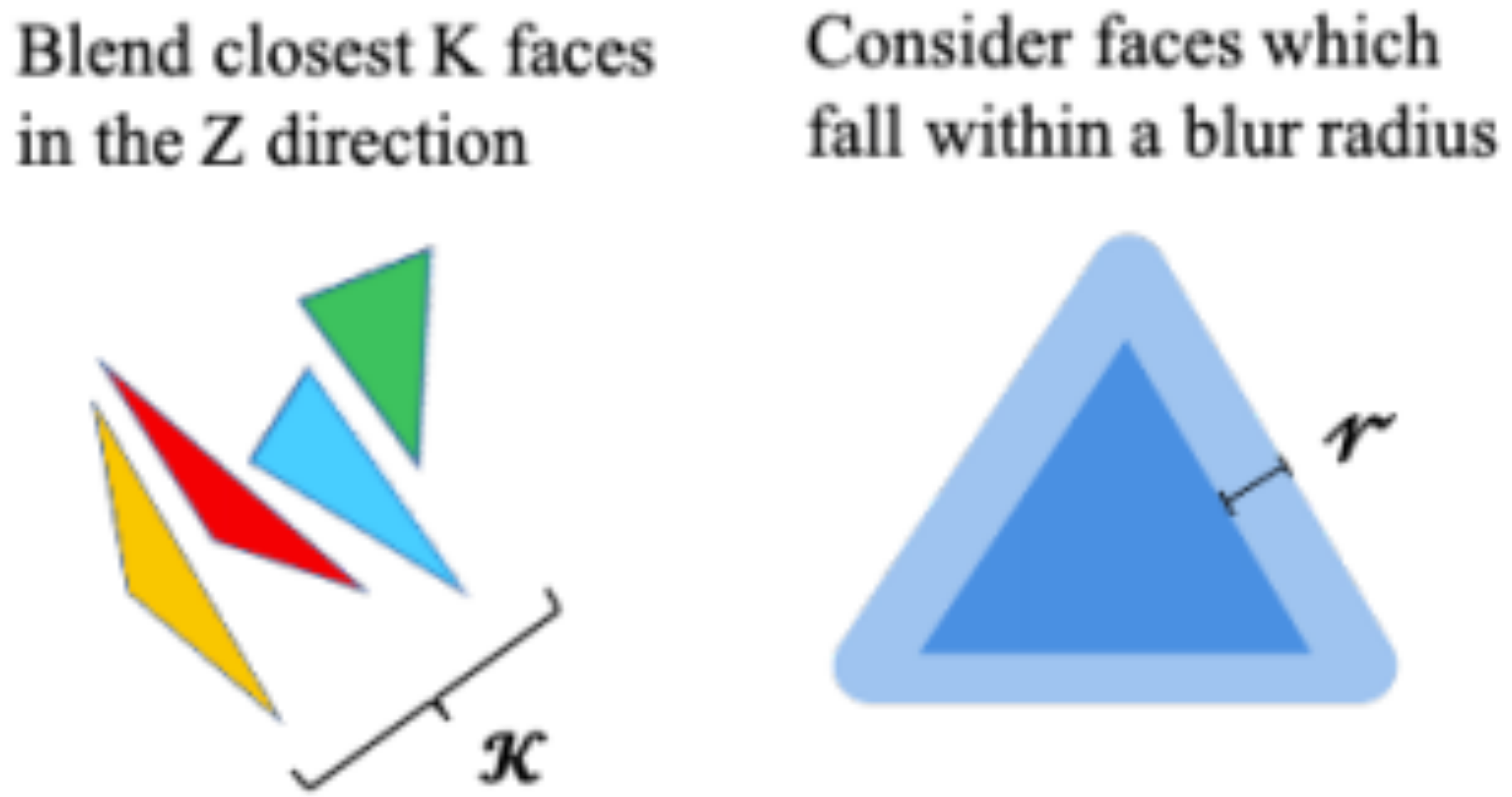
APPROXIMATED SOLUTION

- Our boundary integral is *correct, i.e.,* when the number of samples grows it converges to the integral.
- Two other kinds of approximation:
  - Keep the rendering model, approximate the derivatives (de La Gorce 2011, OpenDR 2014, Kato 2018, …)
  - Change the rendering model (Rhodin 2015, SoftRas 2019, PyTorch3D 2020…)



Kato 2018

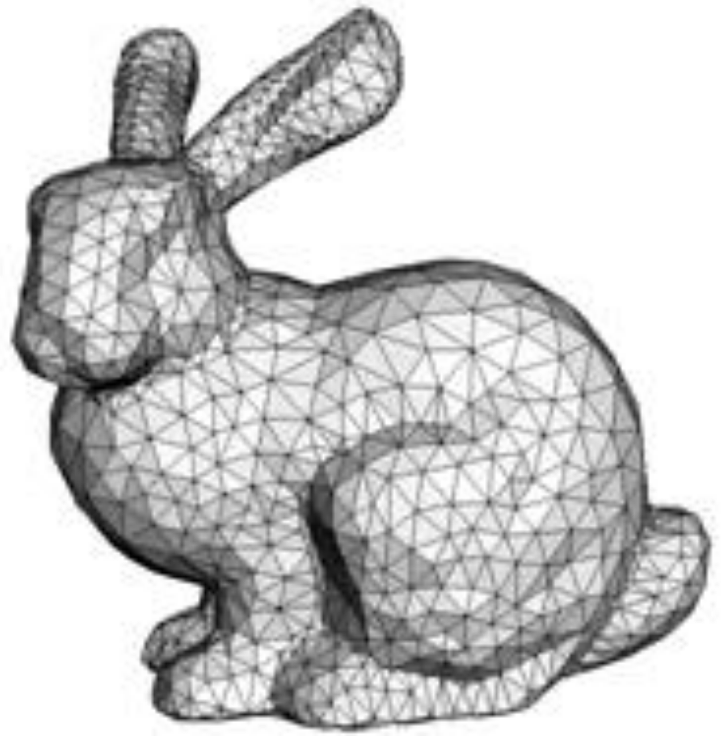
Rhodin 2015



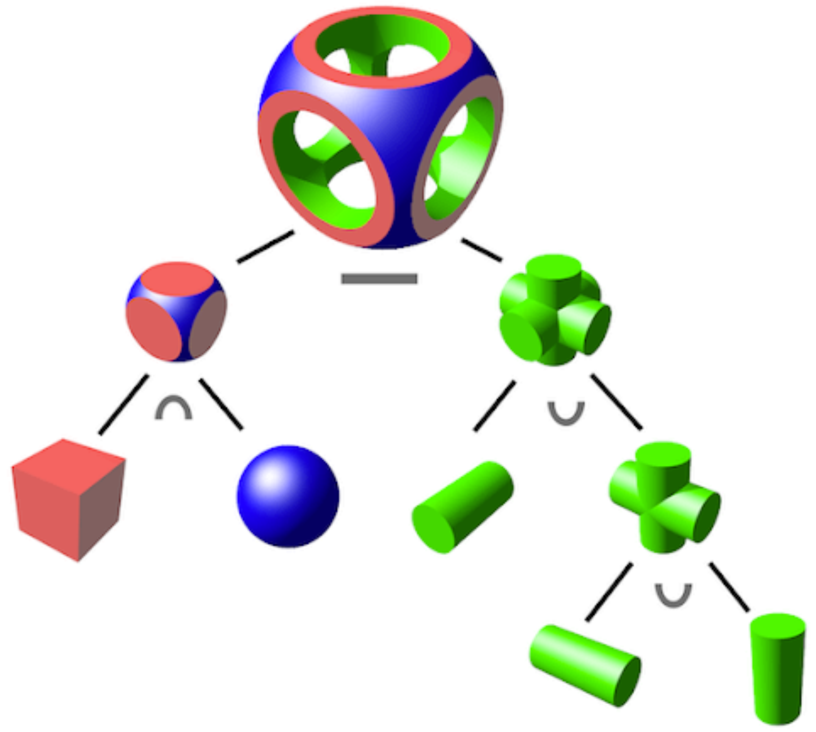
PyTorch3D [Ravi 2020]

GEOMETRY REPRESENTATION

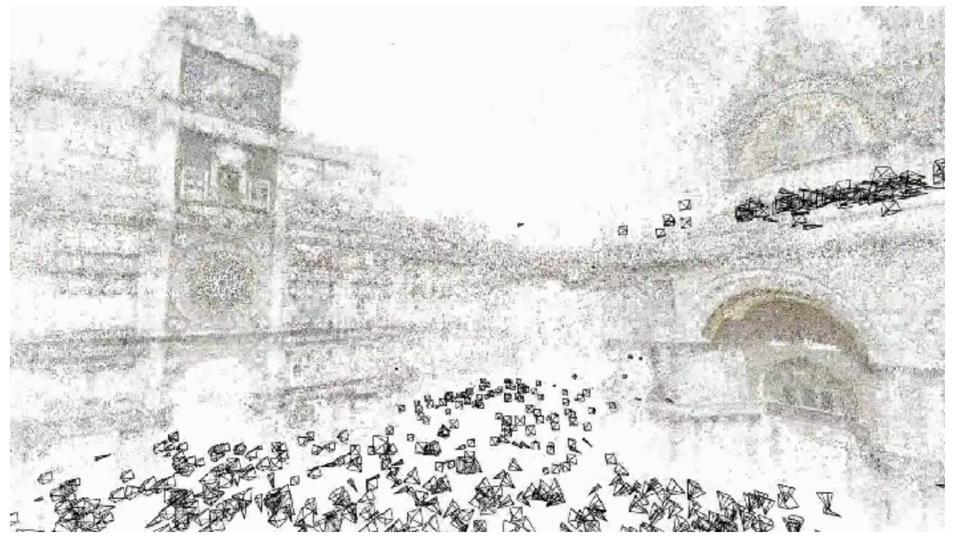
- Need boundary extraction — easier for meshes, harder for implicit representations and fractals



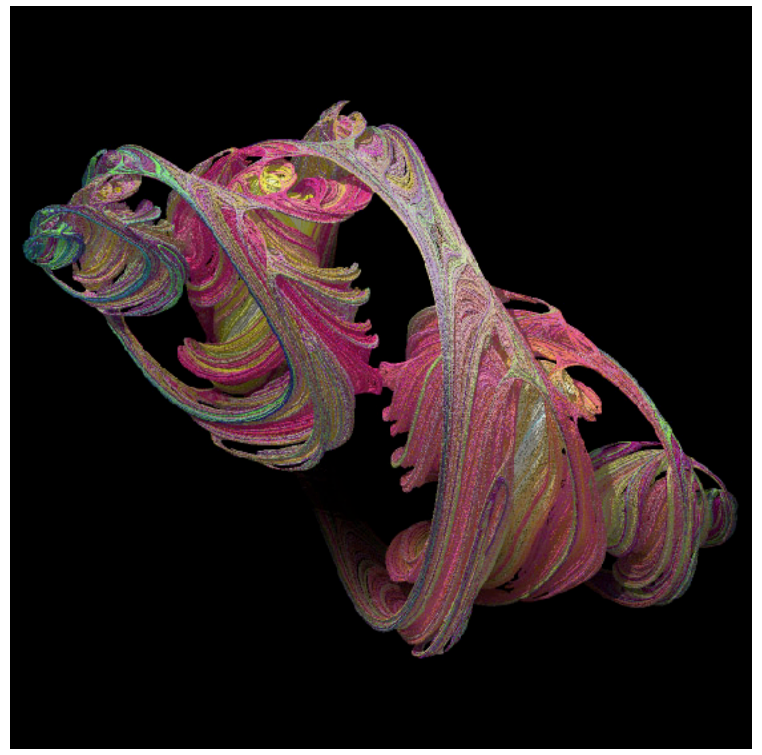
mesh



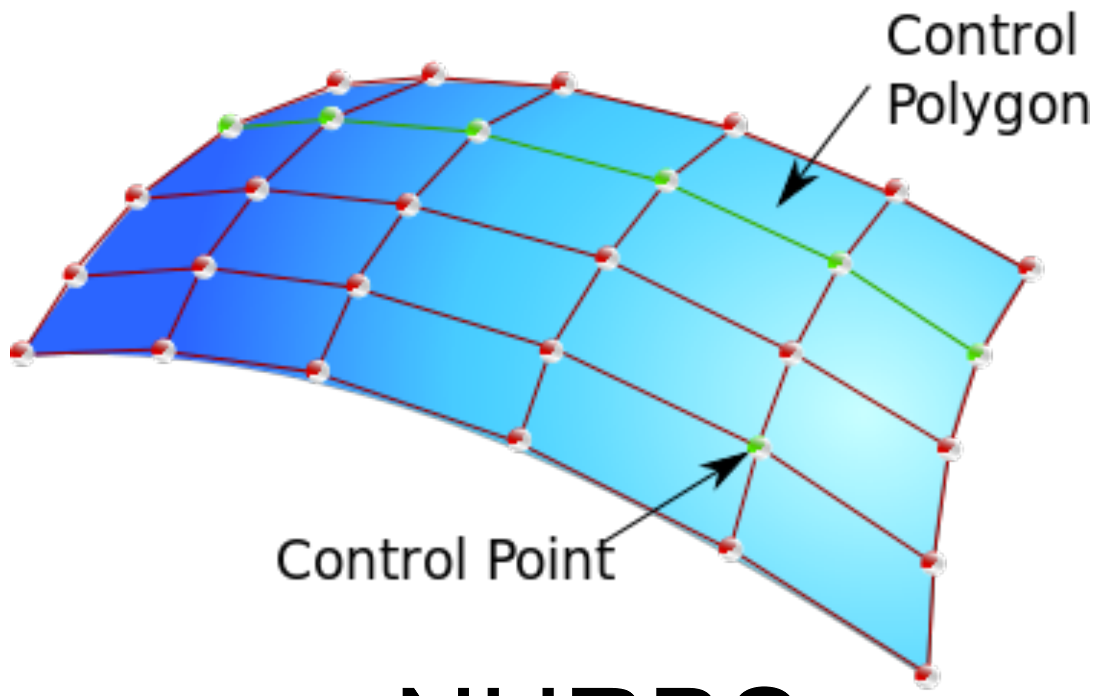
CSG



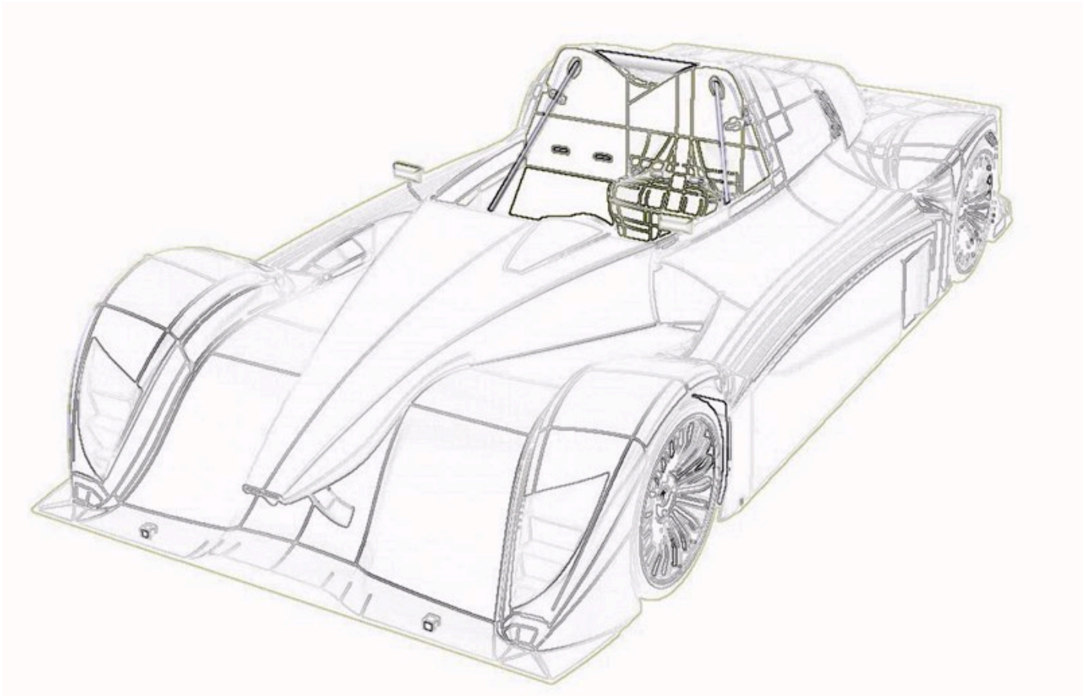
point cloud



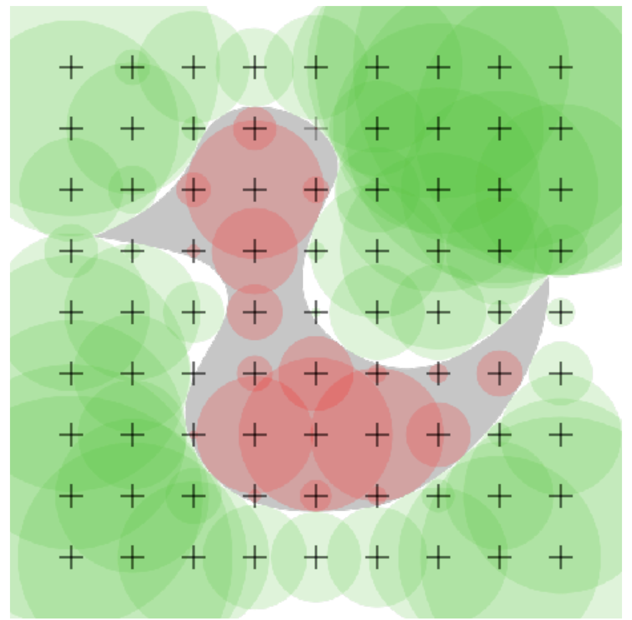
fractal



NURBS



B-rep

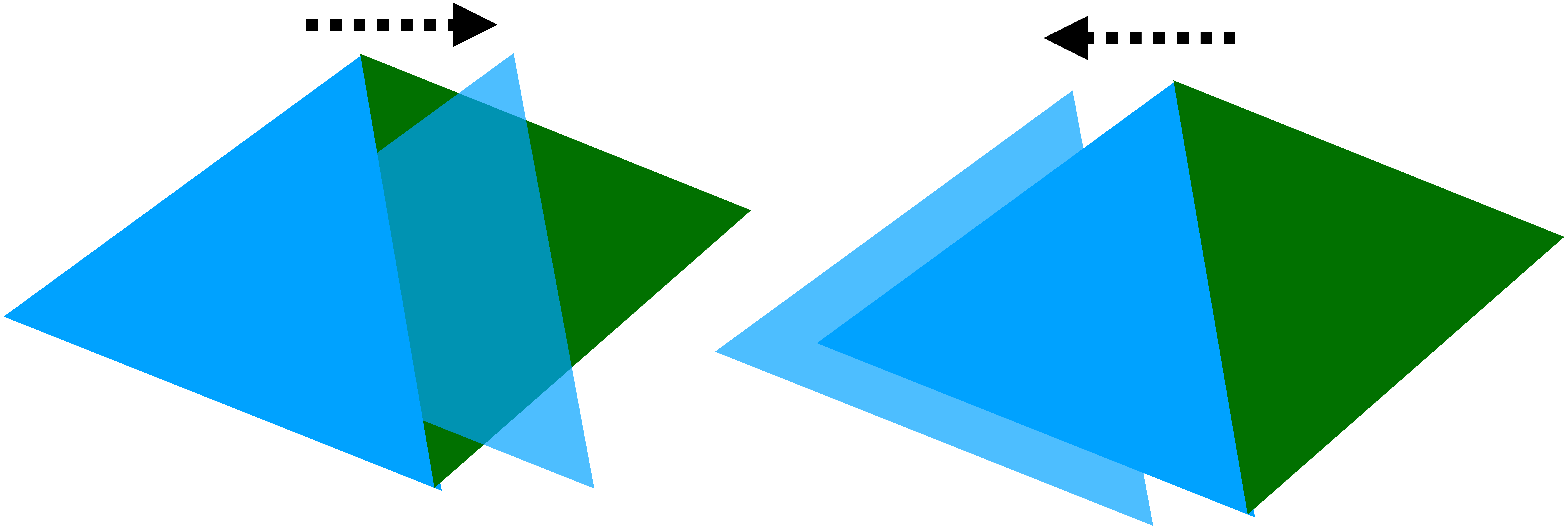


SDF

images courtesy of Carlson et al., Vladsinger, Agarwal et al., Pso, Solkoll, Zottie, Drummyfish

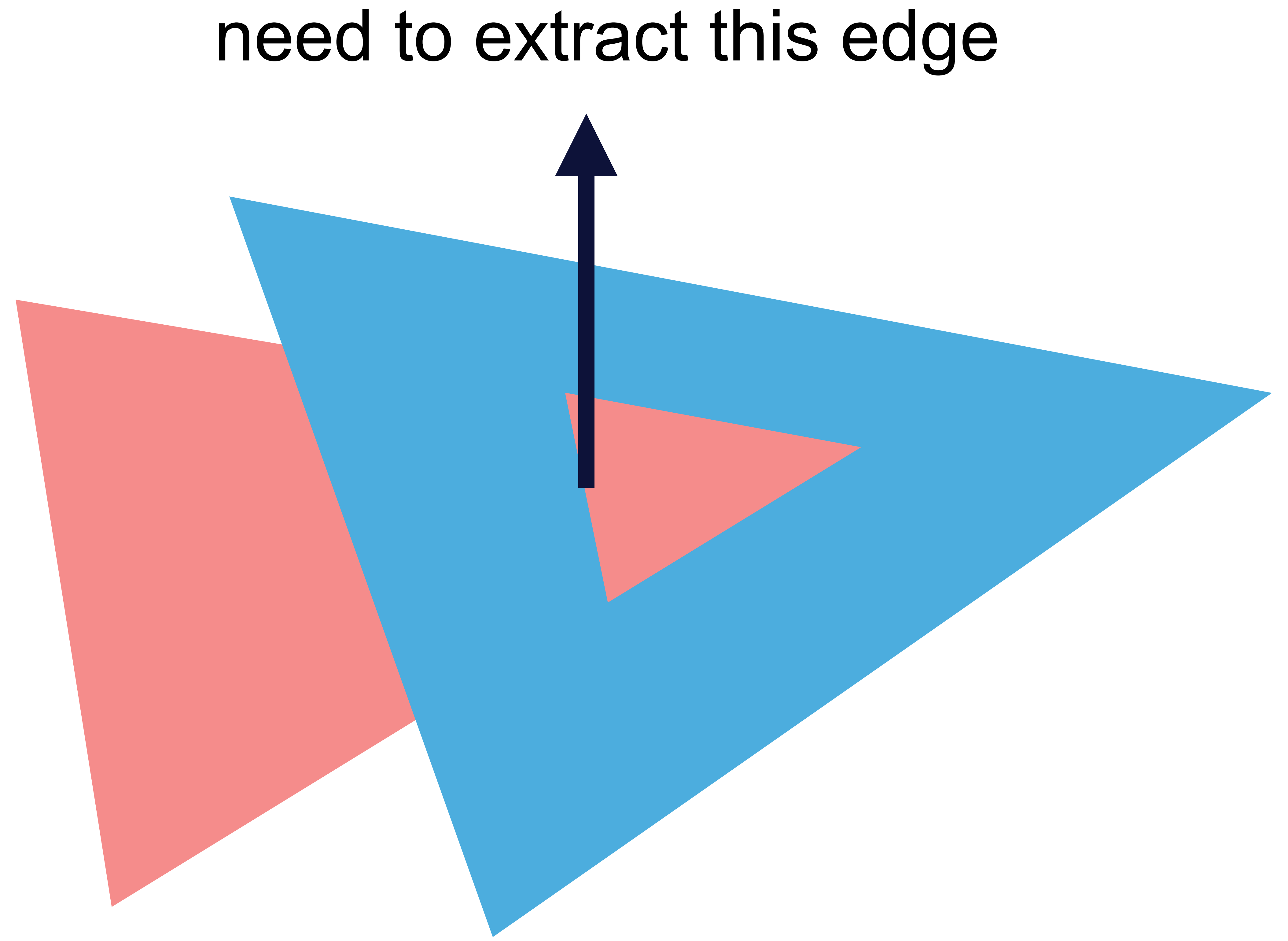
LIMITATIONS

- Non-differentiability of parallel edges of two separate triangles
  - can be resolved by applying a small perturbation to the vertices



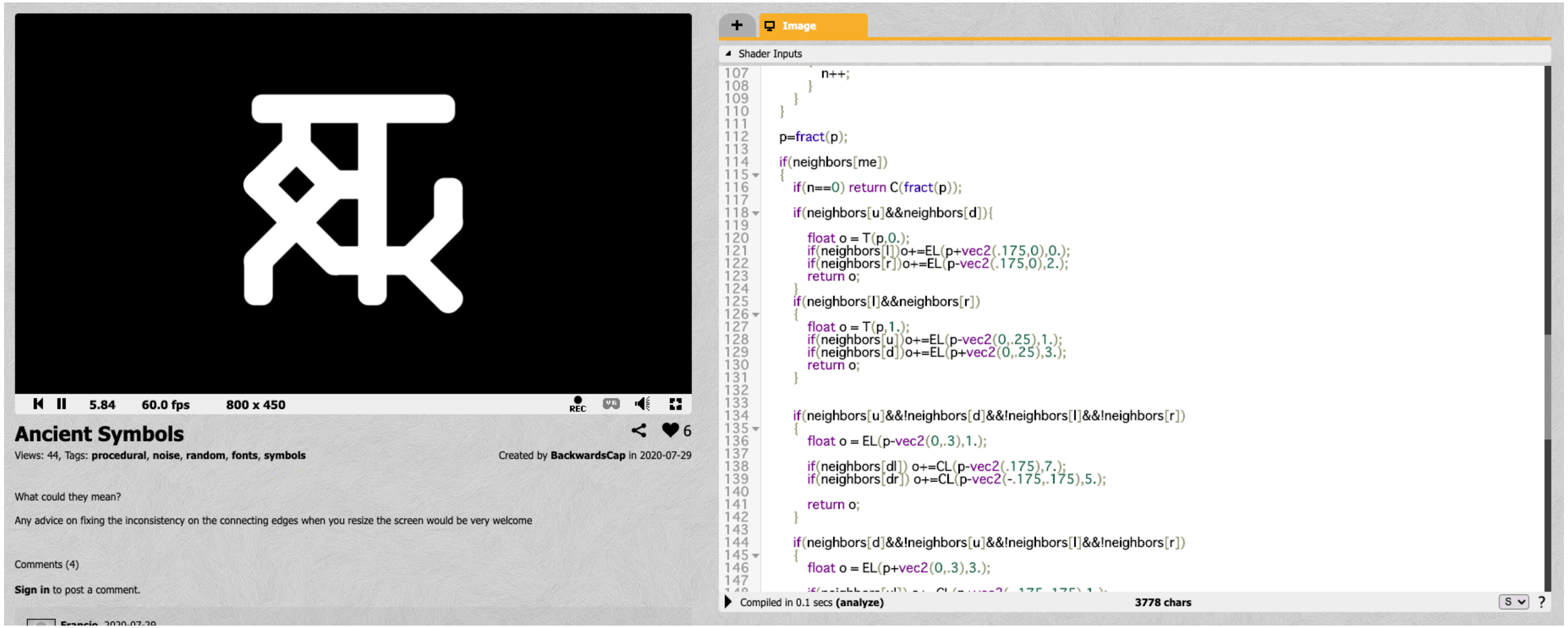
LIMITATIONS

- Non-differentiability of parallel edges of two separate triangles
  - can be resolved by applying a small perturbation to the vertices
- Interpenetration



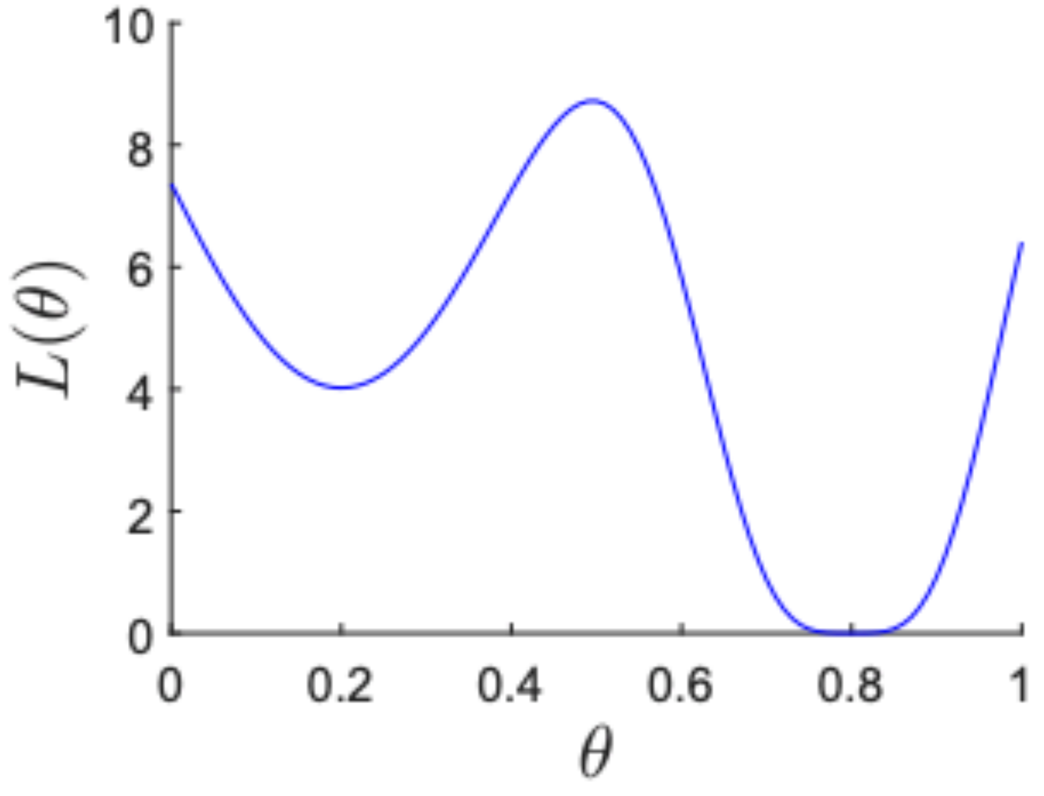
LIMITATIONS

- Non-differentiability of parallel edges of two separate triangles
  - can be resolved by applying a small perturbation to the vertices
- Interpenetration
- If/else conditions in procedural shaders (bitmap texture is 100% fine)

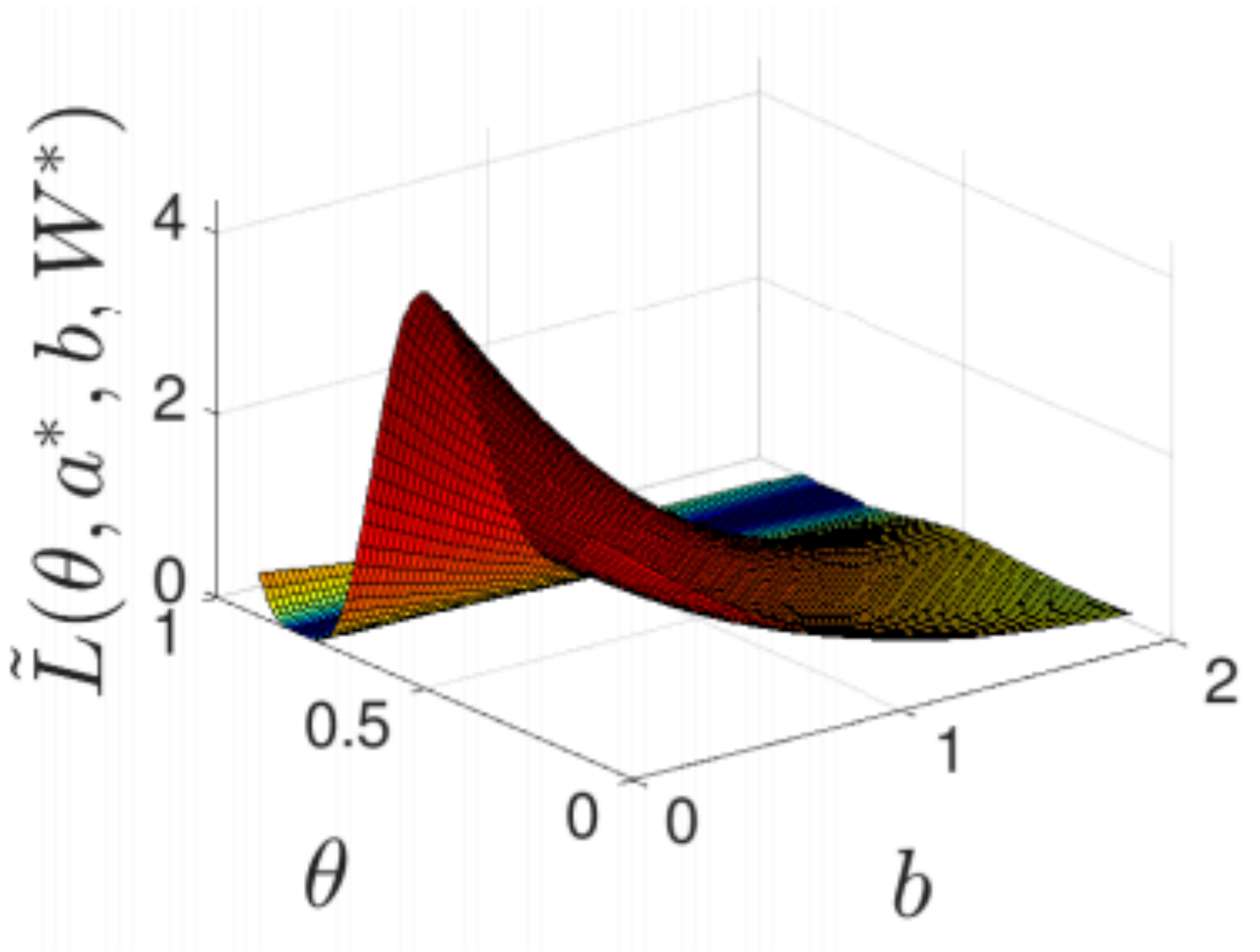


LIMITATIONS

- Non-differentiability of parallel edges of two separate triangles
  - can be resolved by applying a small perturbation to the vertices
- Interpenetration
- If/else conditions in procedural shaders (bitmap texture is 100% fine)
- Local minimum

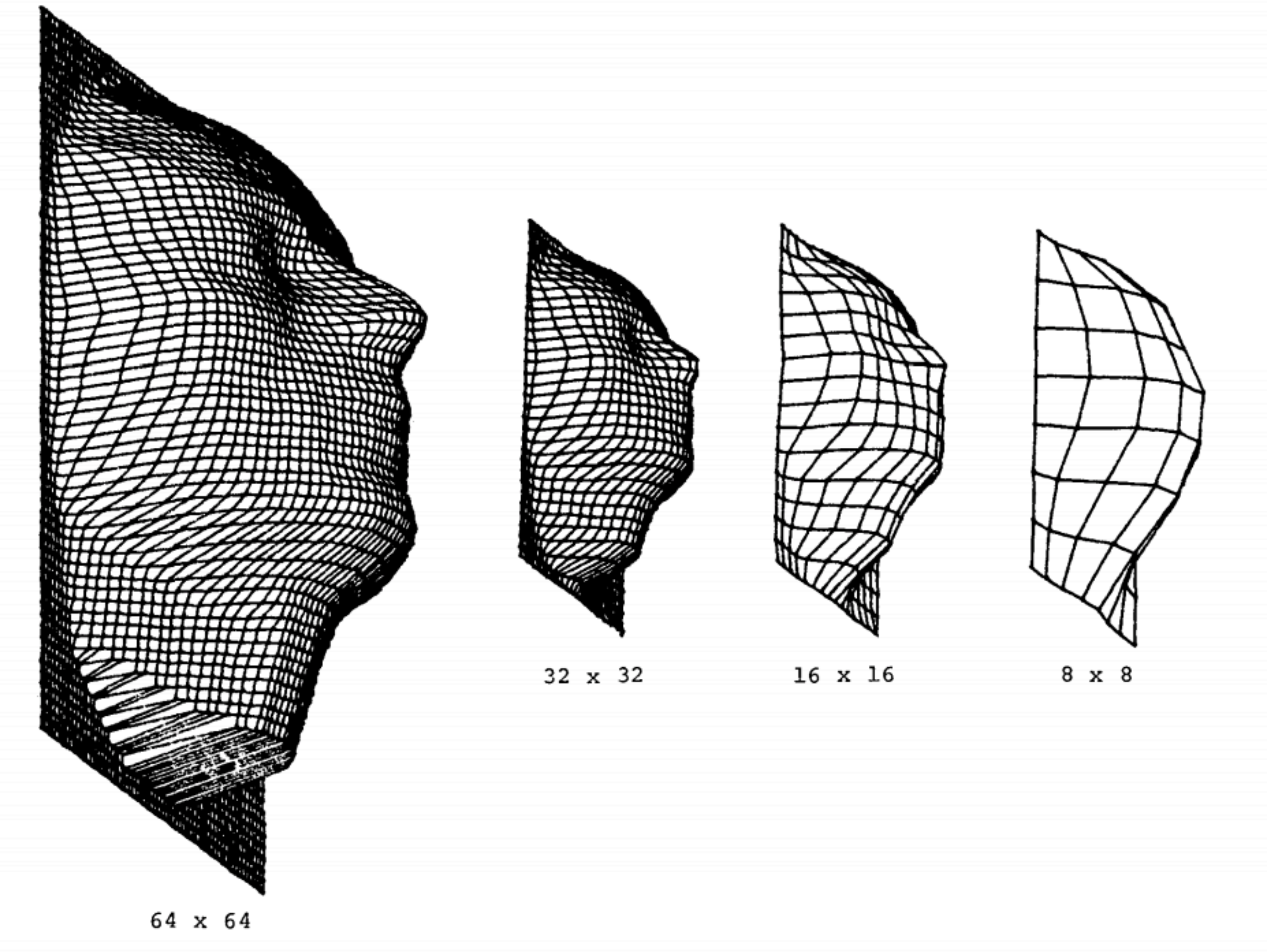


(a) original objective function L



(b) modified objective function $\tilde{L}$

Kawaguchi and Kaelbling 2019



William 1983

THEORY & ALGORITHMS

- Warm-up: differential irradiance

- Warm-up: differential irradiance

- Differentiable path tracing with edge sampling
- Differential radiative transfer

DIFFERENTIABLE RENDERING THEORY & ALGORITHMS

- Warm-up: differential irradiance

- Differentiable path tracing with edge sampling
- Differential radiative transfer

- Another way of dealing with discontinuities
- Radiative backpropagation

DIFFERENTIABLE RENDERING THEORY & ALGORITHMS

- Warm-up: differential irradiance

- Differentiable path tracing with edge sampling
- Differential radiative transfer

- Another way of dealing with discontinuities
- Radiative backpropagation

- Path-space differentiable rendering

WARM-UP: DIFFERENTIAL IRRADIANCE

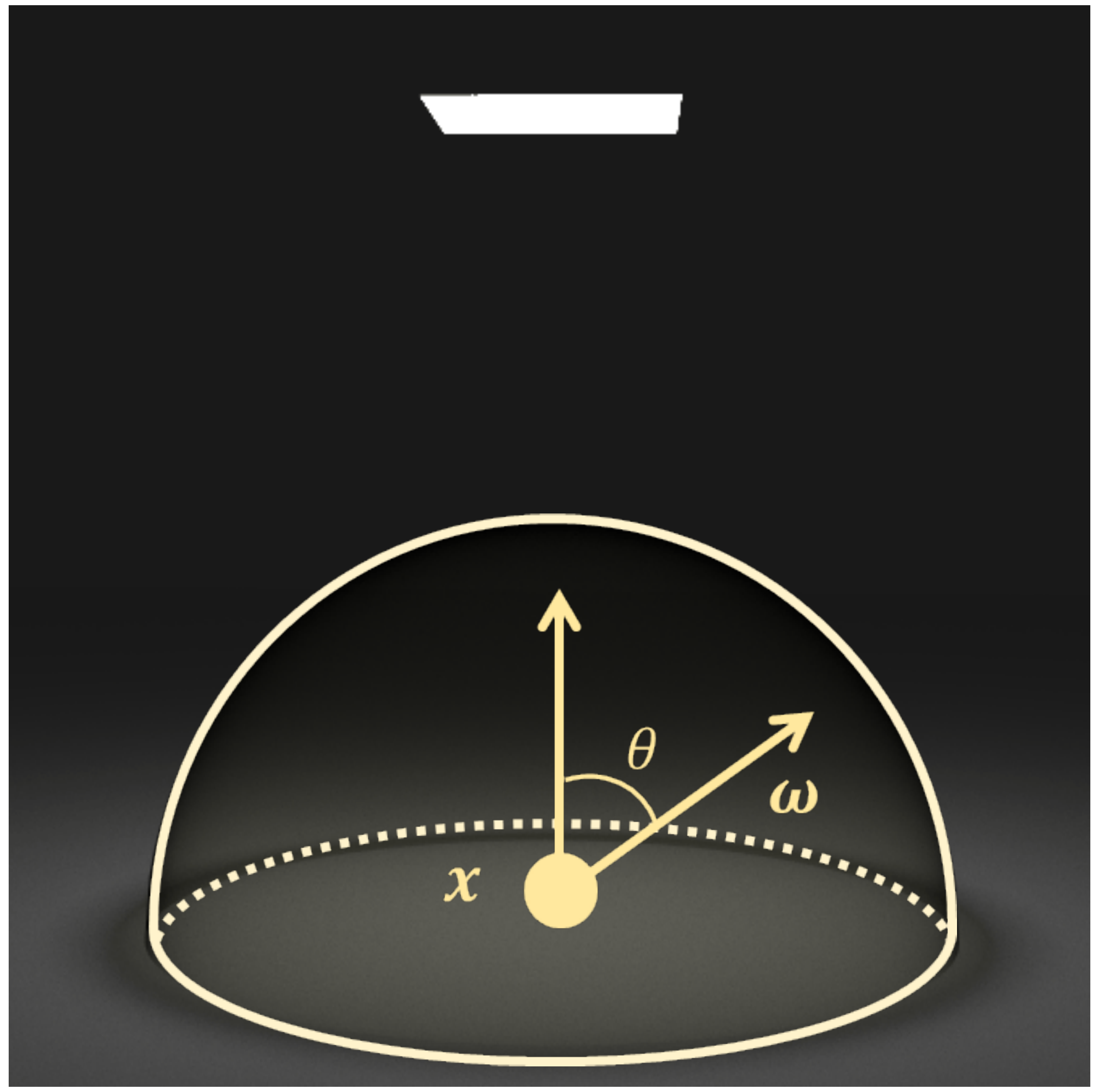
Irradiance at $\mathbf{x}$: $$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

Irradiance at $\mathbf{x}$: $$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

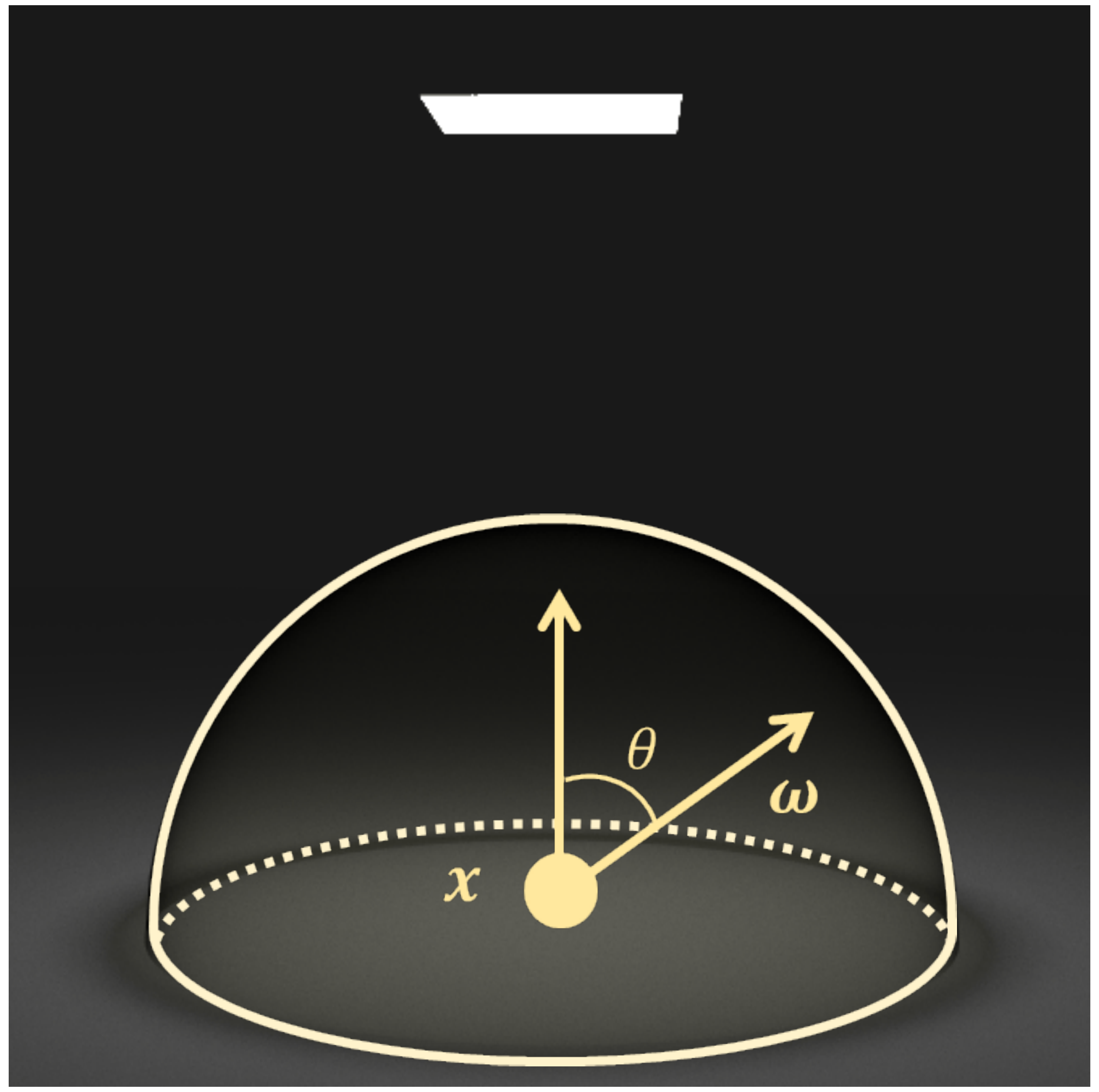
π: emitter size



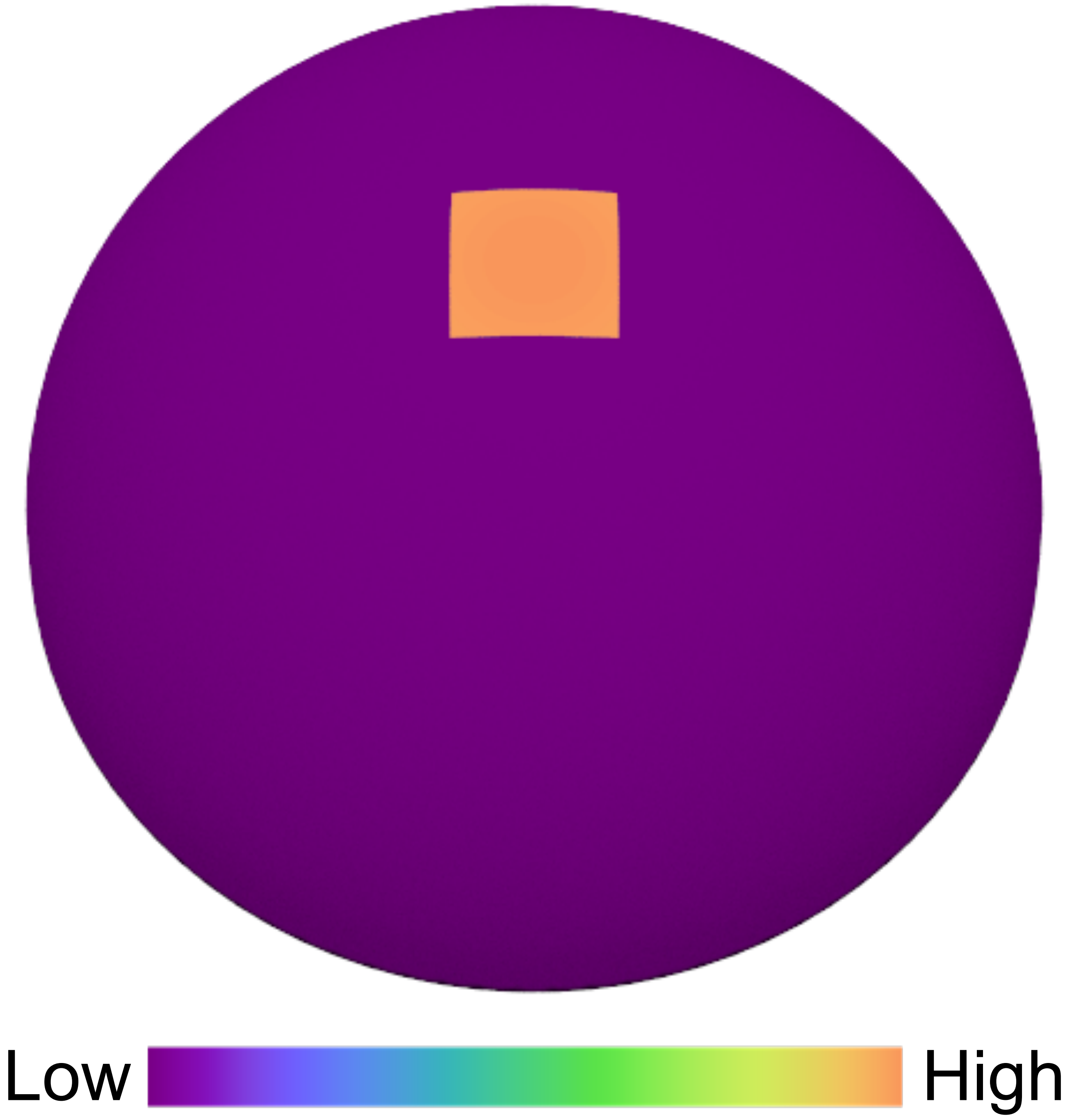
Irradiance at $\mathbf{x}$: $$E = \int_{\mathbb{H}^2} \overbrace{L_i(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, d\sigma(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

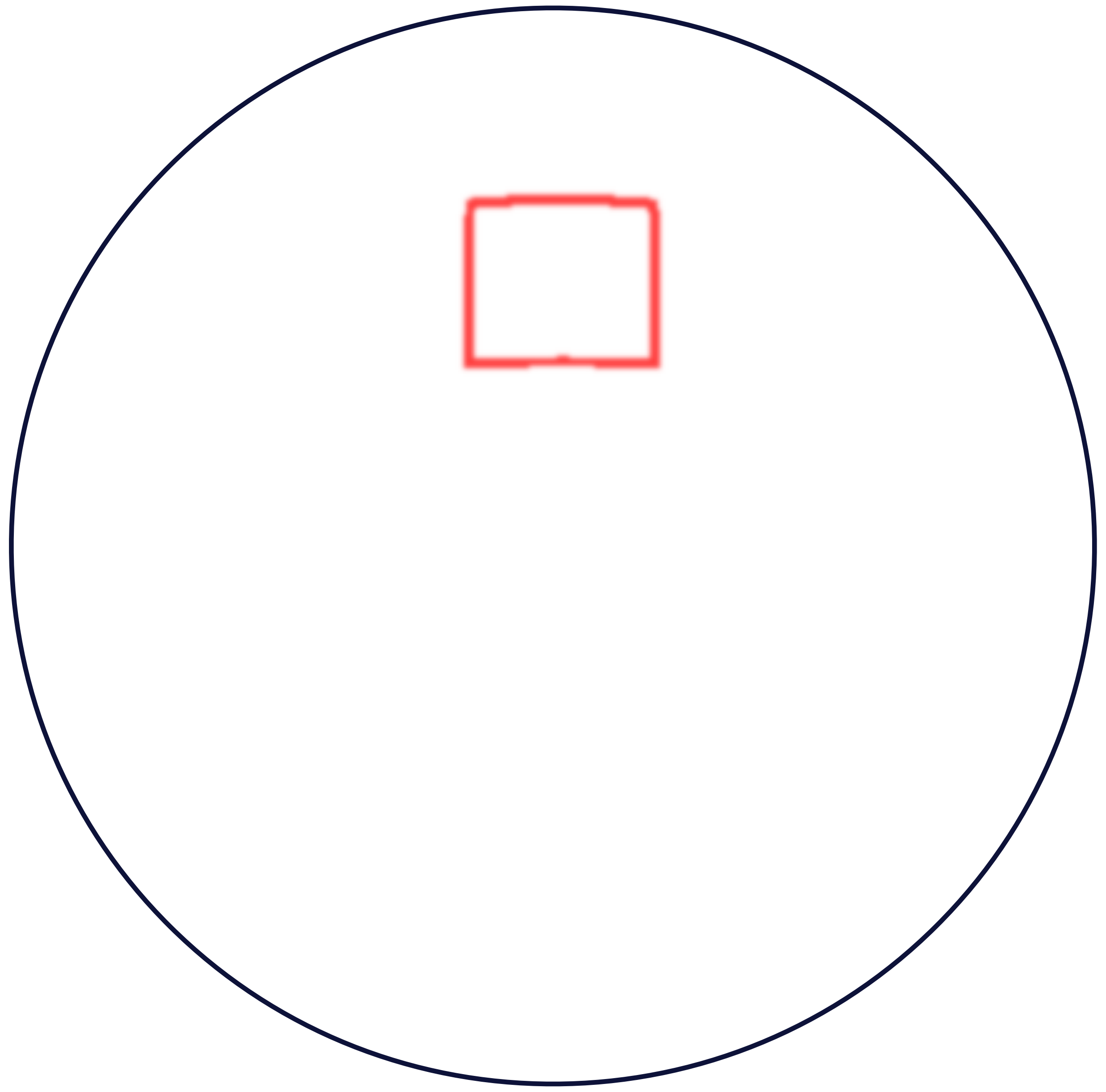
π: emitter size



$f_E(\omega)$



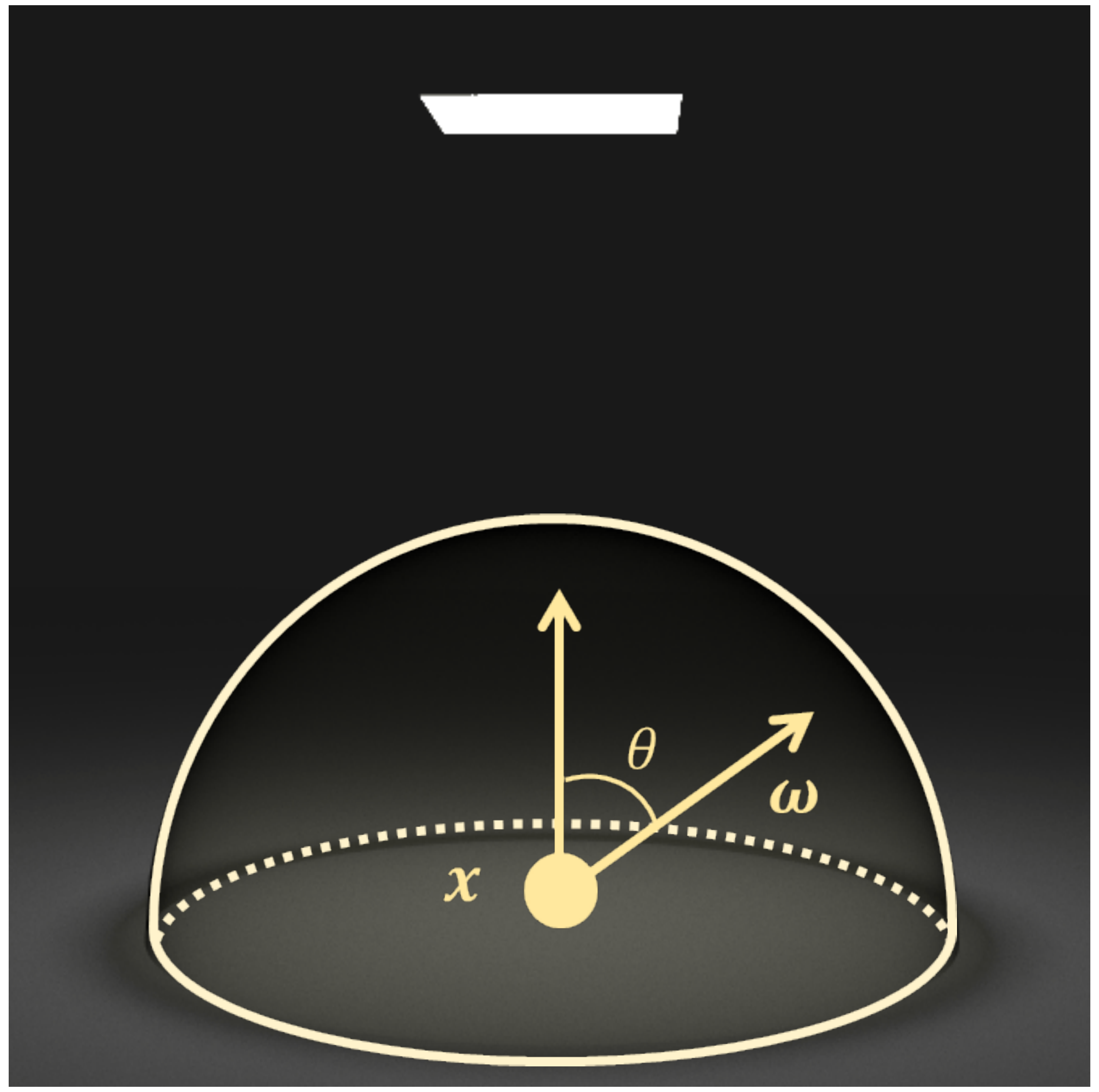
$\partial \mathbb{H}^2$



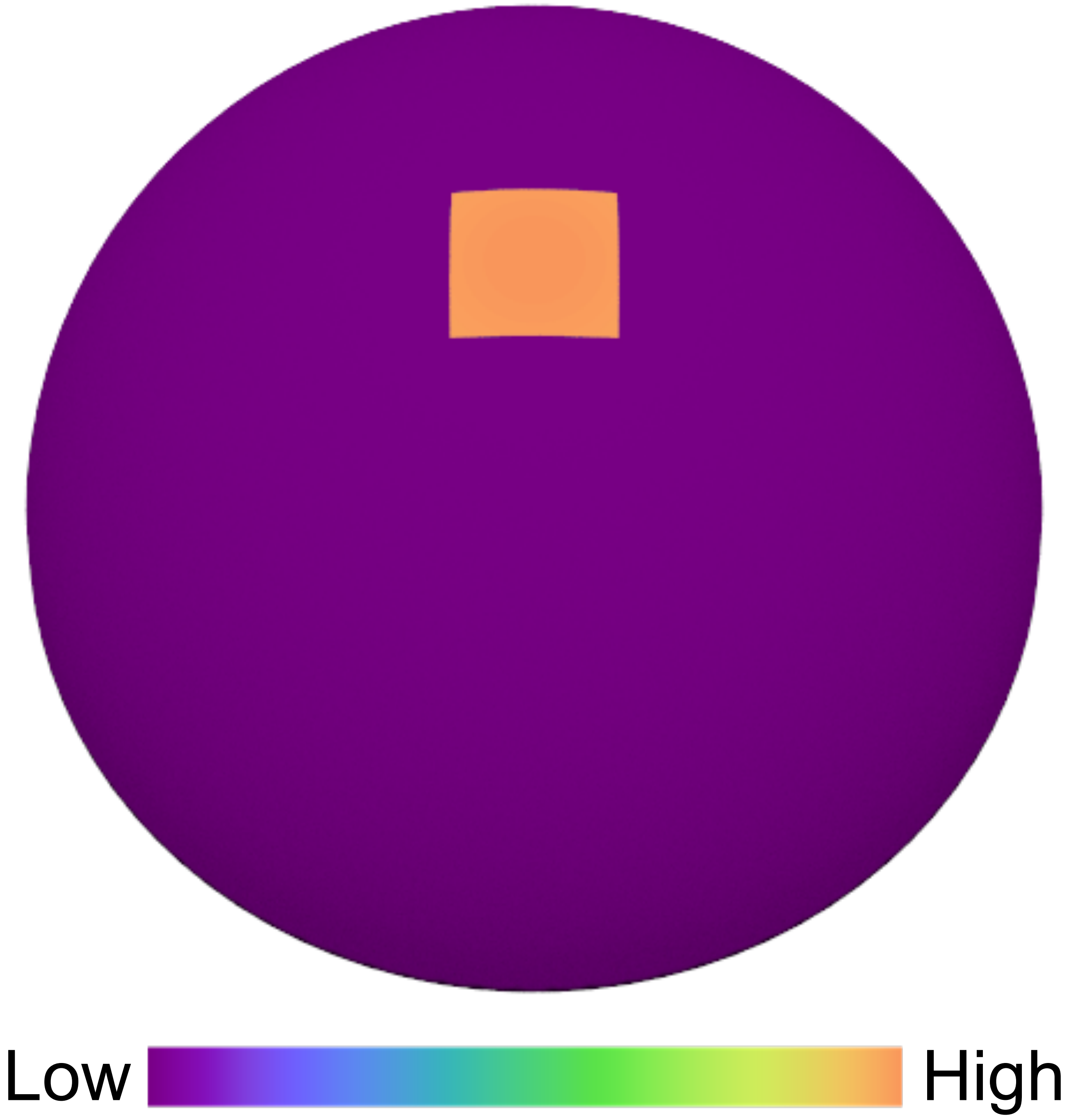
Irradiance at $\mathbf{x}$: $$E = \int_{\mathbb{H}^2} \overbrace{L_i(\omega)\,\cos\theta}^{f_E(\omega)}\, d\sigma(\omega)$$

WARM-UP: DIFFERENTIAL IRRADIANCE

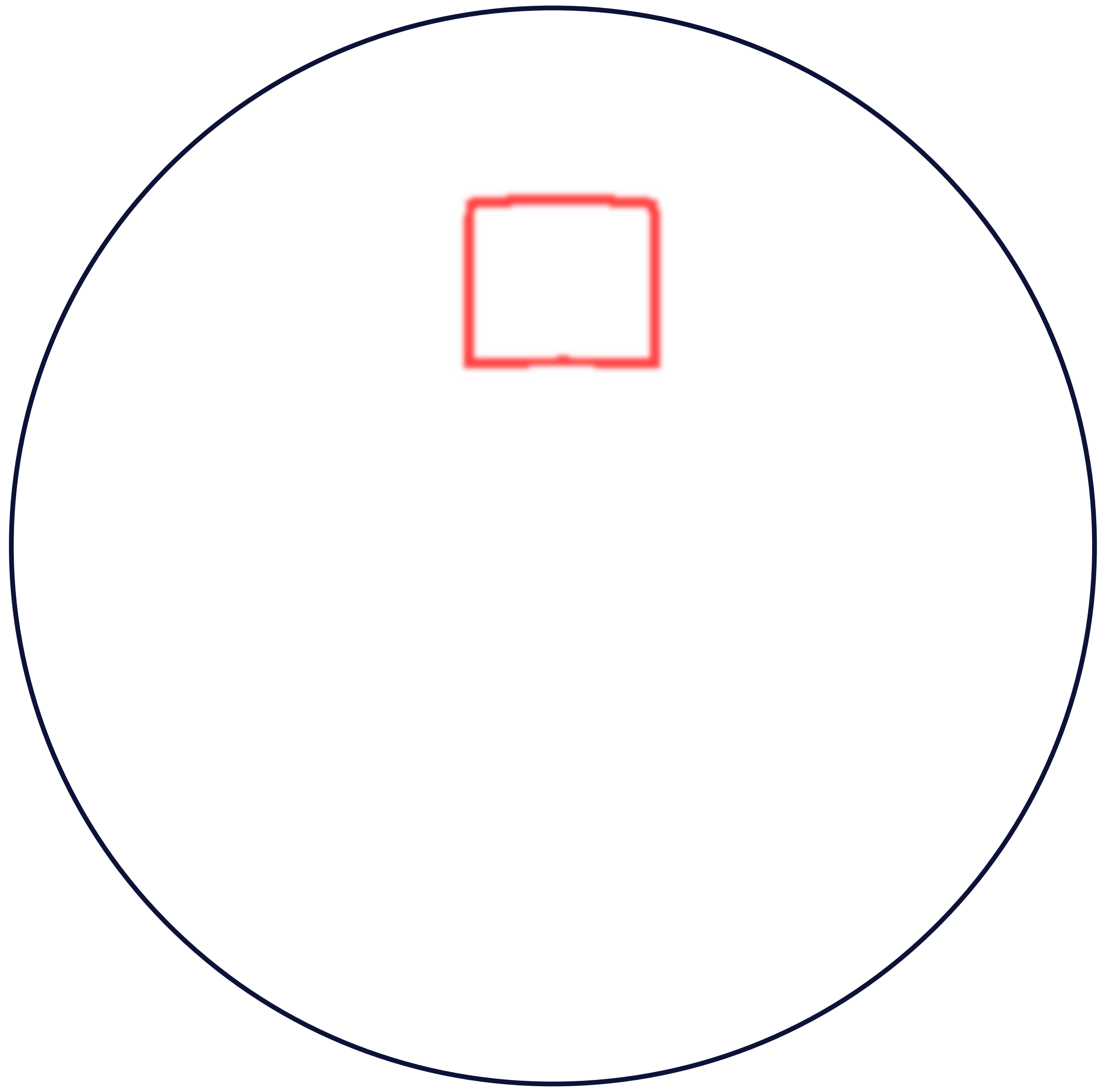
π: emitter size



$f_E(\boldsymbol{\omega})$



$\partial\mathbb{H}^2$



$$E = \int_{\mathbb{H}^2} \overbrace{L_{\mathrm{i}}(\boldsymbol{\omega})\,\cos\theta}^{f_E(\boldsymbol{\omega})}\,\mathrm{d}\sigma(\boldsymbol{\omega})$$

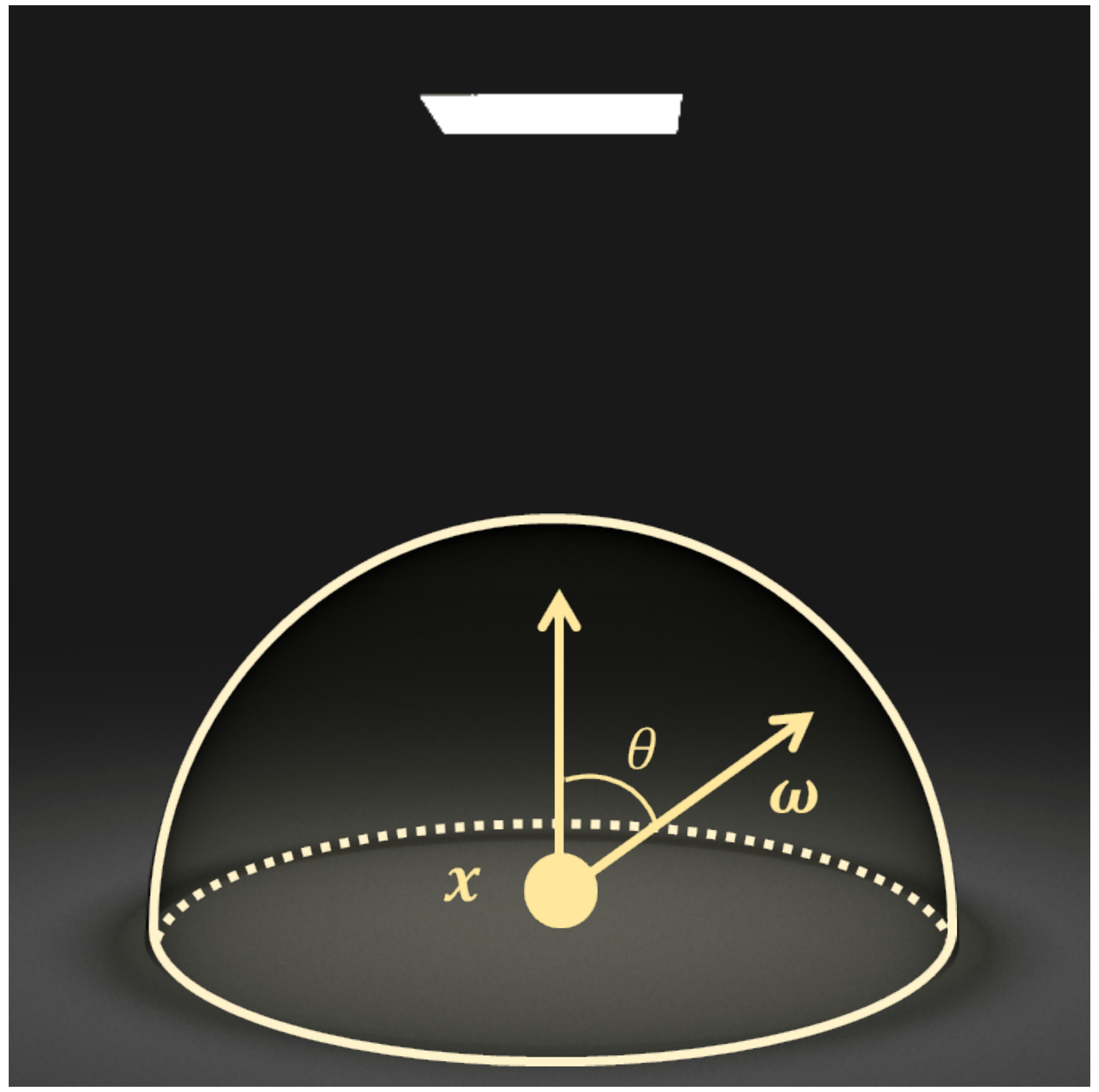
Reynolds ⇒

Interior integral | Boundary integral

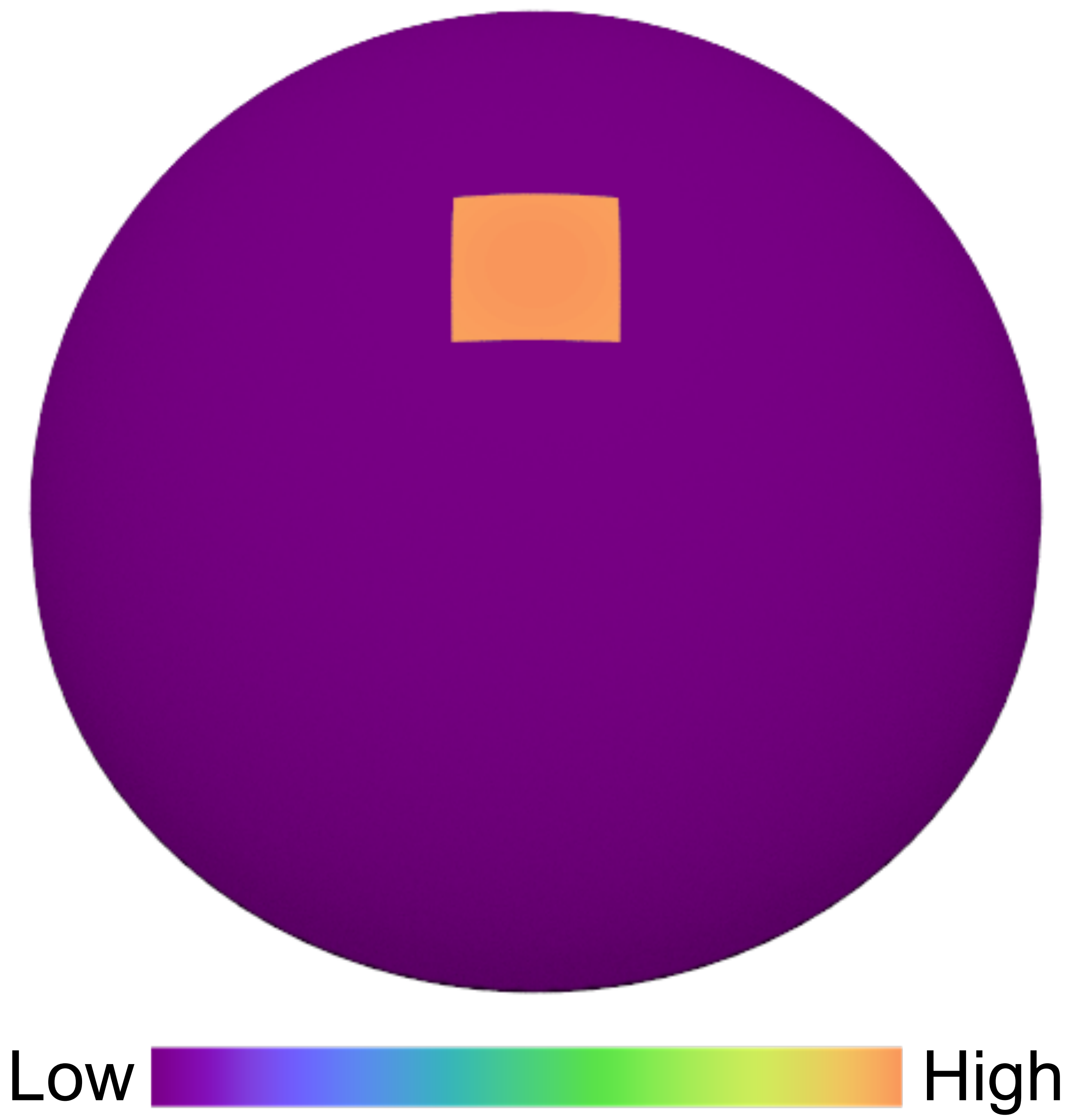
$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega})\,\mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\boldsymbol{\omega})\,\Delta f_E(\boldsymbol{\omega})\,\mathrm{d}\ell(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

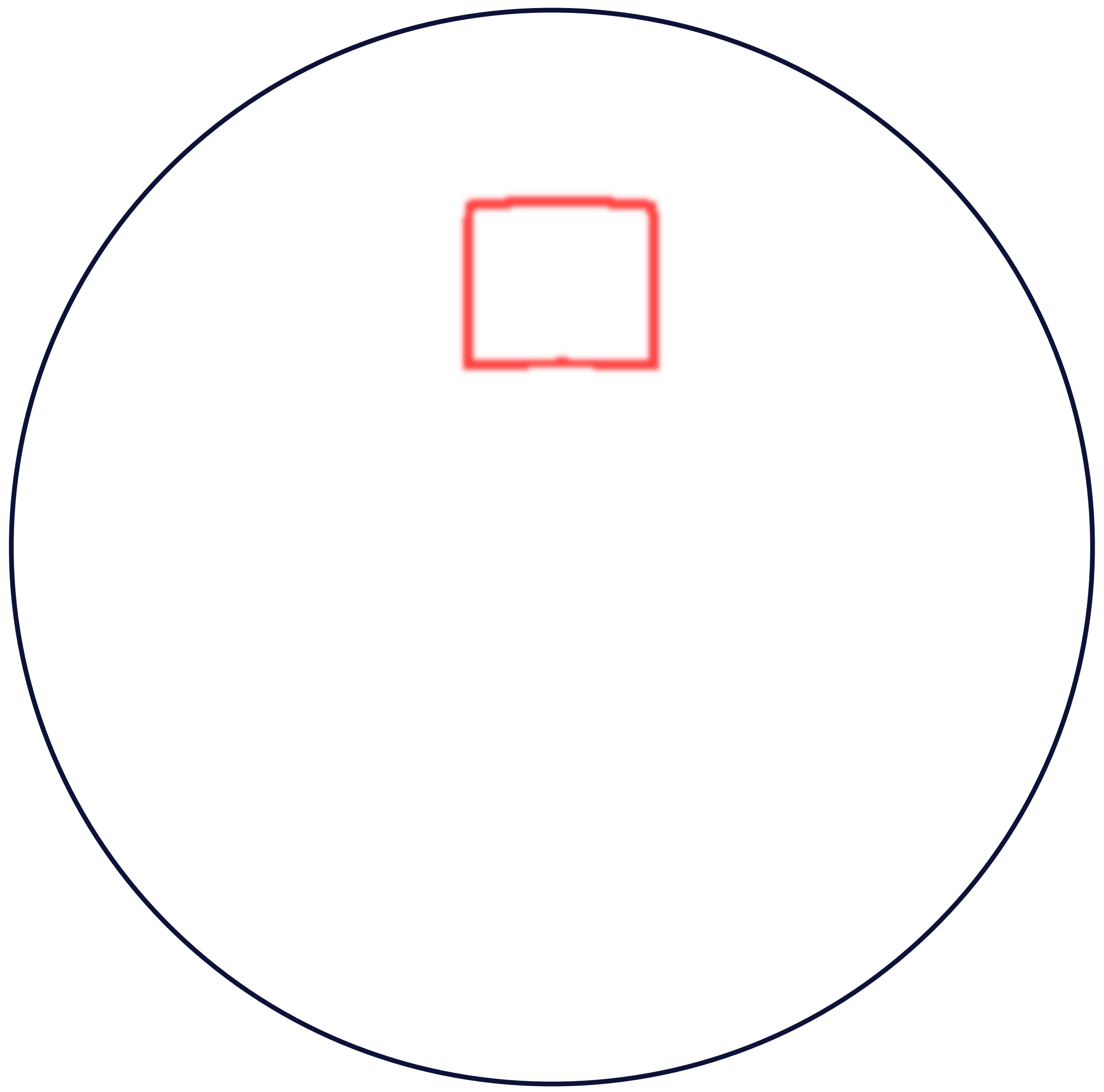
π: emitter size



$f_E(\omega)$



$\partial\mathbb{H}^2$

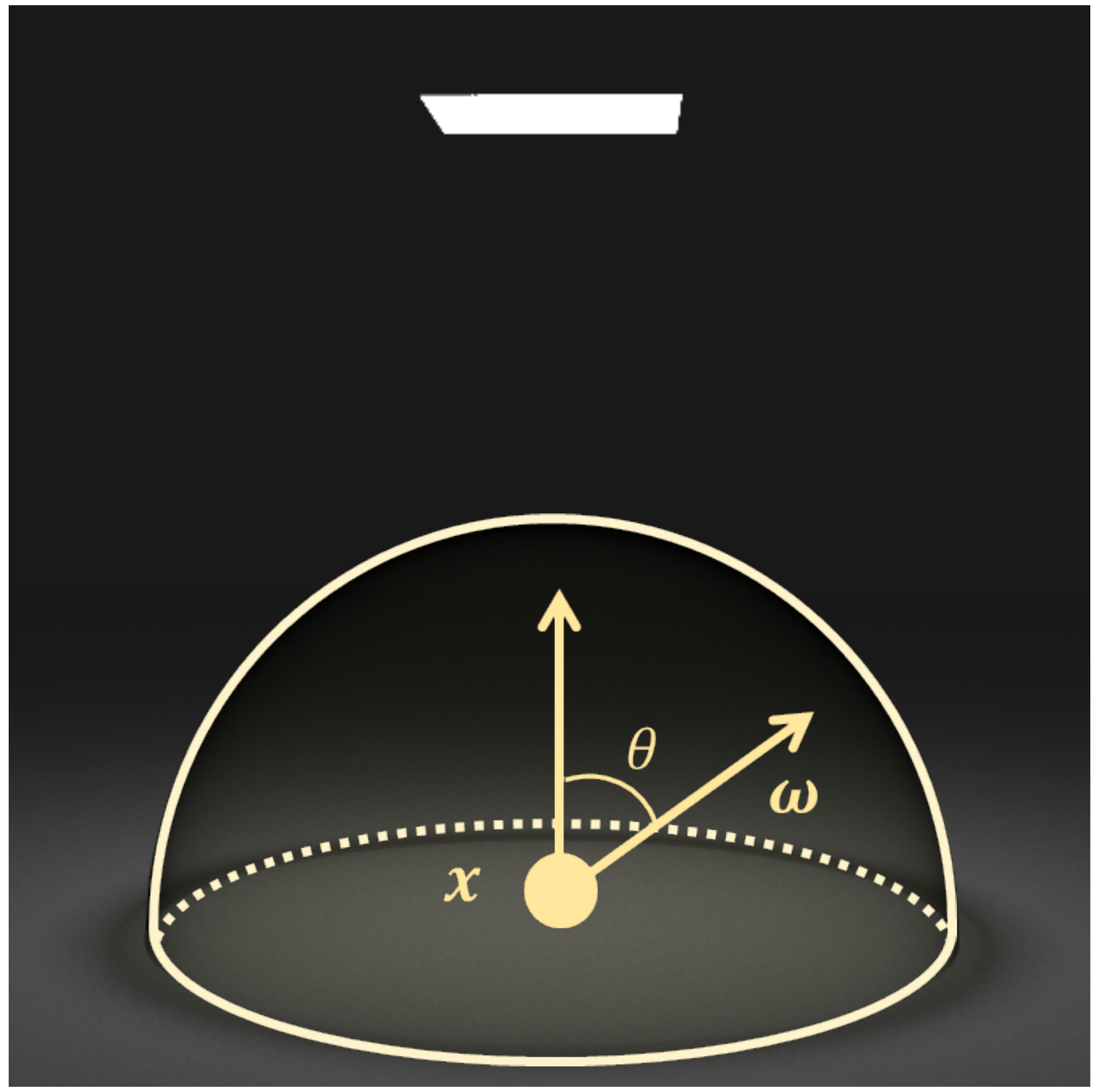


Interior integral = 0

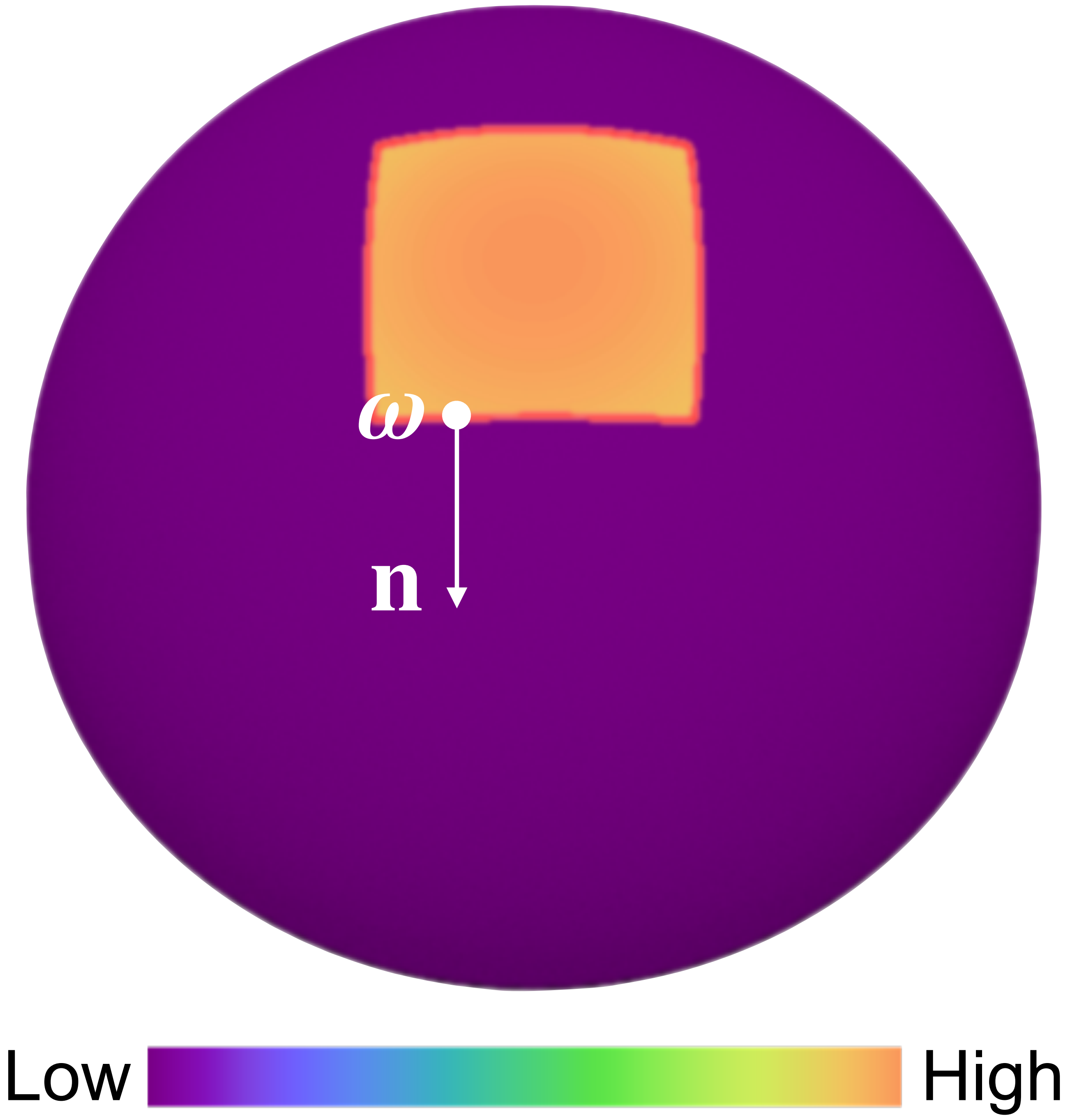
$$E = \int_{\mathbb{H}^2} \overbrace{L_i(\omega)\cos\theta}^{f_E(\omega)} \, d\sigma(\omega) \quad \xrightarrow{\text{Reynolds}} \quad \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E}{d\pi}(\omega)\, d\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega)\, \Delta f_E(\omega)\, d\ell(\omega)$$

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\omega)$



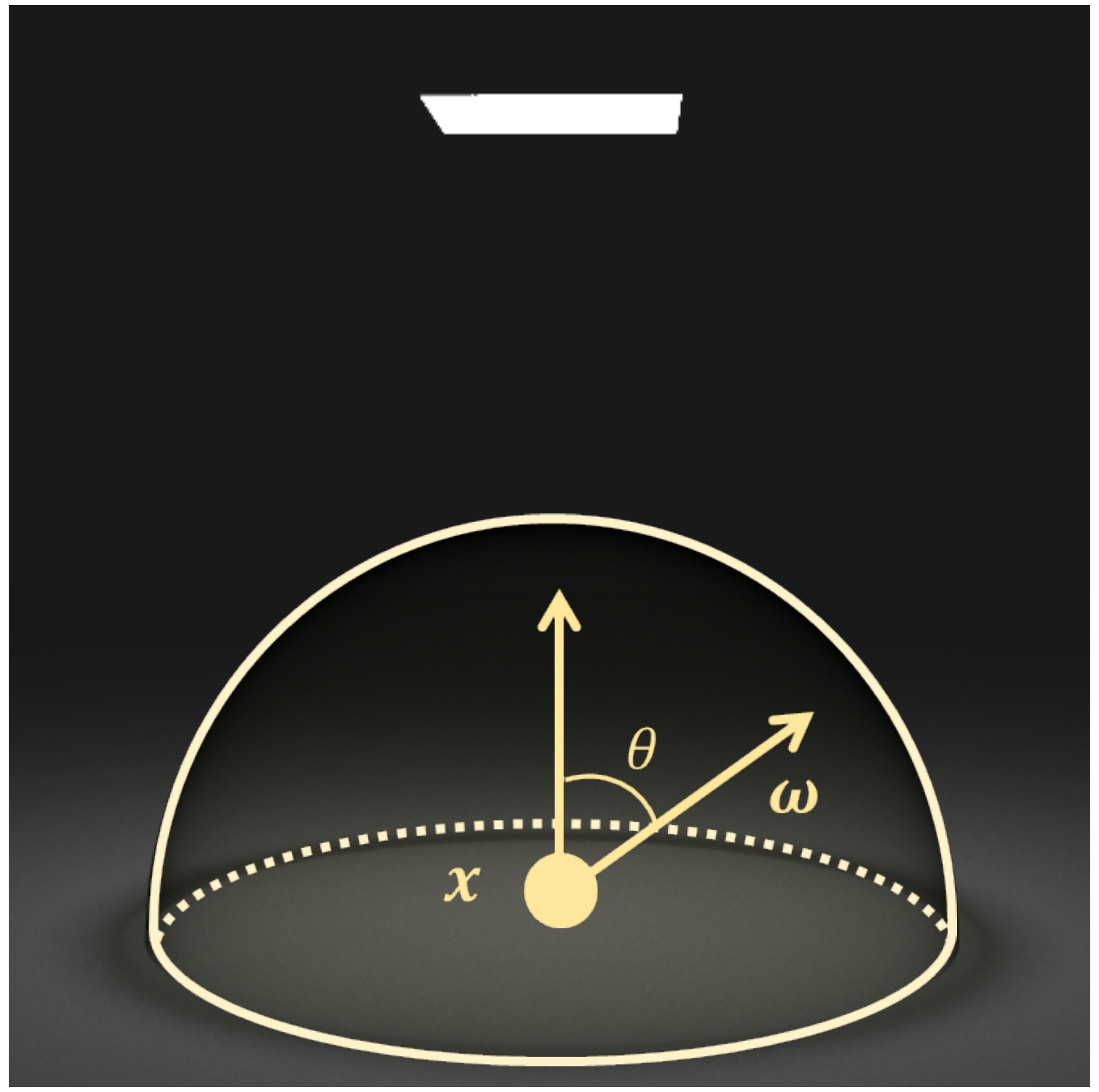
$$E = \int_{\mathbb{H}^2} \overbrace{L_i(\omega)\,\cos\theta}^{f_E(\omega)}\,\mathrm{d}\sigma(\omega)$$

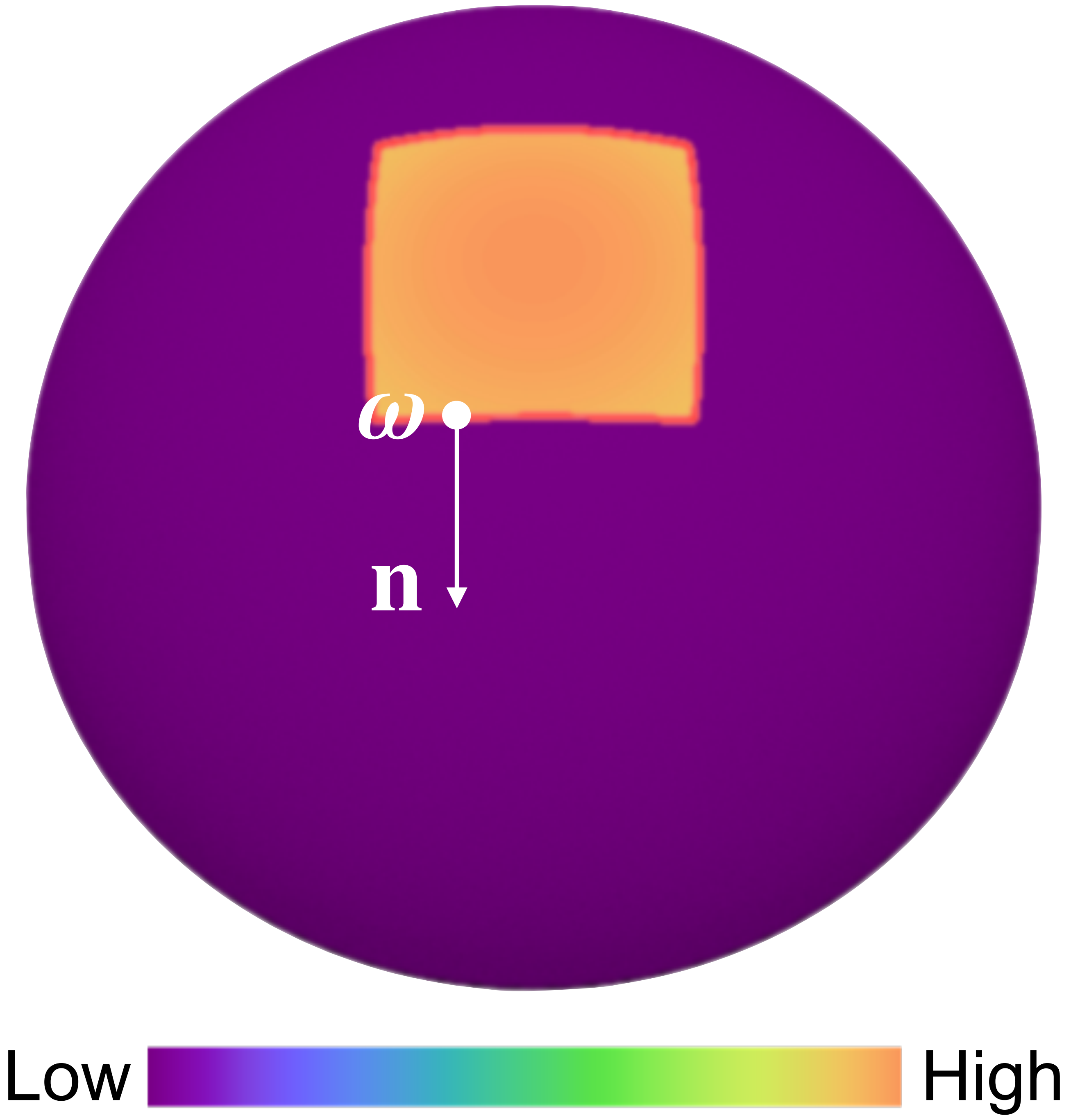
Reynolds ⇒

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\omega)\,\mathrm{d}\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega)\,\Delta f_E(\omega)\,\mathrm{d}\ell(\omega)$$

Boundary integral

WARM-UP: DIFFERENTIAL IRRADIANCE





$$E = \int_{\mathbb{H}^2} \overbrace{L_i(\omega)\, \cos\theta}^{f_E(\omega)}\, \mathrm{d}\sigma(\omega)$$

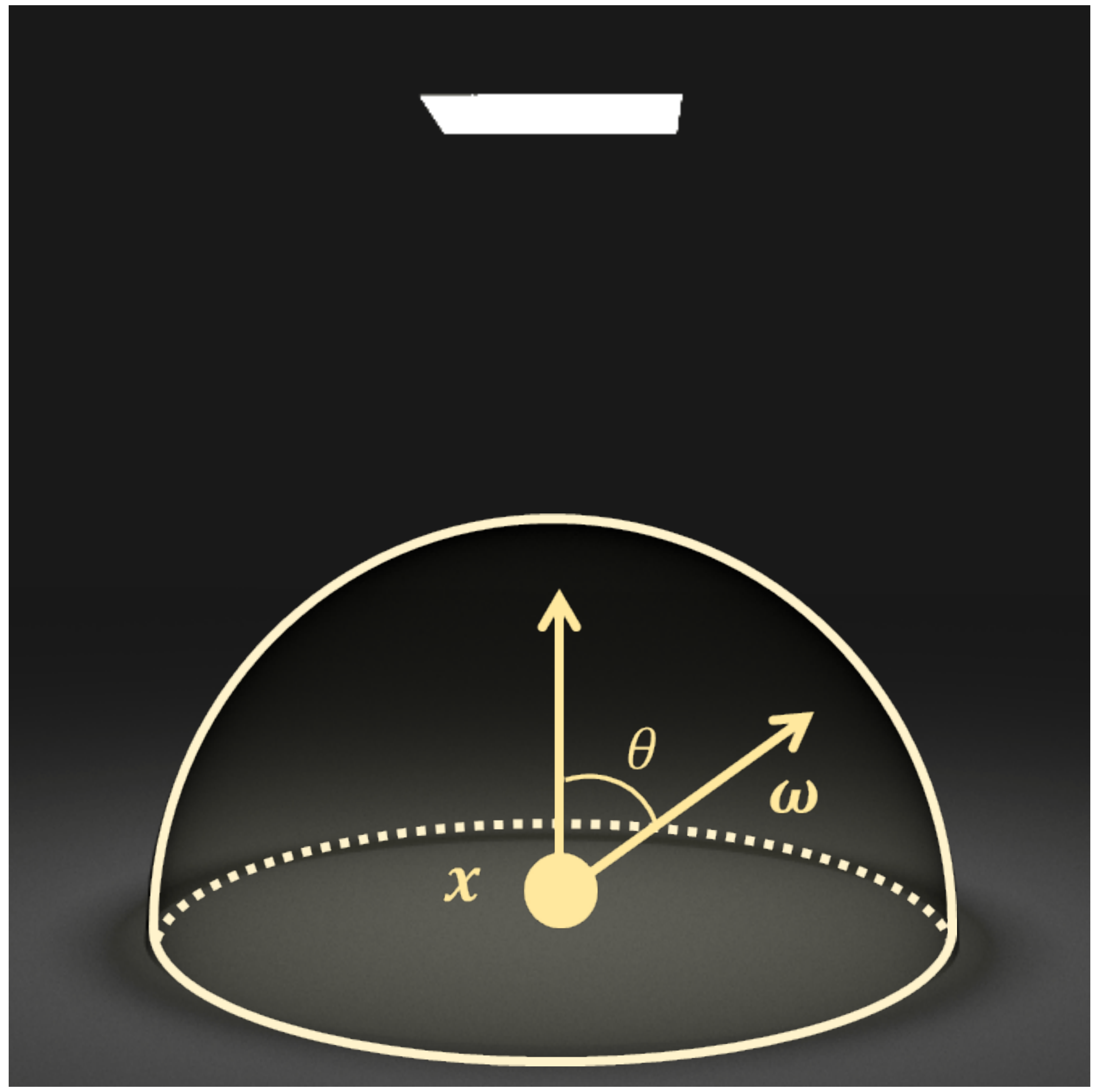
Reynolds ⇒

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\omega)\, \mathrm{d}\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega)\, \Delta f_E(\omega)\, \mathrm{d}\ell(\omega)$$

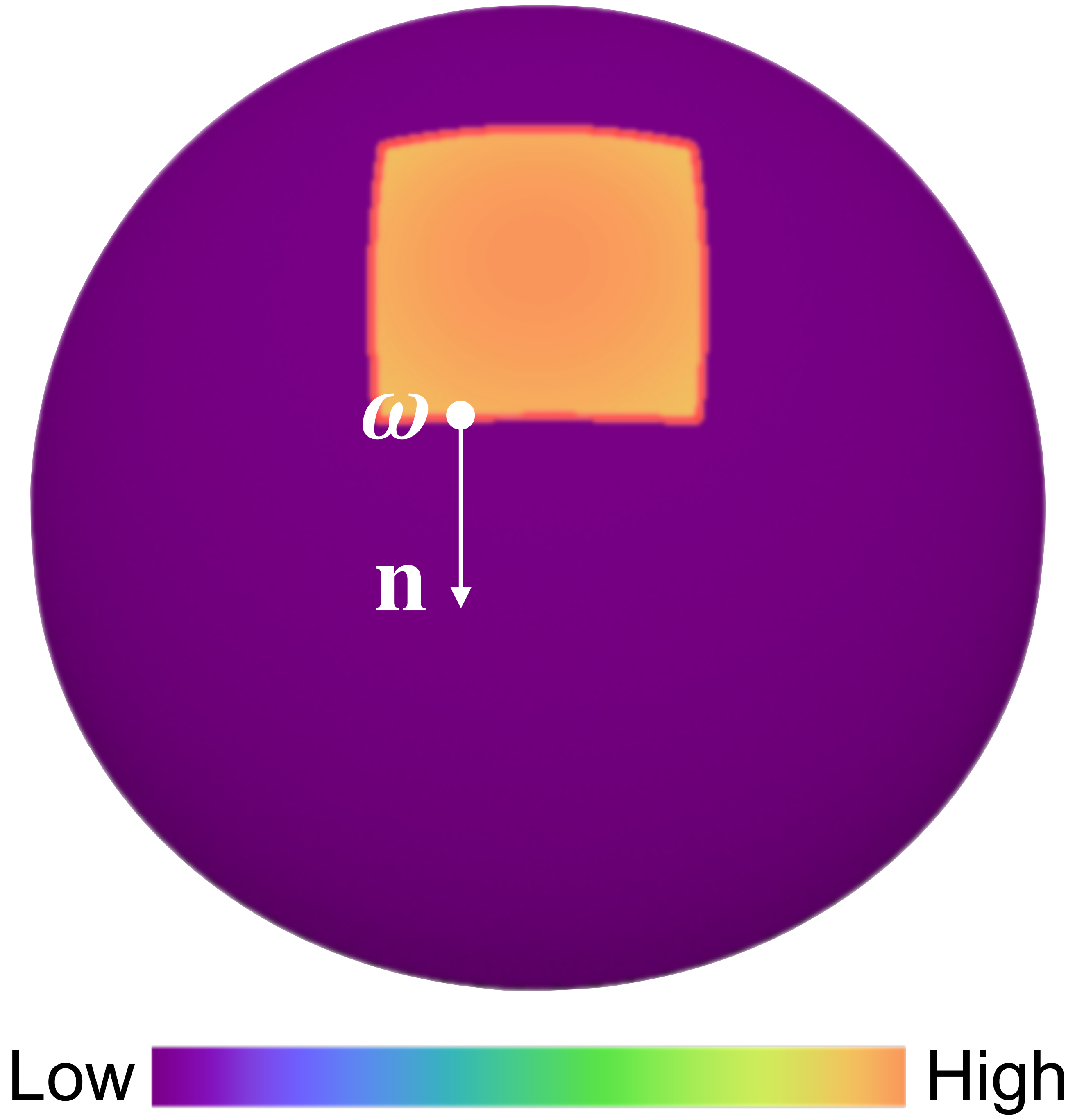
Boundary integral

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\boldsymbol{\omega})$



Scalar normal "velocity" of $\boldsymbol{\omega}$

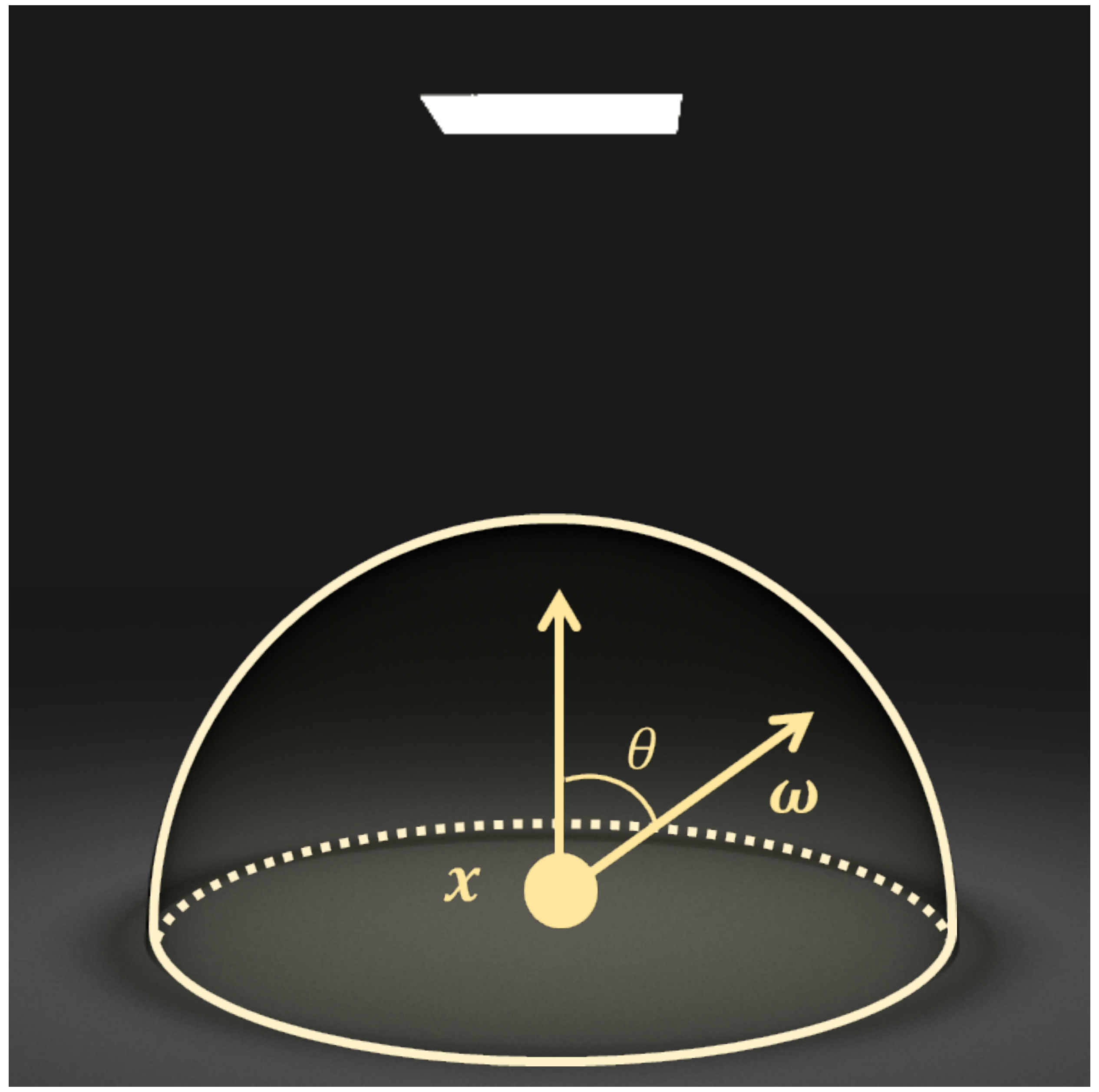
$$V_{\partial \mathbb{H}^2}(\boldsymbol{\omega}) = \left\langle \mathbf{n}(\boldsymbol{\omega}), \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\pi} \right\rangle$$

Boundary integral

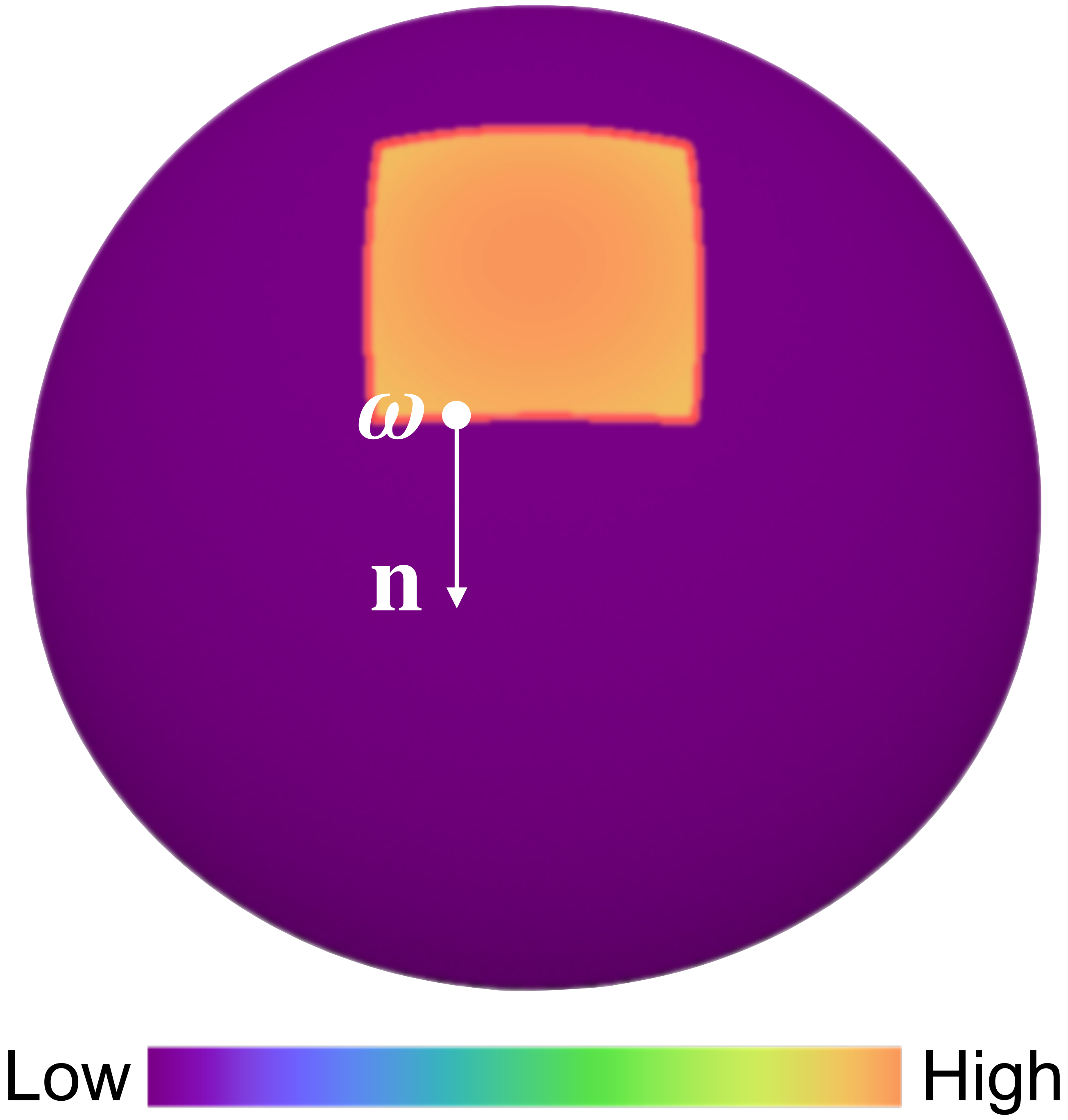
$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega}) \quad \xrightarrow{\text{Reynolds}} \quad \frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega}) \, \mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\boldsymbol{\omega}) \, \Delta f_E(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\boldsymbol{\omega})$



Scalar normal "velocity" of $\boldsymbol{\omega}$

$$V_{\partial \mathbb{H}^2}(\boldsymbol{\omega}) = \left\langle \mathbf{n}(\boldsymbol{\omega}), \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\pi} \right\rangle$$

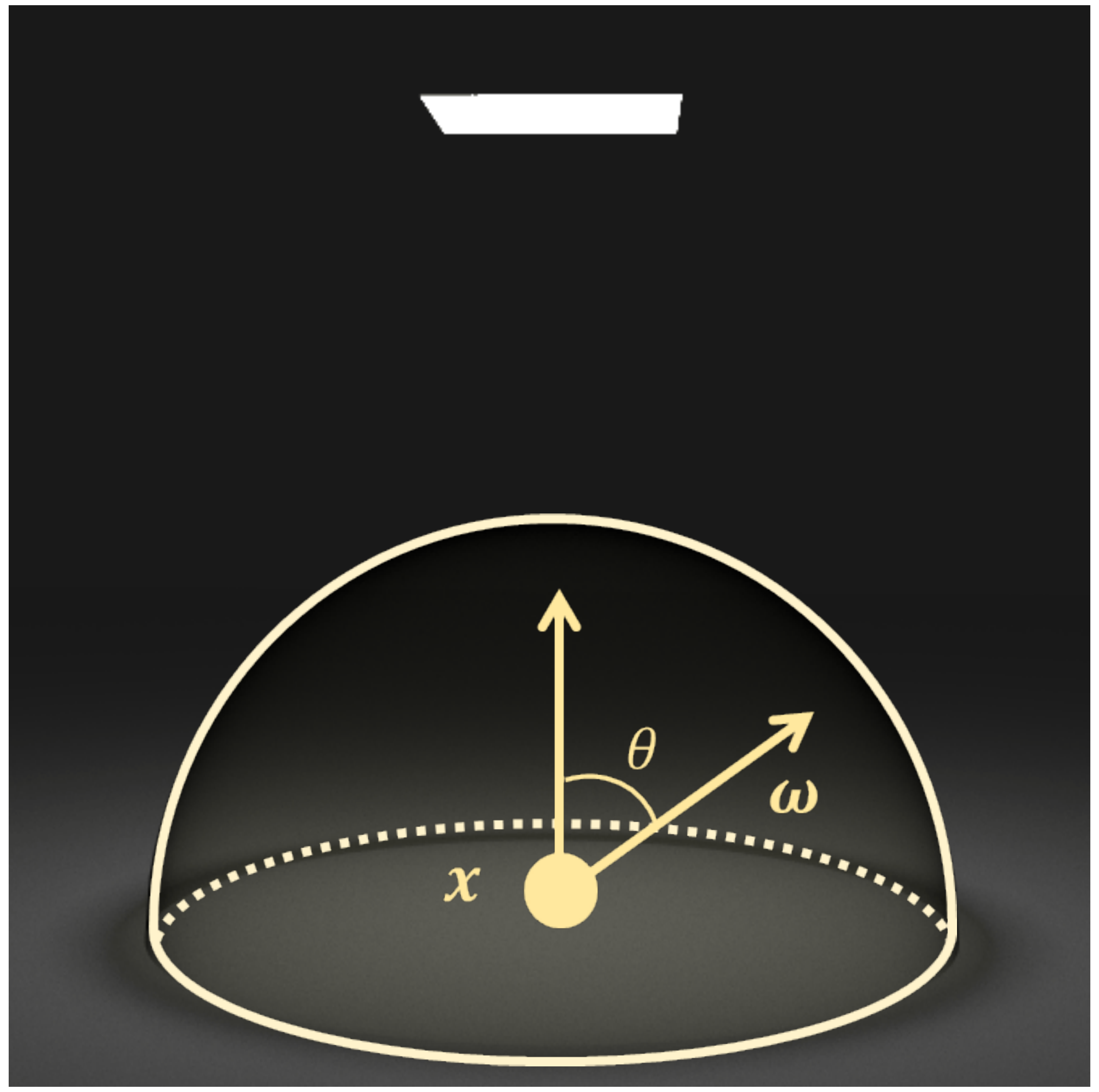
independent of the parameterization of $\partial \mathbb{H}^2$

Boundary integral

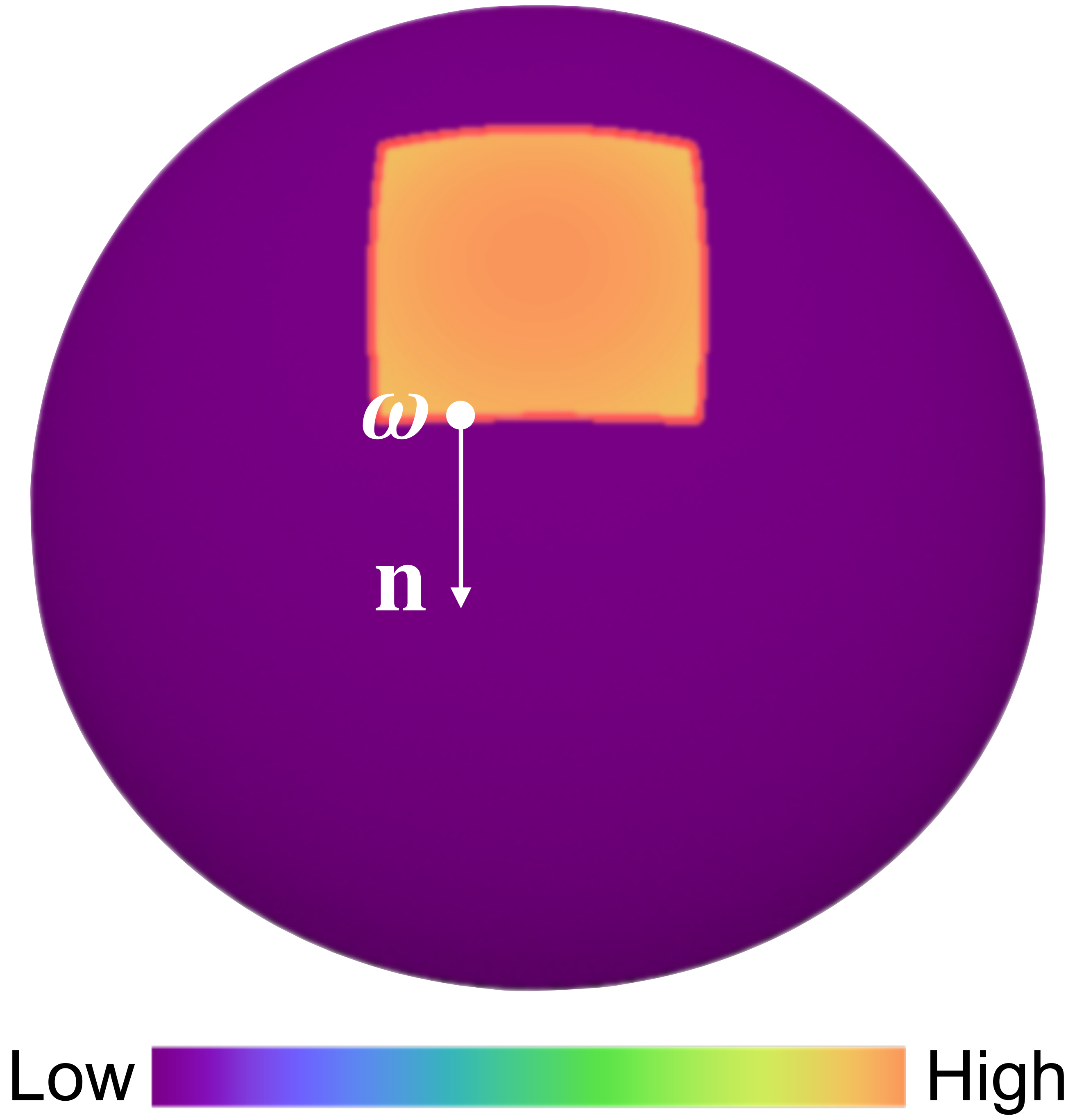
$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega}) \xrightarrow{\text{Reynolds}} \frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega}) \, \mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\boldsymbol{\omega}) \, \Delta f_E(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\boldsymbol{\omega})$



Scalar normal "velocity" of $\boldsymbol{\omega}$

$$V_{\partial\mathbb{H}^2}(\boldsymbol{\omega}) = \left\langle \mathbf{n}(\boldsymbol{\omega}), \frac{\mathrm{d}\boldsymbol{\omega}}{\mathrm{d}\pi} \right\rangle$$

independent of the parameterization of $\partial\mathbb{H}^2$

Difference of the integrand f_E across the boundary

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega})$$

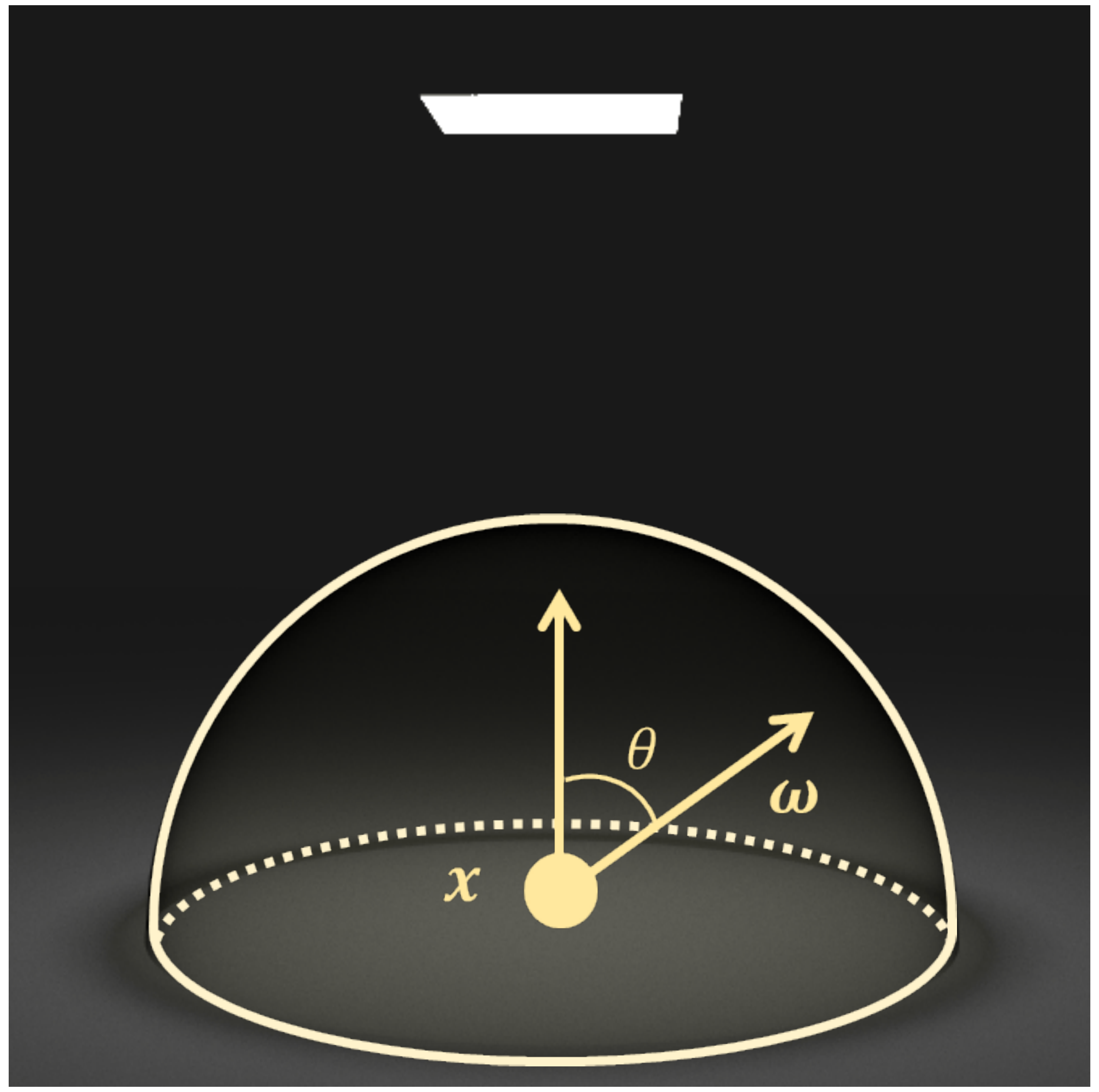
Reynolds ⇒

Boundary integral

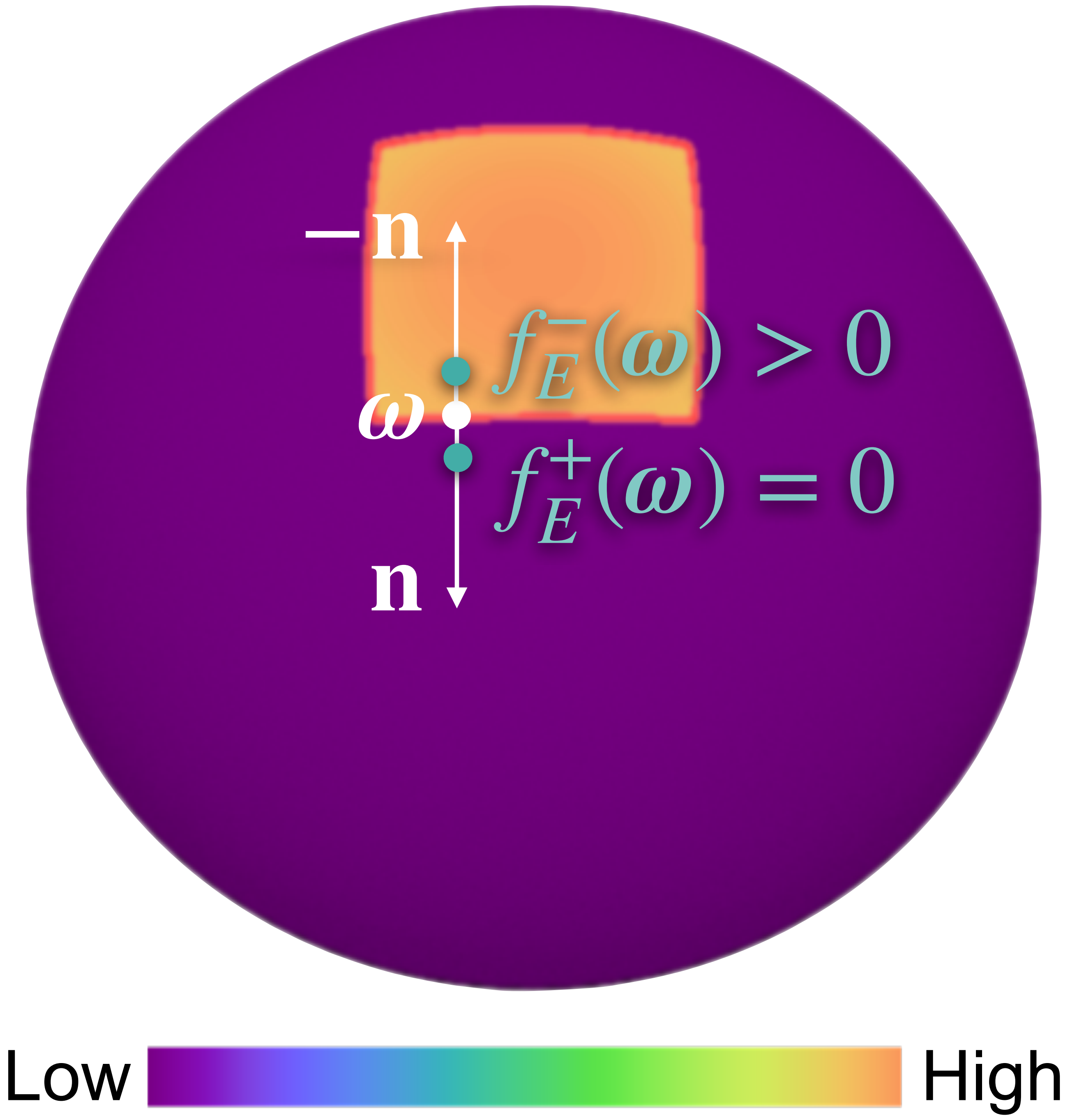
$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega}) \, \mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\boldsymbol{\omega}) \, \Delta f_E(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega})$$

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\omega)$



Scalar normal "velocity" of ω

$$V_{\partial\mathbb{H}^2}(\omega) = \left\langle \mathbf{n}(\omega), \frac{\mathrm{d}\omega}{\mathrm{d}\pi} \right\rangle$$

independent of the parameterization of $\partial\mathbb{H}^2$

Difference of the integrand f_E across the boundary

$$\Delta f_E(\omega) = f_E^-(\omega) - f_E^+(\omega)$$

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\omega)\,\cos\theta}^{f_E(\omega)}\,\mathrm{d}\sigma(\omega)$$

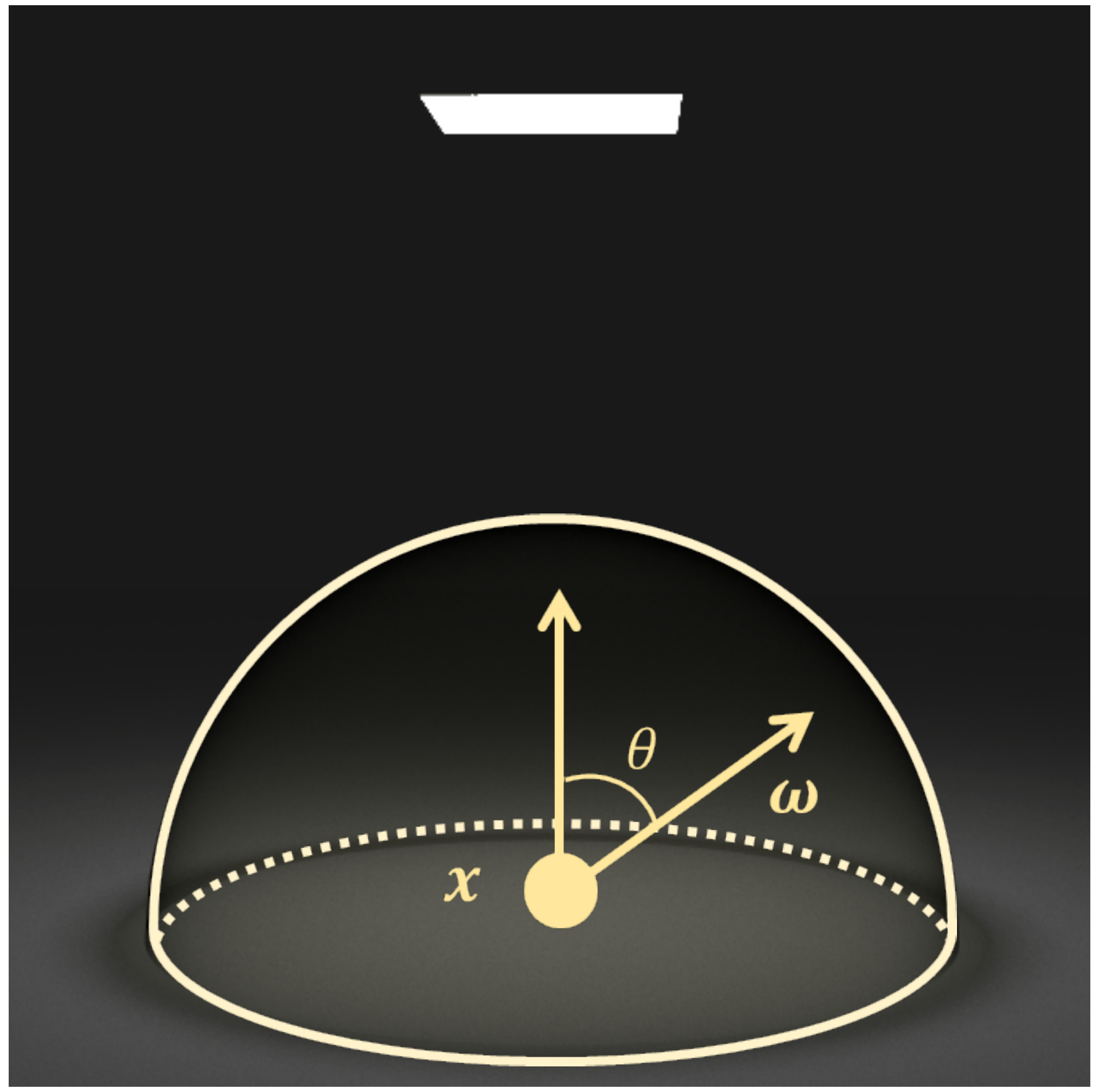
Reynolds →

Boundary integral

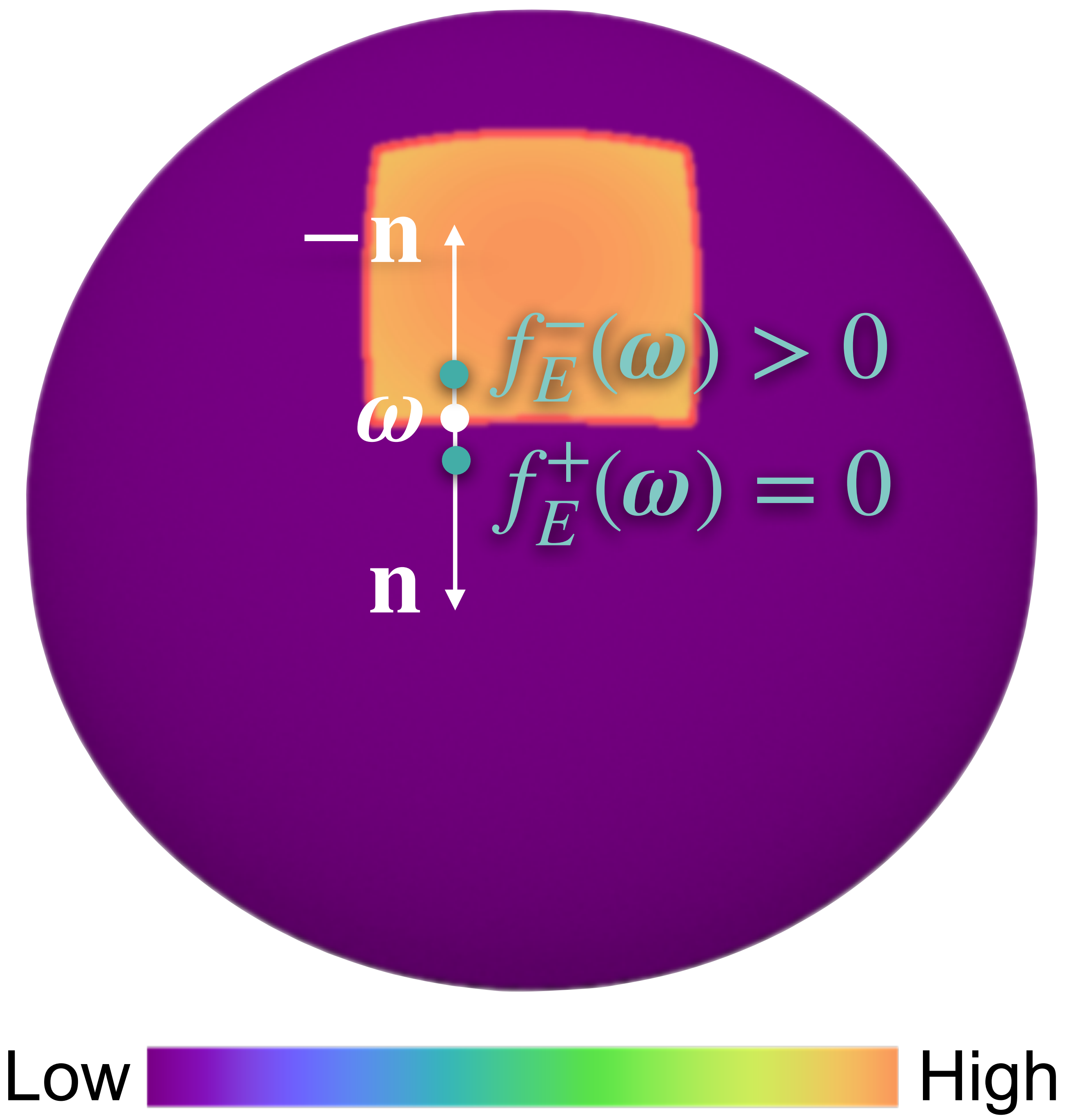
$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\omega)\,\mathrm{d}\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega)\,\Delta f_E(\omega)\,\mathrm{d}\ell(\omega)$$

WARM-UP: DIFFERENTIAL IRRADIANCE

π: emitter size



$f_E(\omega)$



Scalar normal "velocity" of ω

$$V_{\partial\mathbb{H}^2}(\omega) = \left\langle \mathbf{n}(\omega), \frac{\mathrm{d}\omega}{\mathrm{d}\pi} \right\rangle$$

independent of the parameterization of $\partial\mathbb{H}^2$

Difference of the integrand f_E across the boundary

$$\Delta f_E(\omega) = f_E^-(\omega) - f_E^+(\omega)$$

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\omega)\cos\theta}^{f_E(\omega)}\,\mathrm{d}\sigma(\omega)$$

Reynolds

General result

Interior integral

Boundary integral

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\omega)\,\mathrm{d}\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega)\,\Delta f_E(\omega)\,\mathrm{d}\ell(\omega)$$

DIFFERENTIAL RENDERING EQUATION

Interior integral

Boundary integral

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \, \mathrm{d}\sigma(\boldsymbol{\omega})$$

Reynolds

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega}) \, \mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\boldsymbol{\omega}) \, \Delta f_E(\boldsymbol{\omega}) \, \mathrm{d}\ell(\boldsymbol{\omega})$$

DIFFERENTIAL RENDERING EQUATION

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}) \cos\theta}^{f_E(\boldsymbol{\omega})} \,\mathrm{d}\sigma(\boldsymbol{\omega})$$

Reynolds ⇒

Interior integral | Boundary integral

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega})\,\mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\boldsymbol{\omega})\,\Delta f_E(\boldsymbol{\omega})\,\mathrm{d}\ell(\boldsymbol{\omega})$$

This can be generalized easily to obtain the differential rendering equation:

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i}) f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})} \,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

f_s : cosine-weighted BSDF

DIFFERENTIAL RENDERING EQUATION

$$E = \int_{\mathbb{H}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega})\,\cos\theta}^{f_E(\boldsymbol{\omega})}\,\mathrm{d}\sigma(\boldsymbol{\omega})$$

Reynolds ⇒

Interior integral | Boundary integral

$$\frac{\mathrm{d}E}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f_E}{\mathrm{d}\pi}(\boldsymbol{\omega})\,\mathrm{d}\sigma(\boldsymbol{\omega}) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\boldsymbol{\omega})\,\Delta f_E(\boldsymbol{\omega})\,\mathrm{d}\ell(\boldsymbol{\omega})$$

This can be generalized easily to obtain the differential rendering equation:

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})\,f_s(\boldsymbol{\omega}_\mathrm{i},\boldsymbol{\omega}_\mathrm{o})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})}\,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

f_s : cosine-weighted BSDF

⇓ Reynolds

Interior integral | Boundary integral

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi}L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\,\Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\,\mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi}L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

SOURCES OF DISCONTINUITIES

Assumptions:

No **zero-measure** (point and directional) **lights**

No **perfectly specular surfaces**

Continuous BSDFs

SOURCES OF DISCONTINUITIES

Assumptions:

No **zero-measure** (point and directional) **lights**

(which can create *hard shadow boundaries*)

Hard-to-detect discontinuities

No **perfectly specular surfaces**

(which can create *virtual images* of other objects)

Continuous BSDFs

SOURCES OF DISCONTINUITIES

Assumptions:

No **zero-measure** (point and directional) **lights**
(which can create *hard shadow boundaries*)

Hard-to-detect discontinuities

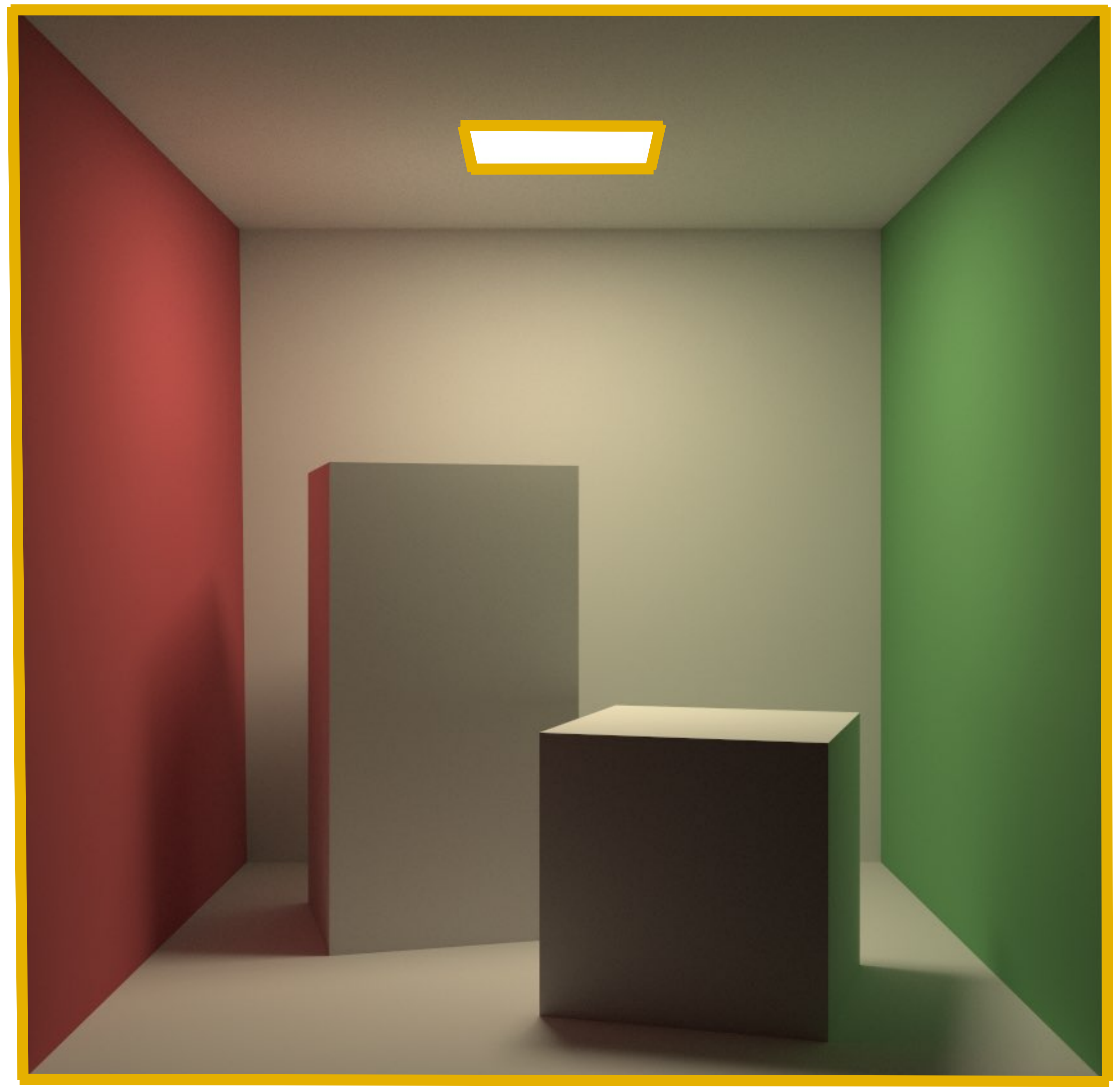
No **perfectly specular surfaces**
(which can create *virtual images* of other objects)

Continuous BSDFs

These limitations are largely practical and can be easily mitigated

SOURCES OF DISCONTINUITIES

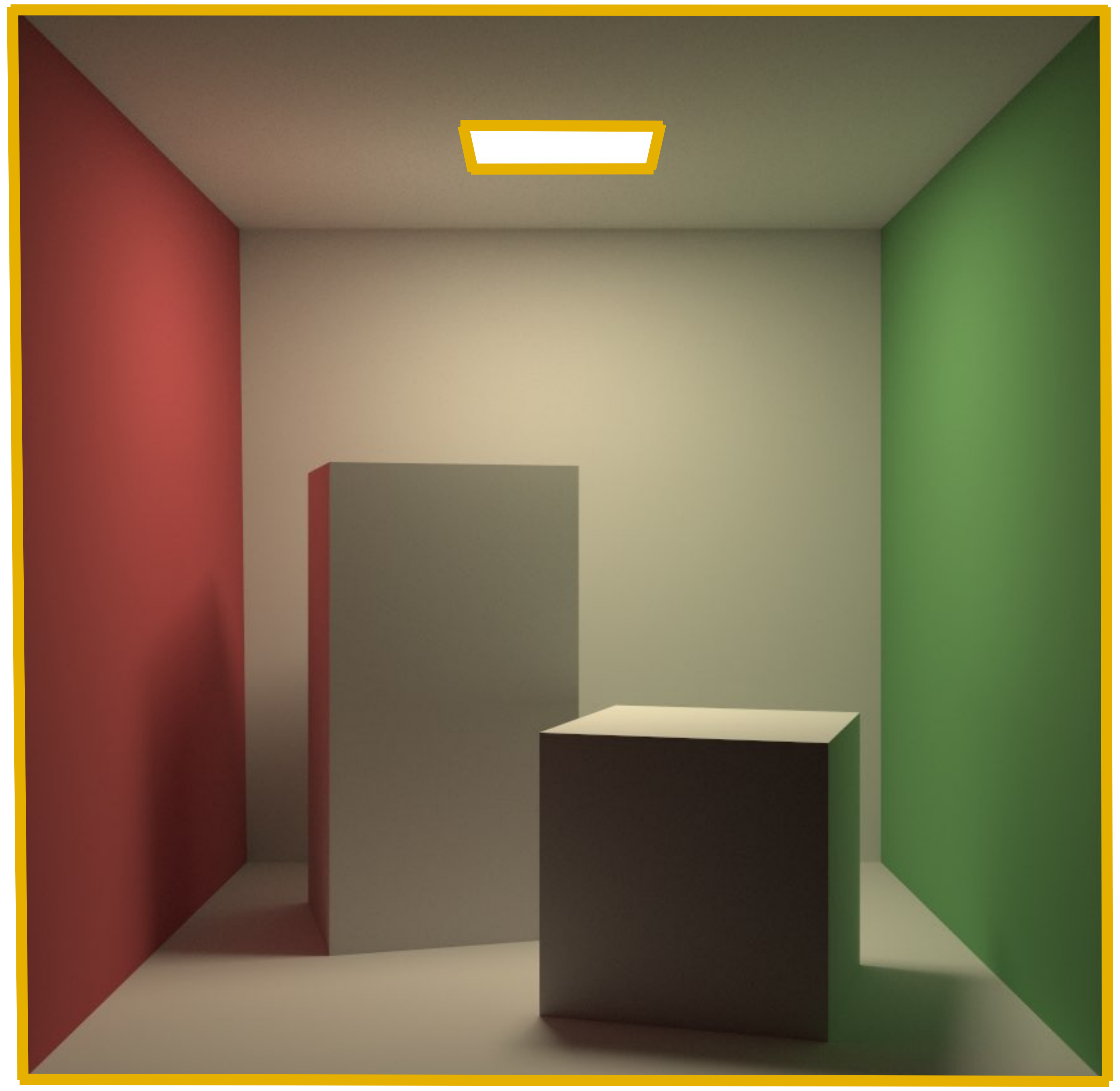
Boundary edges



(Topological) boundary of an object

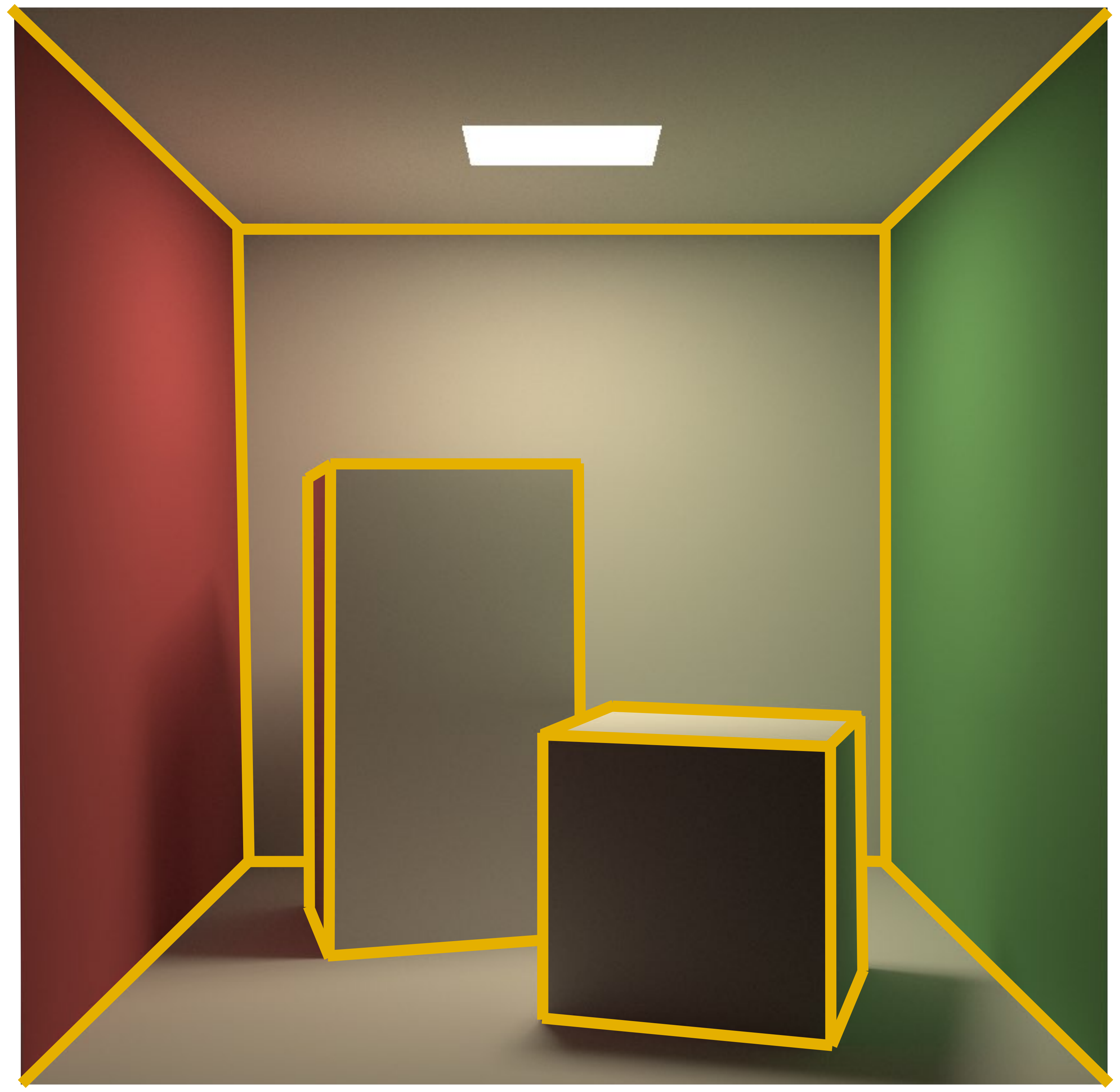
SOURCES OF DISCONTINUITIES

Boundary edges



(Topological) boundary of an object

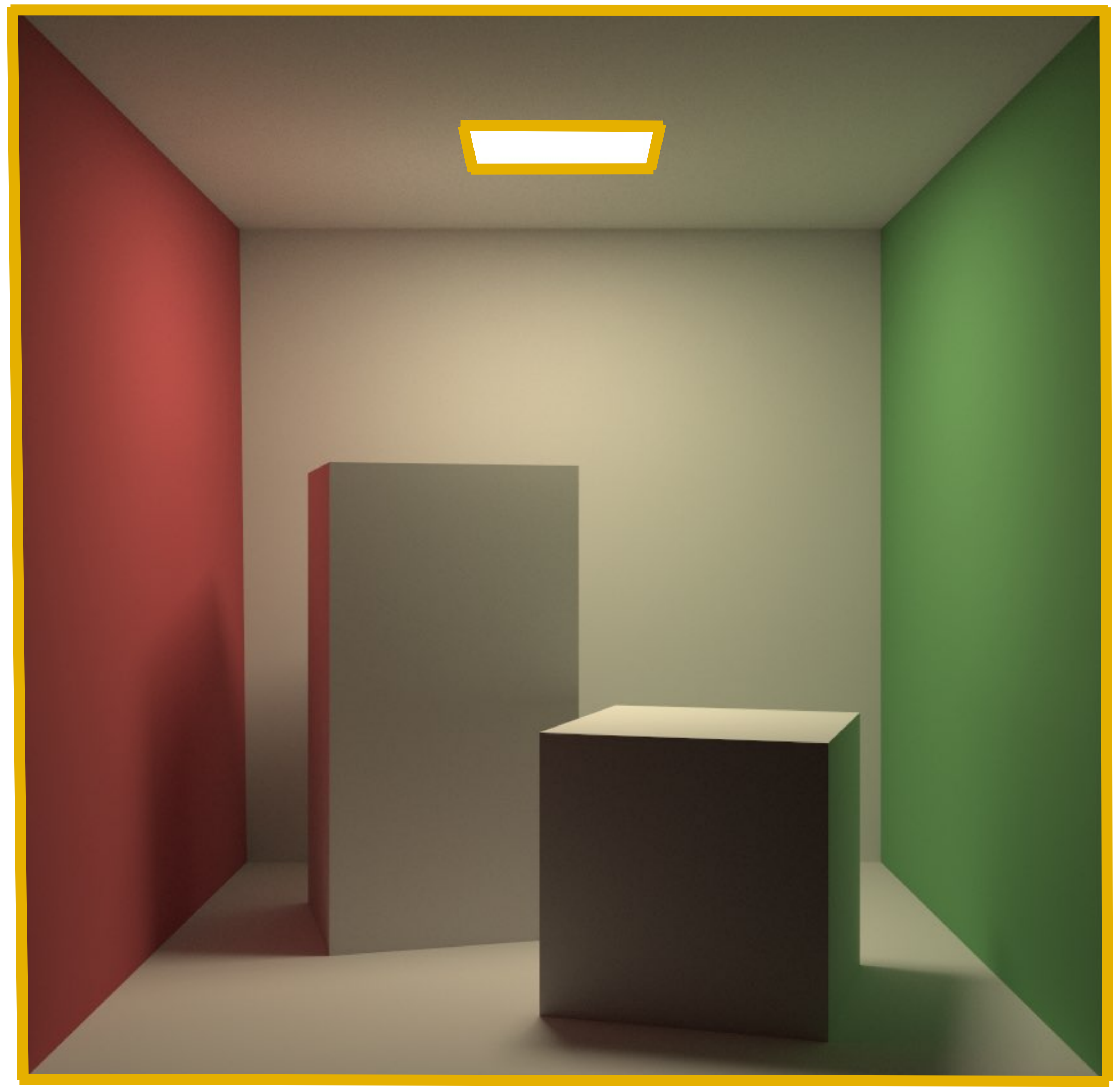
Sharp edges



Surface-normal discontinuities
(e.g., face edges)

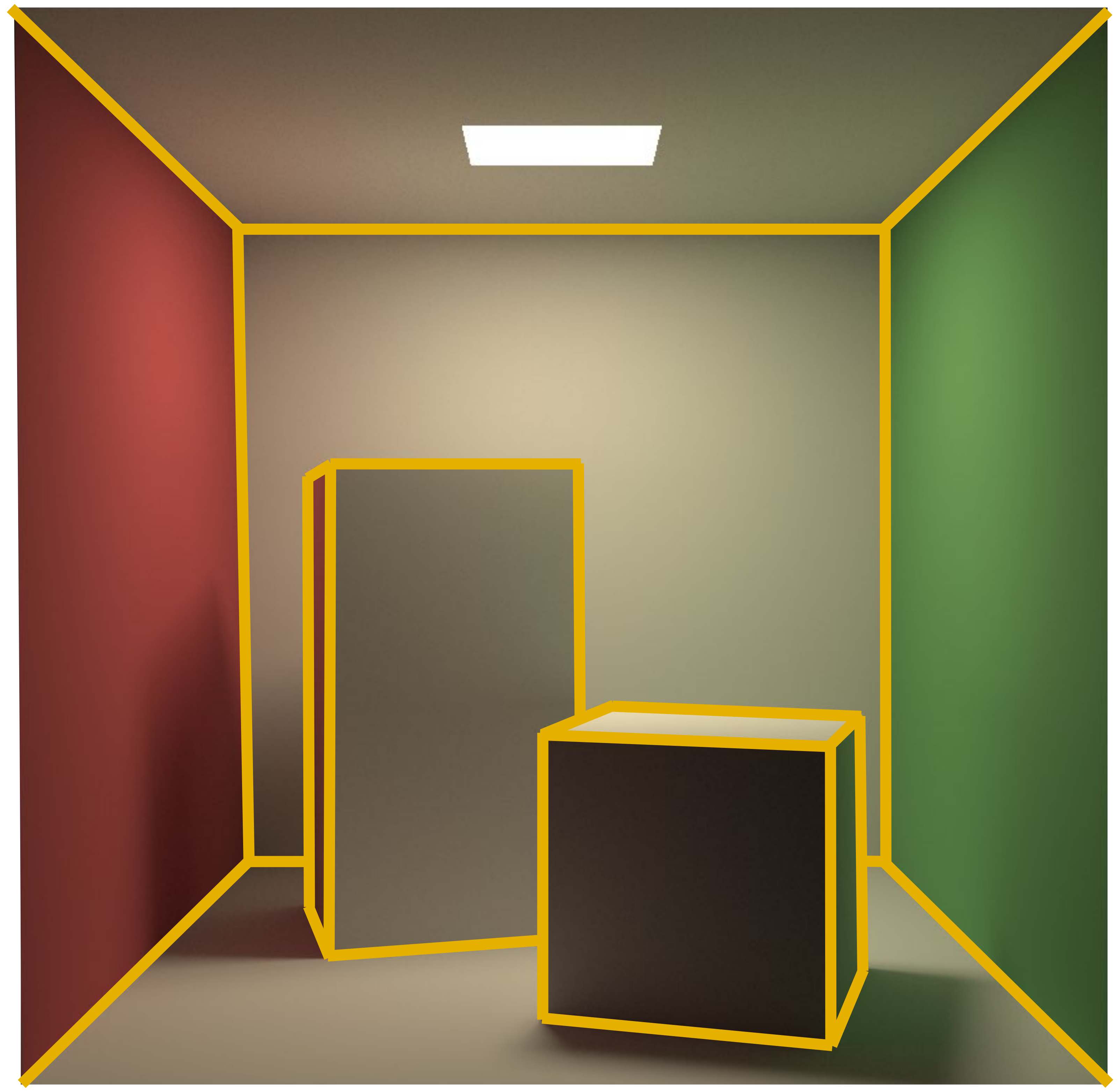
SOURCES OF DISCONTINUITIES

Boundary edges



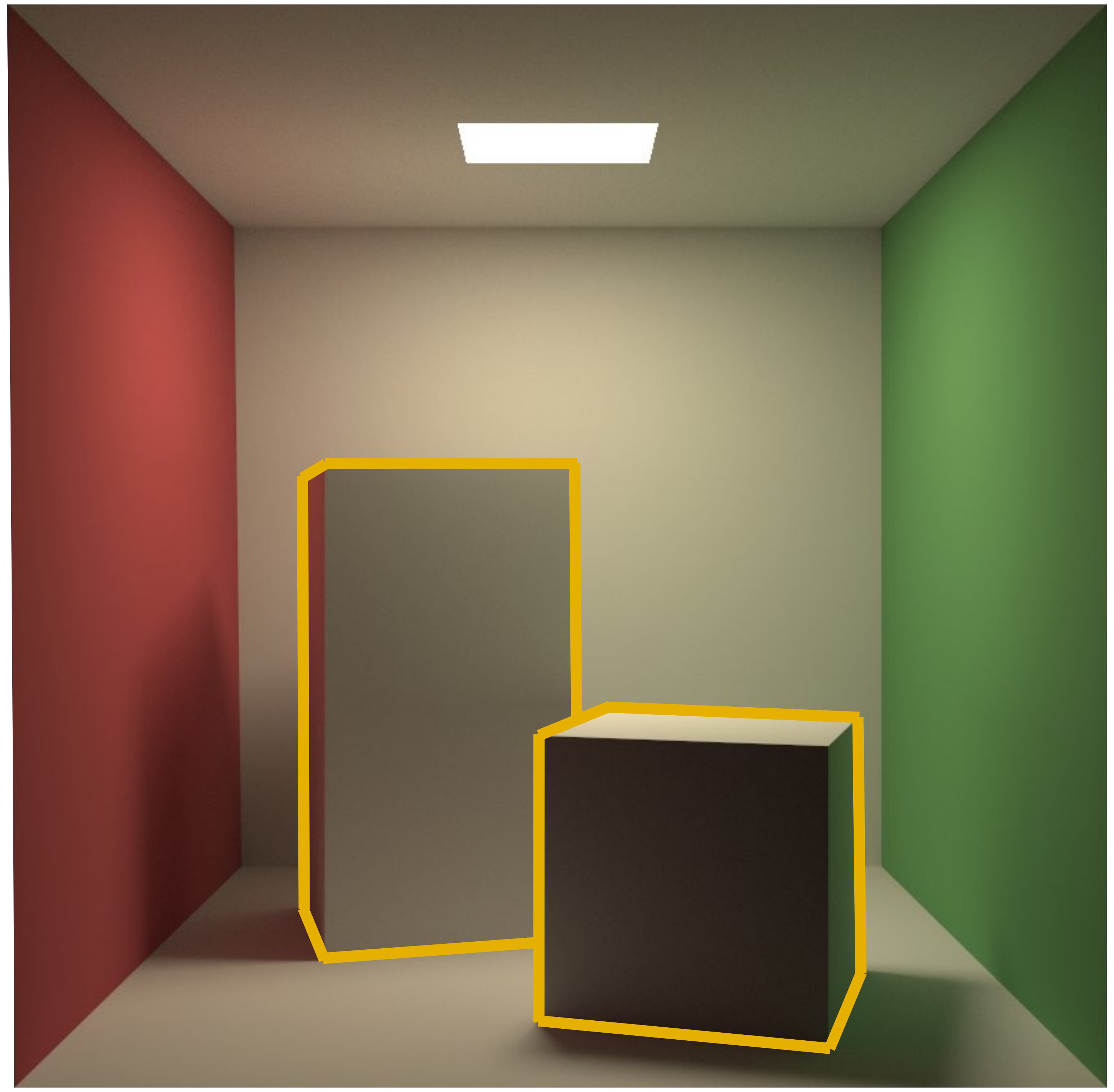
(Topological) boundary of an object

Sharp edges



Surface-normal discontinuities
(e.g., face edges)

Silhouette edges



View-dependent object silhouettes

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

Path tracing can be generalized to estimate L and $\mathrm{d}L/\mathrm{d}\pi$ jointly

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Interior integral

Boundary integral

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

Path tracing can be generalized to estimate L and $\mathrm{d}L/\mathrm{d}\pi$ jointly

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

Interior integral

Boundary integral

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Standard path tracing

Path tracing can be generalized to estimate L and $\mathrm{d}L/\mathrm{d}\pi$ jointly

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})} \,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

Interior integral

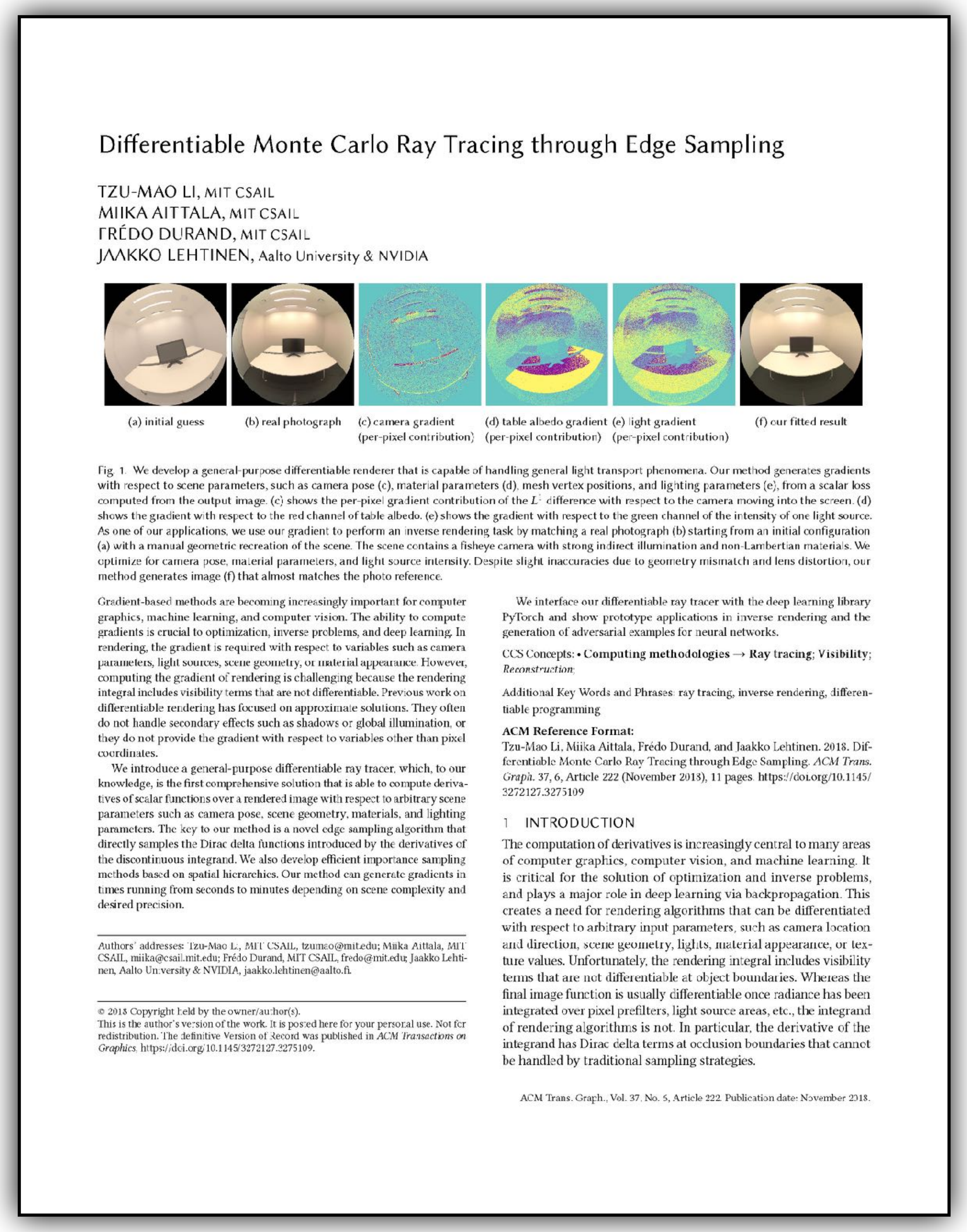
Boundary integral

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\,\Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\,\mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Standard path tracing

Edge sampling

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING



Differentiable Monte Carlo Ray Tracing through Edge Sampling

TZU-MAO LI, MIT CSAIL
MIIKA AITTALA, MIT CSAIL
FRÉDO DURAND, MIT CSAIL
JAAKKO LEHTINEN, Aalto University & NVIDIA

(a) initial guess (b) real photograph (c) camera gradient (per-pixel contribution) (d) table albedo gradient (per-pixel contribution) (e) light gradient (per-pixel contribution) (f) our fitted result

Fig. 1. We develop a general-purpose differentiable renderer that is capable of handling general light transport phenomena. Our method generates gradients with respect to scene parameters, such as camera pose (c), material parameters (d), mesh vertex positions, and lighting parameters (e), from a scalar loss computed from the output image. (c) shows the per-pixel gradient contribution of the L^1 difference with respect to the camera moving into the screen. (d) shows the gradient with respect to the red channel of table albedo. (e) shows the gradient with respect to the green channel of the intensity of one light source. As one of our applications, we use our gradient to perform an inverse rendering task by matching a real photograph (b) starting from an initial configuration (a) with a manual geometric recreation of the scene. The scene contains a fisheye camera with strong indirect illumination and non-Lambertian materials. We optimize for camera pose, material parameters, and light source intensity. Despite slight inaccuracies due to geometry mismatch and lens distortion, our method generates image (f) that almost matches the photo reference.

Gradient-based methods are becoming increasingly important for computer graphics, machine learning, and computer vision. The ability to compute gradients is crucial to optimization, inverse problems, and deep learning. In rendering, the gradient is required with respect to variables such as camera parameters, light sources, scene geometry, or material appearance. However, computing the gradient of rendering is challenging because the rendering integral includes visibility terms that are not differentiable. Previous work on differentiable rendering has focused on approximate solutions. They often do not handle secondary effects such as shadows or global illumination, or they do not provide the gradient with respect to variables other than pixel coordinates.

We introduce a general-purpose differentiable ray tracer, which, to our knowledge, is the first comprehensive solution that is able to compute derivatives of scalar functions over a rendered image with respect to arbitrary scene parameters such as camera pose, scene geometry, materials, and lighting parameters. The key to our method is a novel edge sampling algorithm that directly samples the Dirac delta functions introduced by the derivatives of the discontinuous integrand. We also develop efficient importance sampling methods based on spatial hierarchies. Our method can generate gradients in times running from seconds to minutes depending on scene complexity and desired precision.

We interface our differentiable ray tracer with the deep learning library PyTorch and show prototype applications in inverse rendering and the generation of adversarial examples for neural networks.

Authors' addresses: Tzu-Mao Li, MIT CSAIL, tzumao@mit.edu; Miika Aittala, MIT CSAIL, miika@csail.mit.edu; Frédo Durand, MIT CSAIL, fredo@mit.edu; Jaakko Lehtinen, Aalto University & NVIDIA, jaakko.lehtinen@aalto.fi.

CCS Concepts: • **Computing methodologies → Ray tracing; Visibility**; *Reconstruction*;

Additional Key Words and Phrases: ray tracing, inverse rendering, differentiable programming

ACM Reference Format:
Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo Ray Tracing through Edge Sampling. *ACM Trans. Graph.* 37, 6, Article 222 (November 2018), 11 pages. https://doi.org/10.1145/3272127.3275109

1 INTRODUCTION

The computation of derivatives is increasingly central to many areas of computer graphics, computer vision, and machine learning. It is critical for the solution of optimization and inverse problems, and plays a major role in deep learning via backpropagation. This creates a need for rendering algorithms that can be differentiated with respect to arbitrary input parameters, such as camera location and direction, scene geometry, lights, material appearance, or texture values. Unfortunately, the rendering integral includes visibility terms that are not differentiable at object boundaries. Whereas the final image function is usually differentiable once radiance has been integrated over pixel prefilters, light source areas, etc., the integrand of rendering algorithms is not. In particular, the derivative of the integrand has Dirac delta terms at occlusion boundaries that cannot be handled by traditional sampling strategies.

ACM Trans. Graph., Vol. 37, No. 6, Article 222. Publication date: November 2018.

Differentiable Monte Carlo Ray Tracing through Edge Sampling

Tzu-Mao Li, Miika Aittala, Frédo Durand, Jaakko Lehtinen

SIGGRAPH Asia 2018

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}) f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow \mathrm{rayIntersect}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow \mathrm{rayIntersect}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})}\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Standard PT w/ symbolic differentiation

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i})}^{f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})} \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})\, \Delta f_{\mathrm{RE}}(\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i})$$

$$+ \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

$$\Delta f_{\mathrm{RE}} = \Delta(f_s\, L_\mathrm{i}) = f_s\, \Delta L_\mathrm{i}$$

(assuming f_s to be continuous)

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

MONTE CARLO EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining $\partial\mathbb{S}^2$, the discontinuity points of ΔL_i (w.r.t. incident direction $\boldsymbol{\omega}_\mathrm{i}$)

MONTE CARLO EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

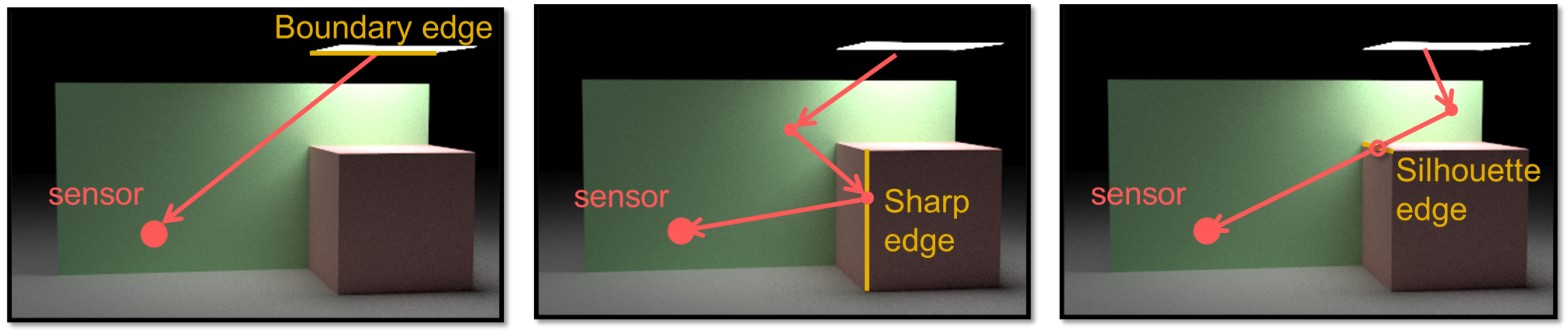
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining $\partial\mathbb{S}^2$, the discontinuity points of ΔL_i (w.r.t. incident direction $\boldsymbol{\omega}_\mathrm{i}$)



- For polygonal meshes, $\partial\mathbb{S}^2$ can involve:
  - Boundary edges (associated with only one face)
  - Face edges (when not using smooth shading)
  - Silhouette edges (shared by a front and a back face)

MONTE CARLO EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

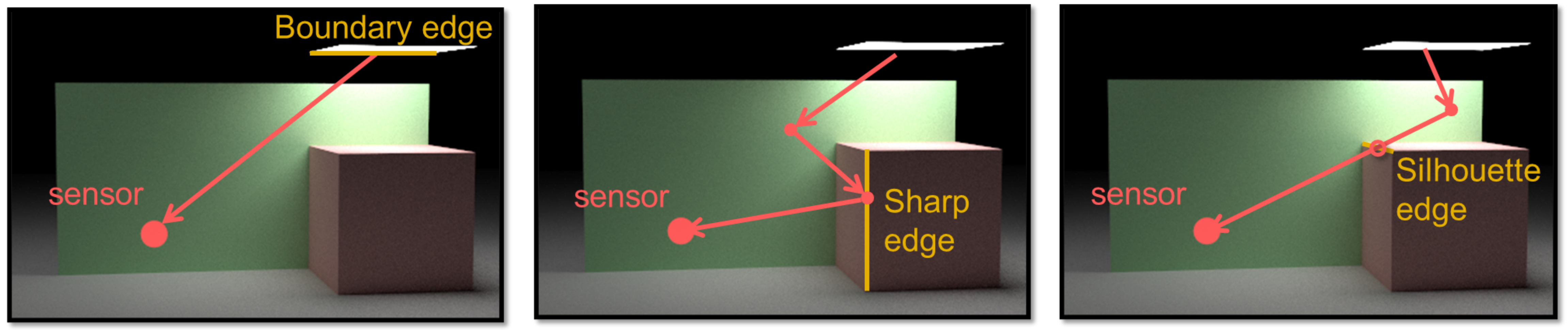
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining $\partial\mathbb{S}^2$, the discontinuity points of ΔL_i (w.r.t. incident direction $\boldsymbol{\omega}_\mathrm{i}$)



- For polygonal meshes, $\partial\mathbb{S}^2$ can involve:
  - Boundary edges (associated with only one face)
  - Face edges (when not using smooth shading)
  - Silhouette edges (shared by a front and a back face)
    - Requires traversing a 6D BVH
    - Expensive for complex scenes

MONTE CARLO EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow \mathrm{rayIntersect}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

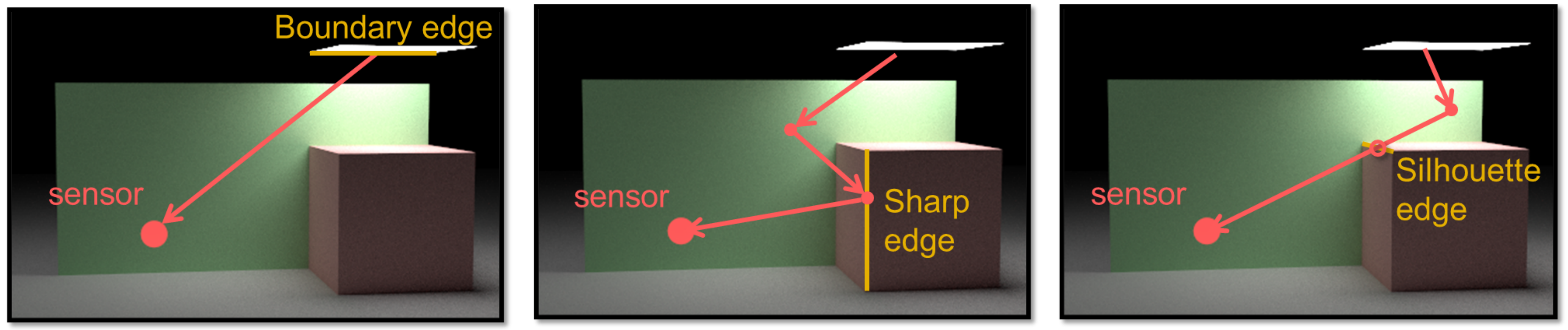
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining $\partial\mathbb{S}^2$, the discontinuity points of ΔL_i (w.r.t. incident direction $\boldsymbol{\omega}_\mathrm{i}$)



- For polygonal meshes, $\partial\mathbb{S}^2$ can involve:
  - Boundary edges (associated with only one face)
  - Face edges (when not using smooth shading)
  - Silhouette edges (shared by a front and a back face)
    - Requires traversing a 6D BVH
    - Expensive for complex scenes
    - To be addressed later!

COMPUTING ΔL_i

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

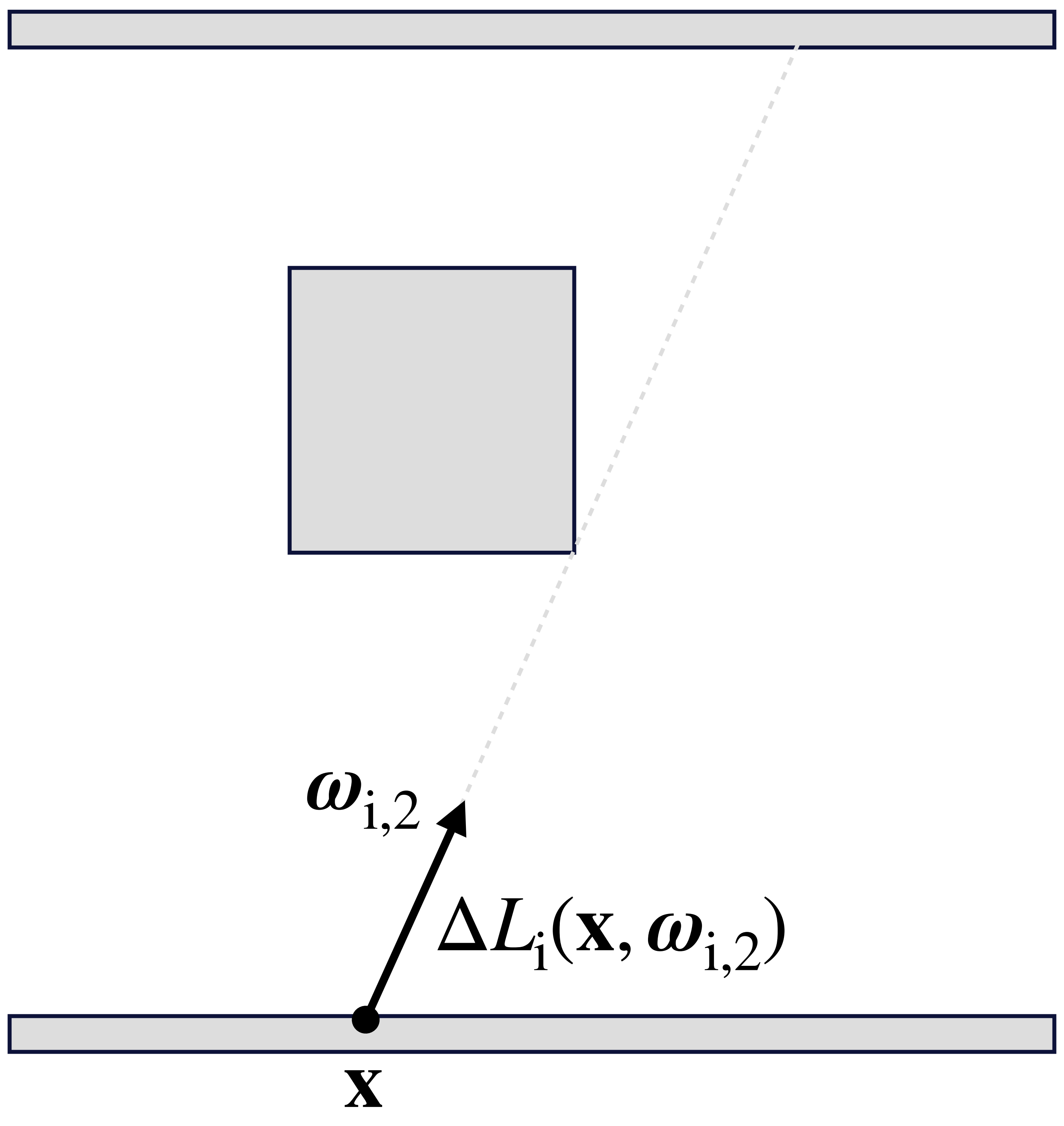
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$



COMPUTING ΔL_i

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

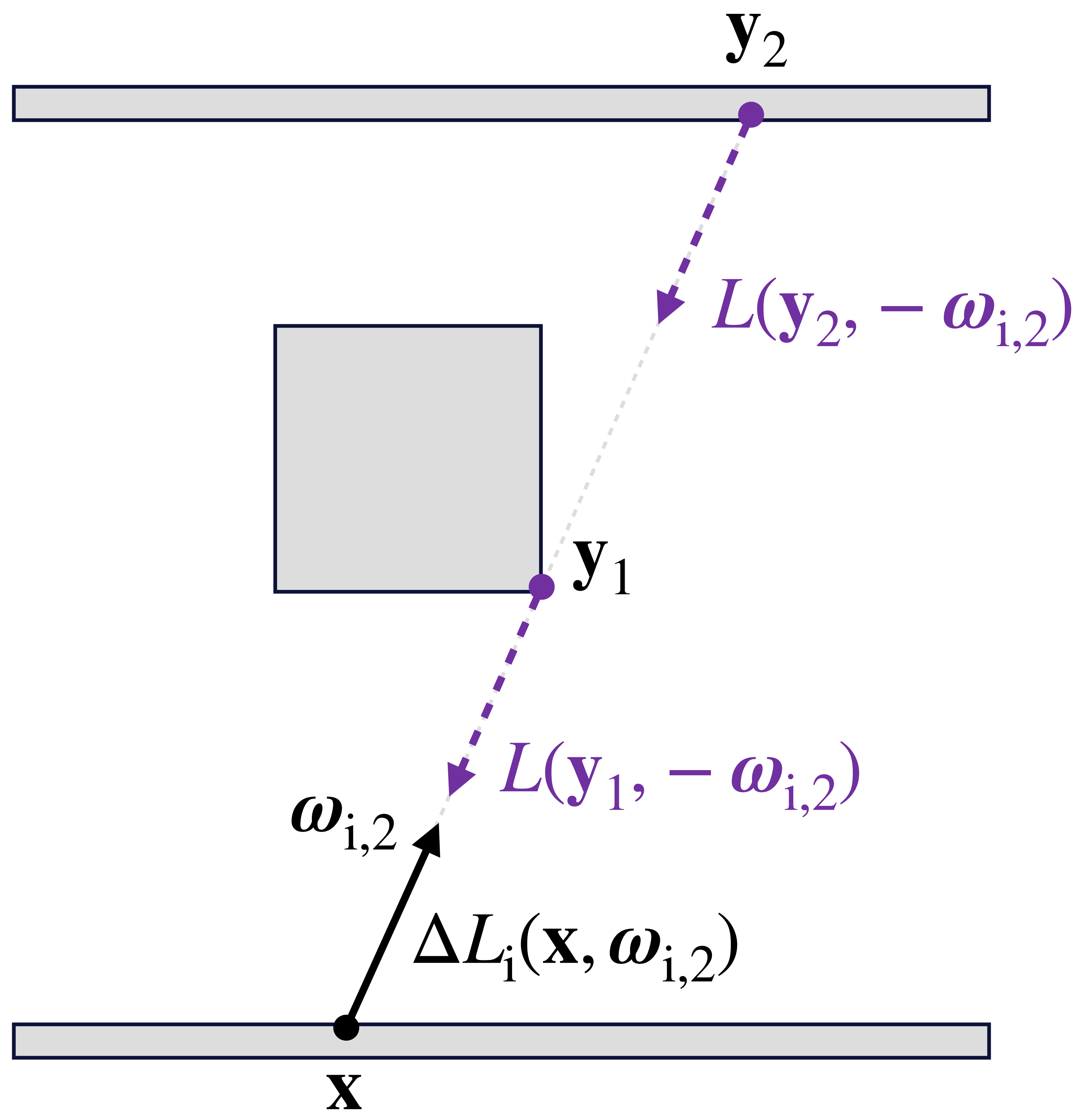
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$



$$\Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}) = \pm \left[L(\mathbf{y}_1, -\boldsymbol{\omega}_{\mathrm{i},2}) - L(\mathbf{y}_2, -\boldsymbol{\omega}_{\mathrm{i},2})\right]$$

COMPUTING ΔL_i

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

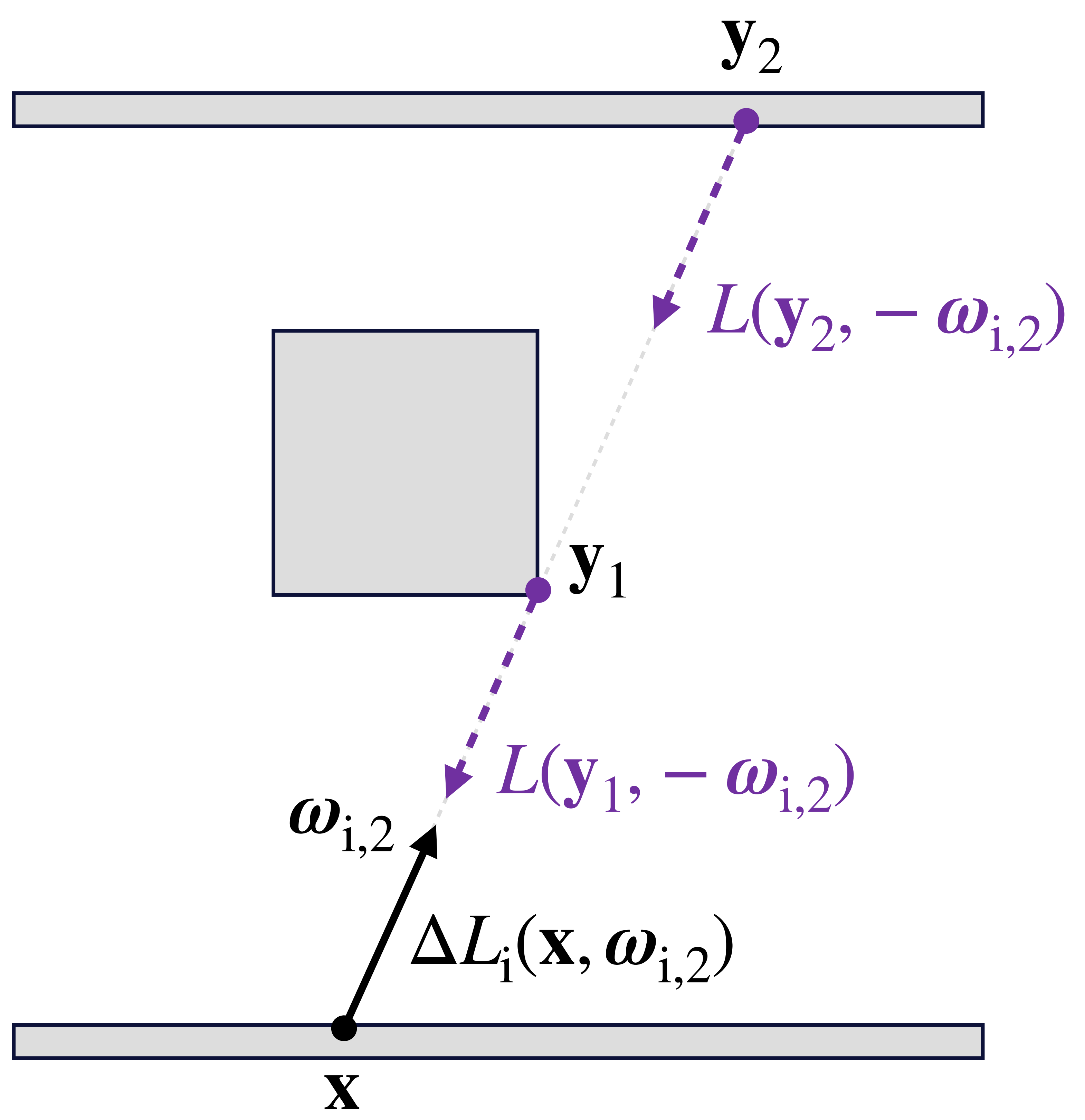
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$



$$\Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}) = \pm \left[L(\mathbf{y}_1, -\boldsymbol{\omega}_{\mathrm{i},2}) - L(\mathbf{y}_2, -\boldsymbol{\omega}_{\mathrm{i},2})\right]$$

Radiance values $L(\mathbf{y}_1, -\boldsymbol{\omega}_{\mathrm{i},2})$ and $L(\mathbf{y}_2, -\boldsymbol{\omega}_{\mathrm{i},2})$ can be computed by tracing additional “side” paths

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow \mathrm{rayIntersect}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

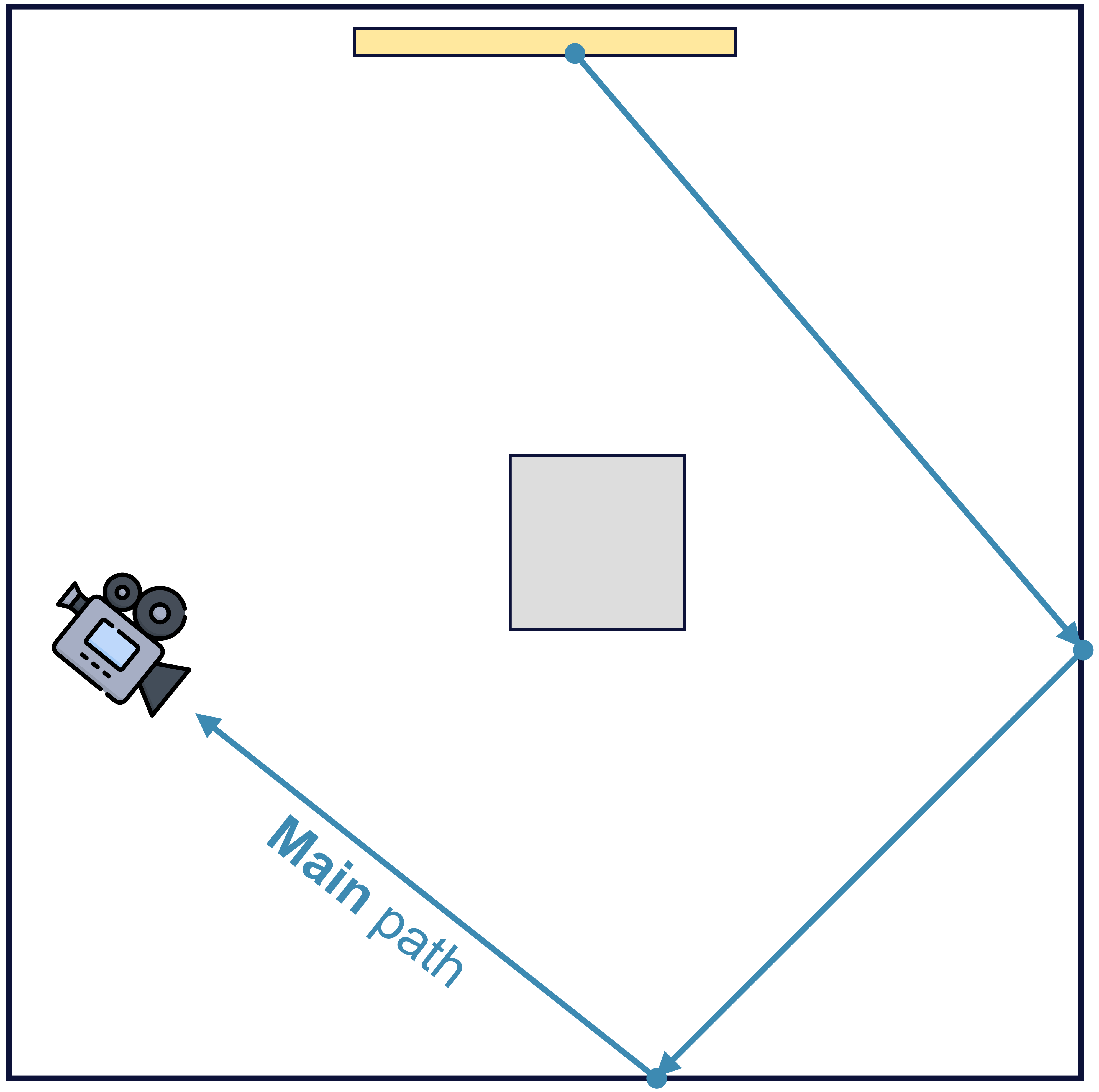
Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$



DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\text{dPT}(\mathbf{x}, \boldsymbol{\omega}_\text{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\text{o})$ and $\frac{\text{d}}{\text{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\text{o})]$ jointly

sample $\boldsymbol{\omega}_{\text{i},1} \in \mathbb{S}^2$ with probability $p_{\text{i},1}$

$\mathbf{y} \leftarrow \text{rayIntersect}(\mathbf{x}, \boldsymbol{\omega}_{\text{i},1})$

$(L_\text{i}, \dot{L}_\text{i}) \leftarrow \text{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\text{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\text{i},1}, \boldsymbol{\omega}_\text{o})\, L_\text{i}}{p_{\text{i},1}}$$

$$\dot{L} \leftarrow \frac{\frac{\text{d}}{\text{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\text{i},1}, \boldsymbol{\omega}_\text{o})]\, L_\text{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\text{i},1}, \boldsymbol{\omega}_\text{o})\, \dot{L}_\text{i}}{p_{\text{i},1}}$$

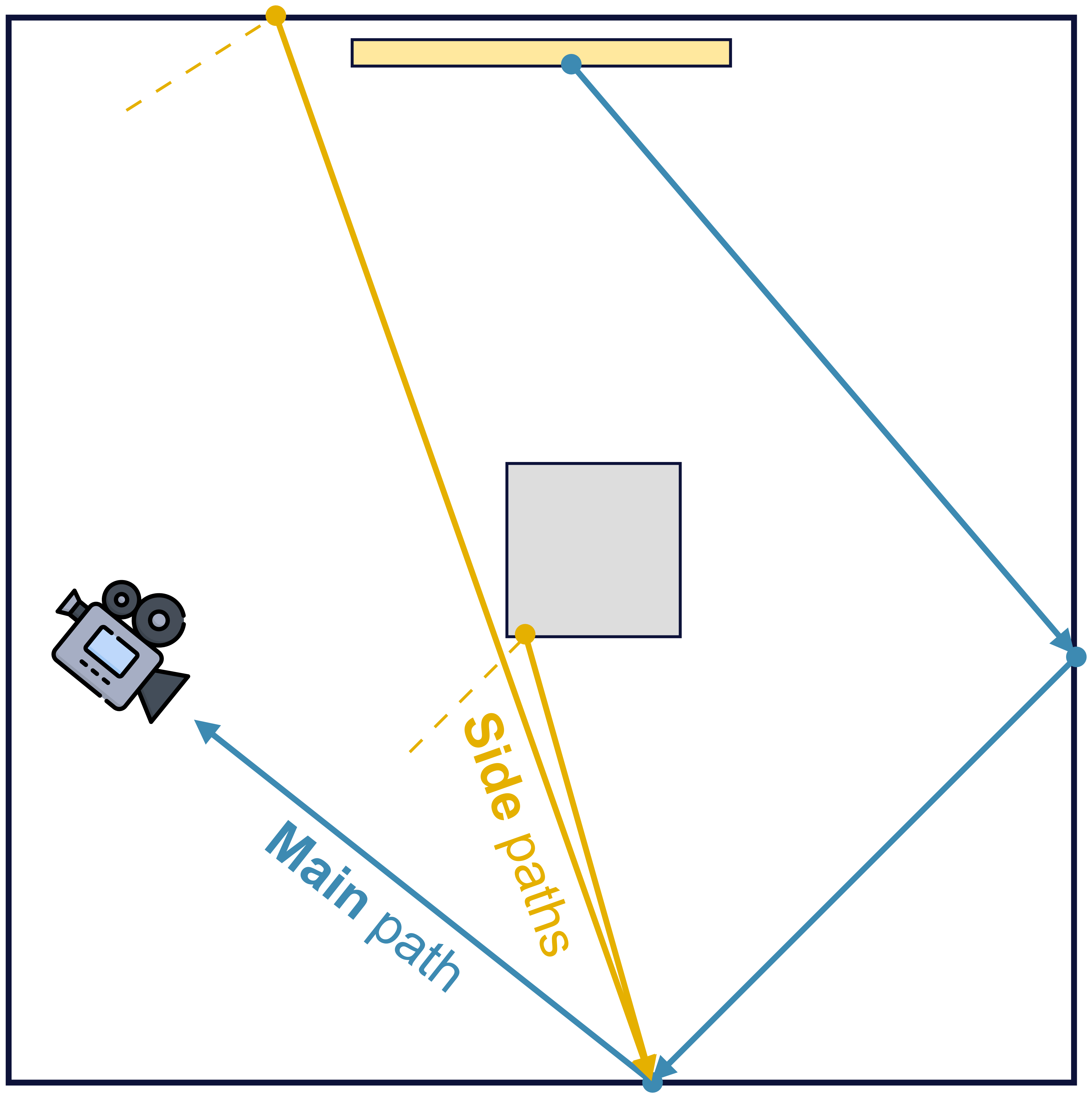
Standard PT w/ symbolic differentiation

sample $\boldsymbol{\omega}_{\text{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\text{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\text{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\text{i},2}, \boldsymbol{\omega}_\text{o})\, \Delta L_\text{i}(\mathbf{x}, \boldsymbol{\omega}_{\text{i},2})}{p_{\text{i},2}}$$

Monte Carlo edge sampling

return $\left(L + L_\text{e}(\mathbf{x}, \boldsymbol{\omega}_\text{o}),\ \dot{L} + \frac{\text{d}}{\text{d}\pi} L_\text{e}(\mathbf{x}, \boldsymbol{\omega}_\text{o})\right)$



DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

$\mathrm{dPT}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$: # Estimate $L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$ and $\frac{\mathrm{d}}{\mathrm{d}\pi}[L(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})]$ jointly

Standard PT w/ symbolic differentiation:

sample $\boldsymbol{\omega}_{\mathrm{i},1} \in \mathbb{S}^2$ with probability $p_{\mathrm{i},1}$

$\mathbf{y} \leftarrow$ rayIntersect$(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1})$

$(L_\mathrm{i}, \dot{L}_\mathrm{i}) \leftarrow \mathrm{dPT}(\mathbf{y}, -\boldsymbol{\omega}_{\mathrm{i},1})$

$$L \leftarrow \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}}{p_{\mathrm{i},1}}$$

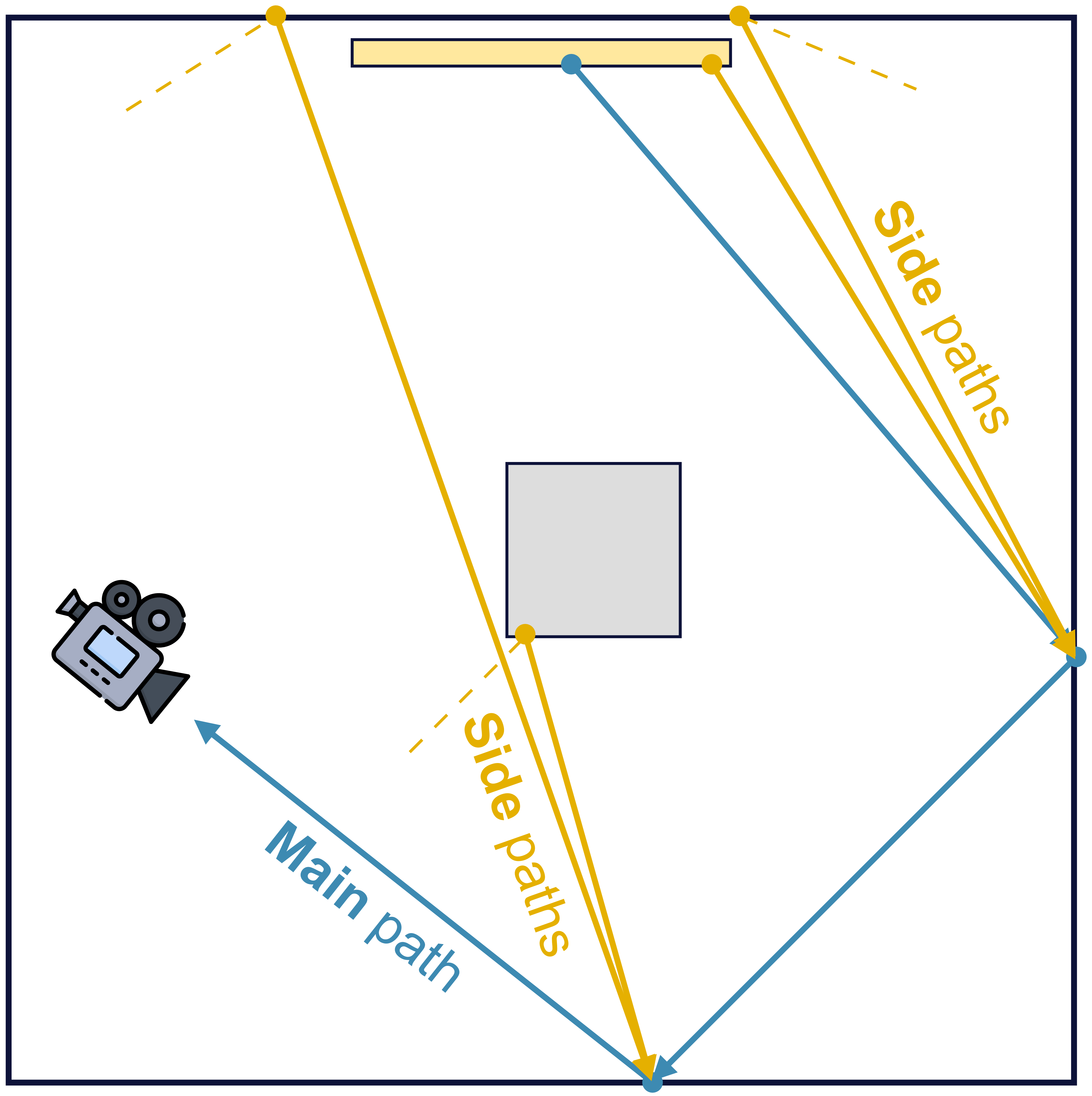
$$\dot{L} \leftarrow \frac{\frac{\mathrm{d}}{\mathrm{d}\pi}[f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})]\, L_\mathrm{i} + f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},1}, \boldsymbol{\omega}_\mathrm{o})\, \dot{L}_\mathrm{i}}{p_{\mathrm{i},1}}$$

Monte Carlo edge sampling:

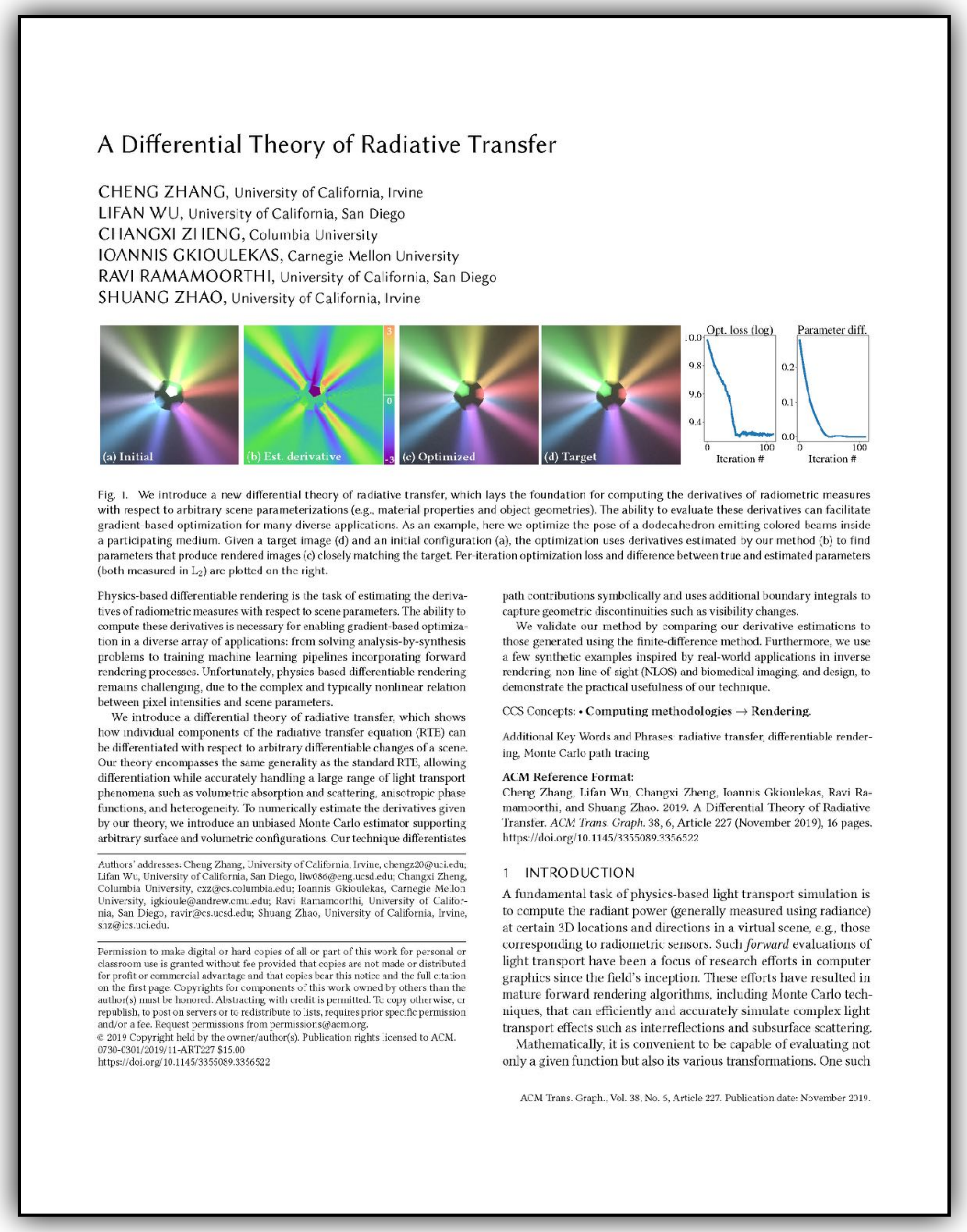
sample $\boldsymbol{\omega}_{\mathrm{i},2} \in \partial\mathbb{S}^2$ with probability $p_{\mathrm{i},2}$

$$\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})\, f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2}, \boldsymbol{\omega}_\mathrm{o})\, \Delta L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i},2})}{p_{\mathrm{i},2}}$$

return $\left(L + L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}),\ \dot{L} + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})\right)$



DIFFERENTIAL RADIATIVE TRANSFER



A Differential Theory of Radiative Transfer

CHENG ZHANG, University of California, Irvine
LIFAN WU, University of California, San Diego
CHANGXI ZHENG, Columbia University
IOANNIS GKIOULEKAS, Carnegie Mellon University
RAVI RAMAMOORTHI, University of California, San Diego
SHUANG ZHAO, University of California, Irvine

Fig. 1. We introduce a new differential theory of radiative transfer, which lays the foundation for computing the derivatives of radiometric measures with respect to arbitrary scene parameterizations (e.g., material properties and object geometries). The ability to evaluate these derivatives can facilitate gradient-based optimization for many diverse applications. As an example, here we optimize the pose of a dodecahedron emitting colored beams inside a participating medium. Given a target image (d) and an initial configuration (a), the optimization uses derivatives estimated by our method (b) to find parameters that produce rendered images (c) closely matching the target. Per-iteration optimization loss and difference between true and estimated parameters (both measured in L_2) are plotted on the right.

Physics-based differentiable rendering is the task of estimating the derivatives of radiometric measures with respect to scene parameters. The ability to compute these derivatives is necessary for enabling gradient-based optimization in a diverse array of applications: from solving analysis-by-synthesis problems to training machine learning pipelines incorporating forward rendering processes. Unfortunately, physics based differentiable rendering remains challenging, due to the complex and typically nonlinear relation between pixel intensities and scene parameters.

We introduce a differential theory of radiative transfer, which shows how individual components of the radiative transfer equation (RTE) can be differentiated with respect to arbitrary differentiable changes of a scene. Our theory encompasses the same generality as the standard RTE, allowing differentiation while accurately handling a large range of light transport phenomena such as volumetric absorption and scattering, anisotropic phase functions, and heterogeneity. To numerically estimate the derivatives given by our theory, we introduce an unbiased Monte Carlo estimator supporting arbitrary surface and volumetric configurations. Our technique differentiates path contributions symbolically and uses additional boundary integrals to capture geometric discontinuities such as visibility changes.

We validate our method by comparing our derivative estimations to those generated using the finite-difference method. Furthermore, we use a few synthetic examples inspired by real-world applications in inverse rendering, non line of sight (NLOS) and biomedical imaging, and design, to demonstrate the practical usefulness of our technique.

CCS Concepts: • **Computing methodologies → Rendering.**

Additional Key Words and Phrases: radiative transfer, differentiable rendering, Monte Carlo path tracing

ACM Reference Format:
Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A Differential Theory of Radiative Transfer. *ACM Trans. Graph.* 38, 6, Article 227 (November 2019), 16 pages. https://doi.org/10.1145/3355089.3356522

Authors' addresses: Cheng Zhang, University of California, Irvine, chengz20@uci.edu; Lifan Wu, University of California, San Diego, liw086@eng.ucsd.edu; Changxi Zheng, Columbia University, cxz@cs.columbia.edu; Ioannis Gkioulekas, Carnegie Mellon University, igkioule@andrew.cmu.edu; Ravi Ramamoorthi, University of California, San Diego, ravir@cs.ucsd.edu; Shuang Zhao, University of California, Irvine, shz@ics.uci.edu.

0730-0301/2019/11-ART227 $15.00
https://doi.org/10.1145/3355089.3356522

1 INTRODUCTION

A fundamental task of physics-based light transport simulation is to compute the radiant power (generally measured using radiance) at certain 3D locations and directions in a virtual scene, e.g., those corresponding to radiometric sensors. Such *forward* evaluations of light transport have been a focus of research efforts in computer graphics since the field's inception. These efforts have resulted in mature forward rendering algorithms, including Monte Carlo techniques, that can efficiently and accurately simulate complex light transport effects such as interreflections and subsurface scattering.

Mathematically, it is convenient to be capable of evaluating not only a given function but also its various transformations. One such

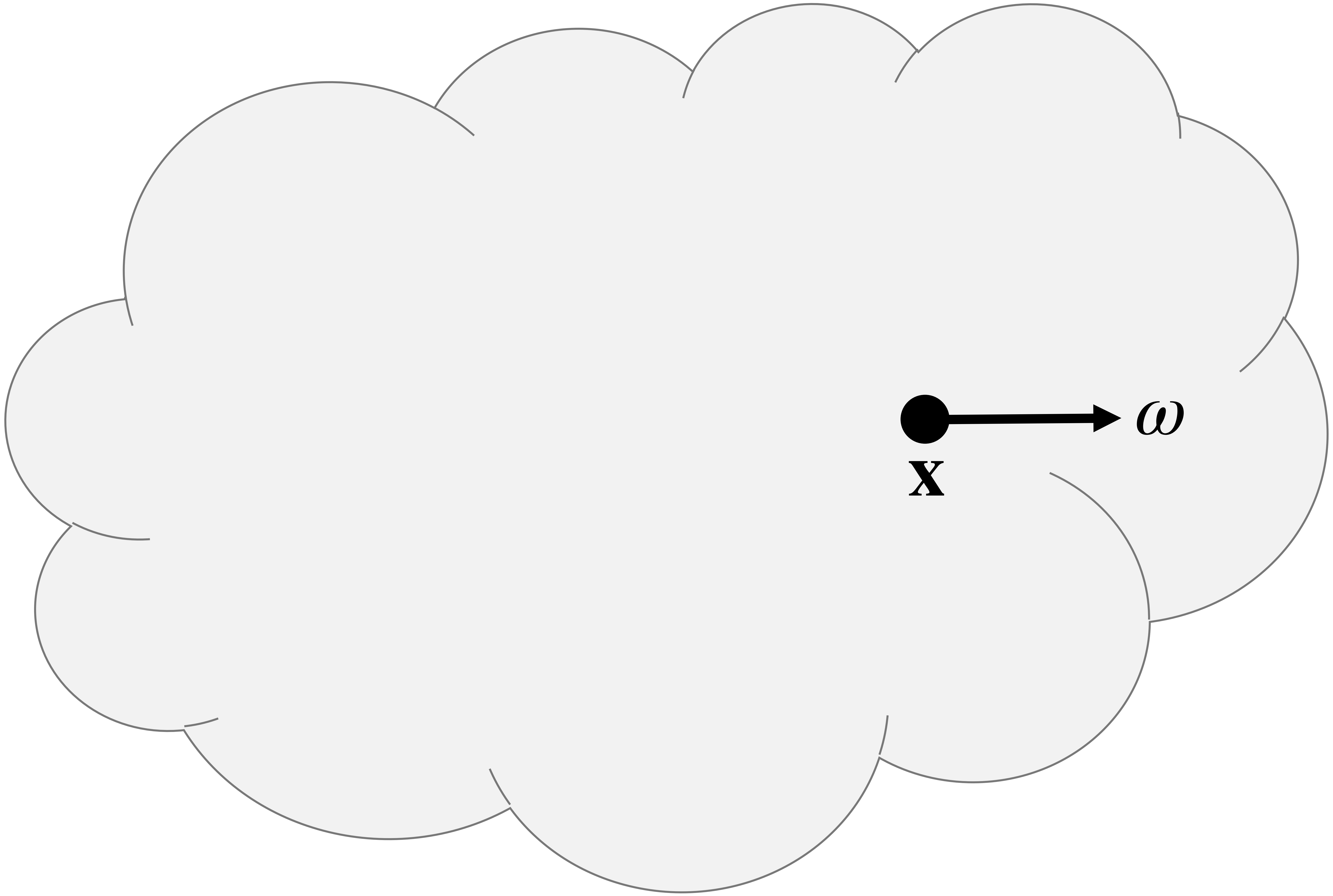
ACM Trans. Graph., Vol. 38, No. 6, Article 227. Publication date: November 2019.

A Differential Theory of Radiative Transfer

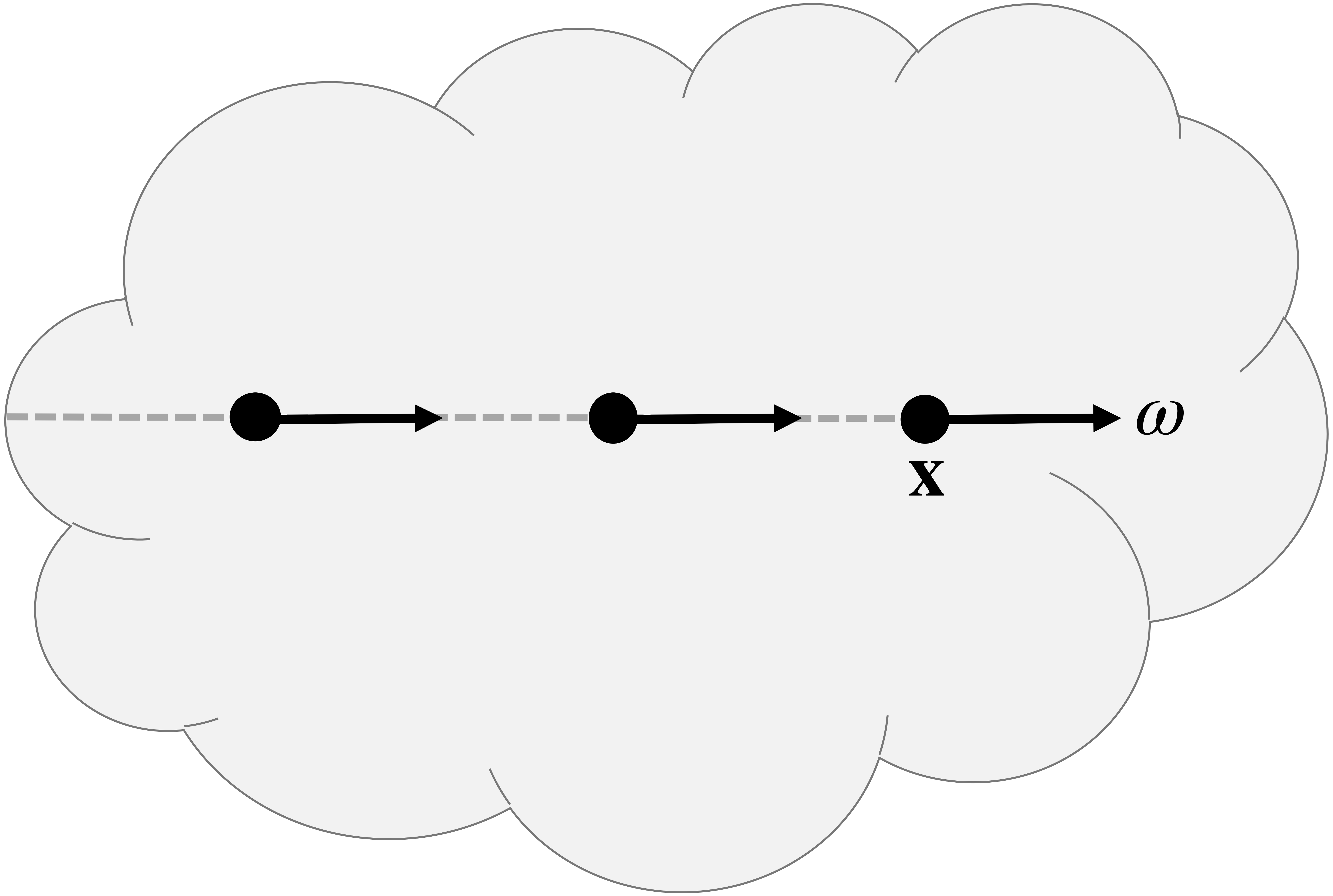
Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, Shuang Zhao

SIGGRAPH Asia 2019

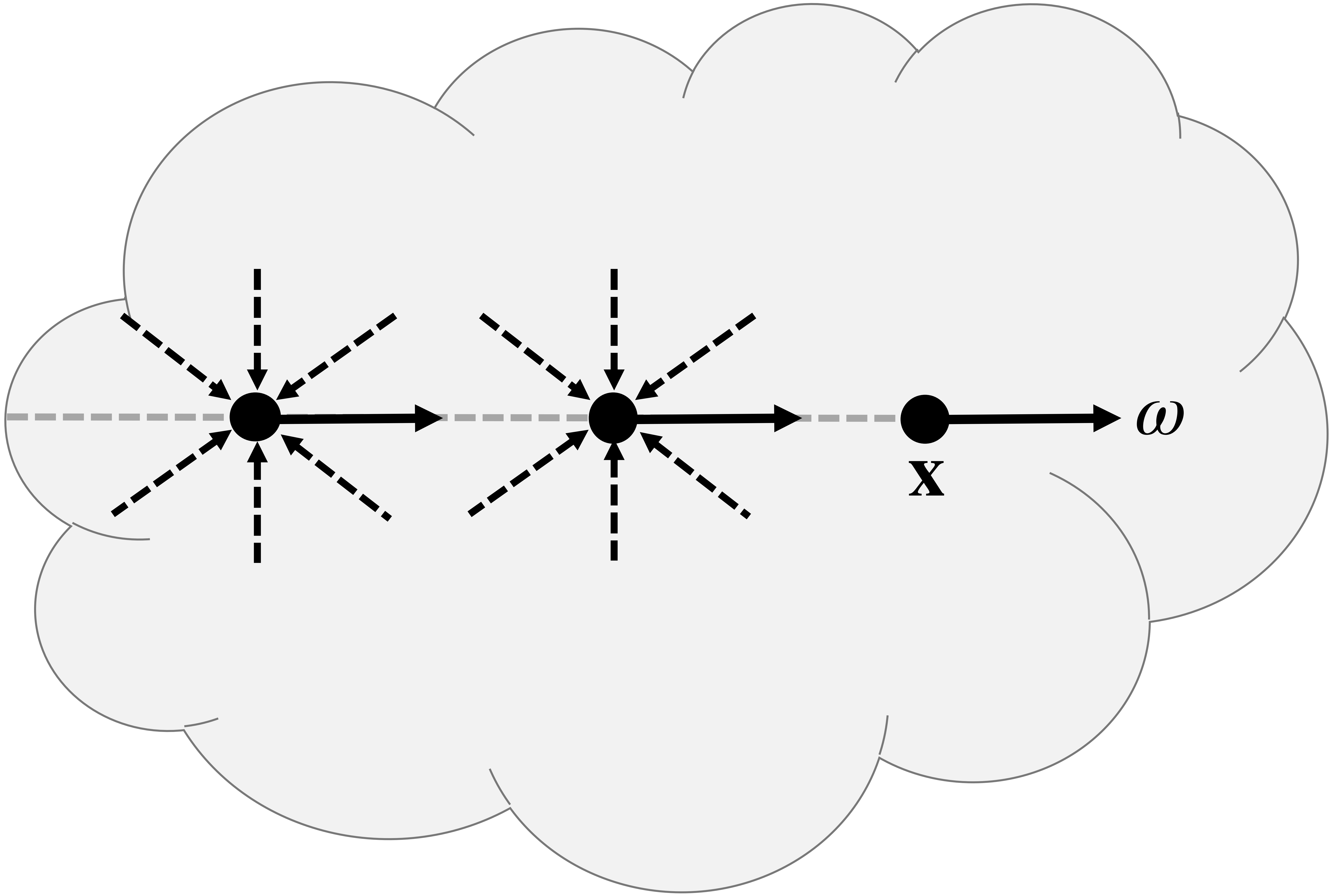
RECAP: RADIATIVE TRANSFER THEORY



RECAP: RADIATIVE TRANSFER THEORY



RECAP: RADIATIVE TRANSFER THEORY



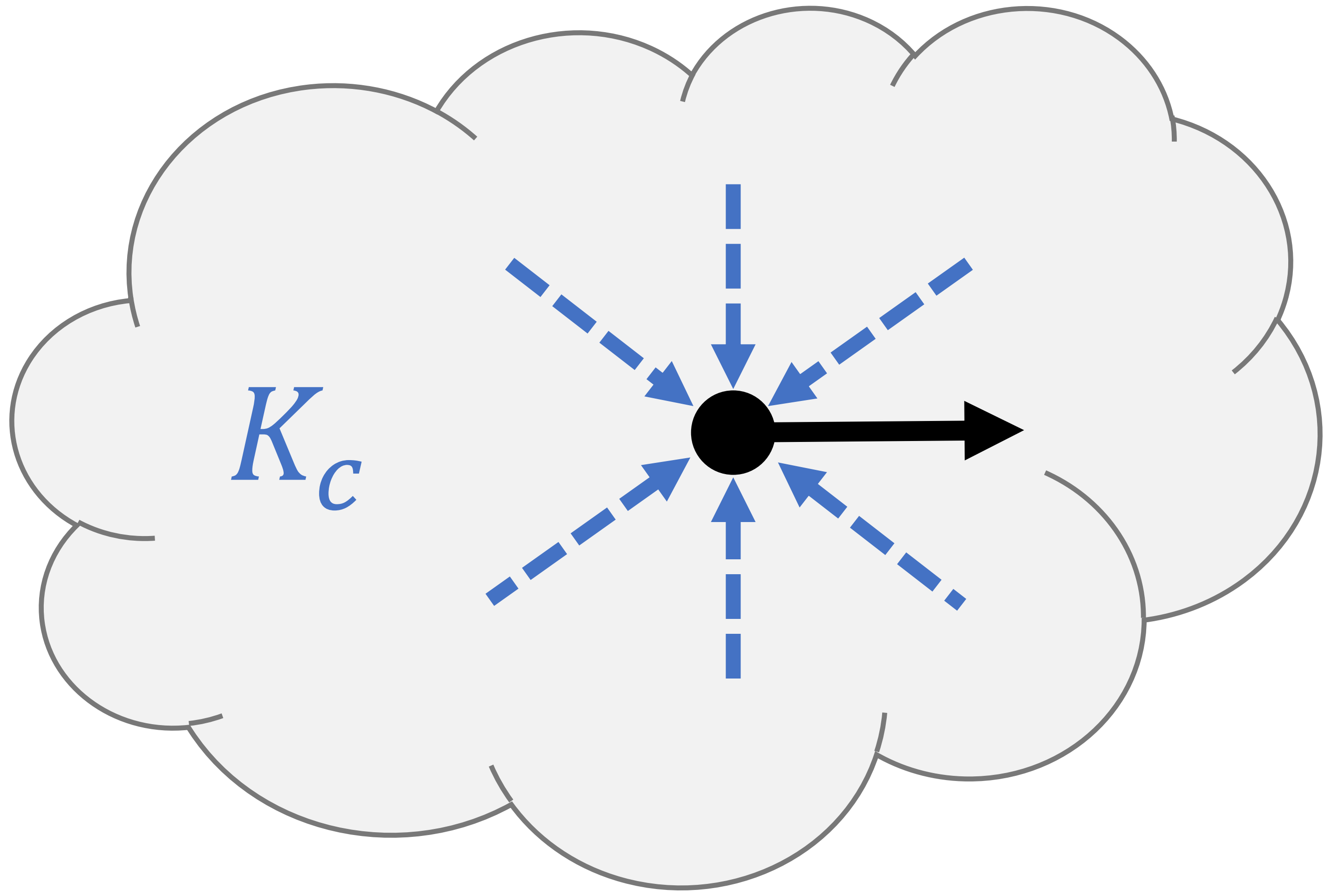
RECAP: RADIATIVE TRANSFER THEORY

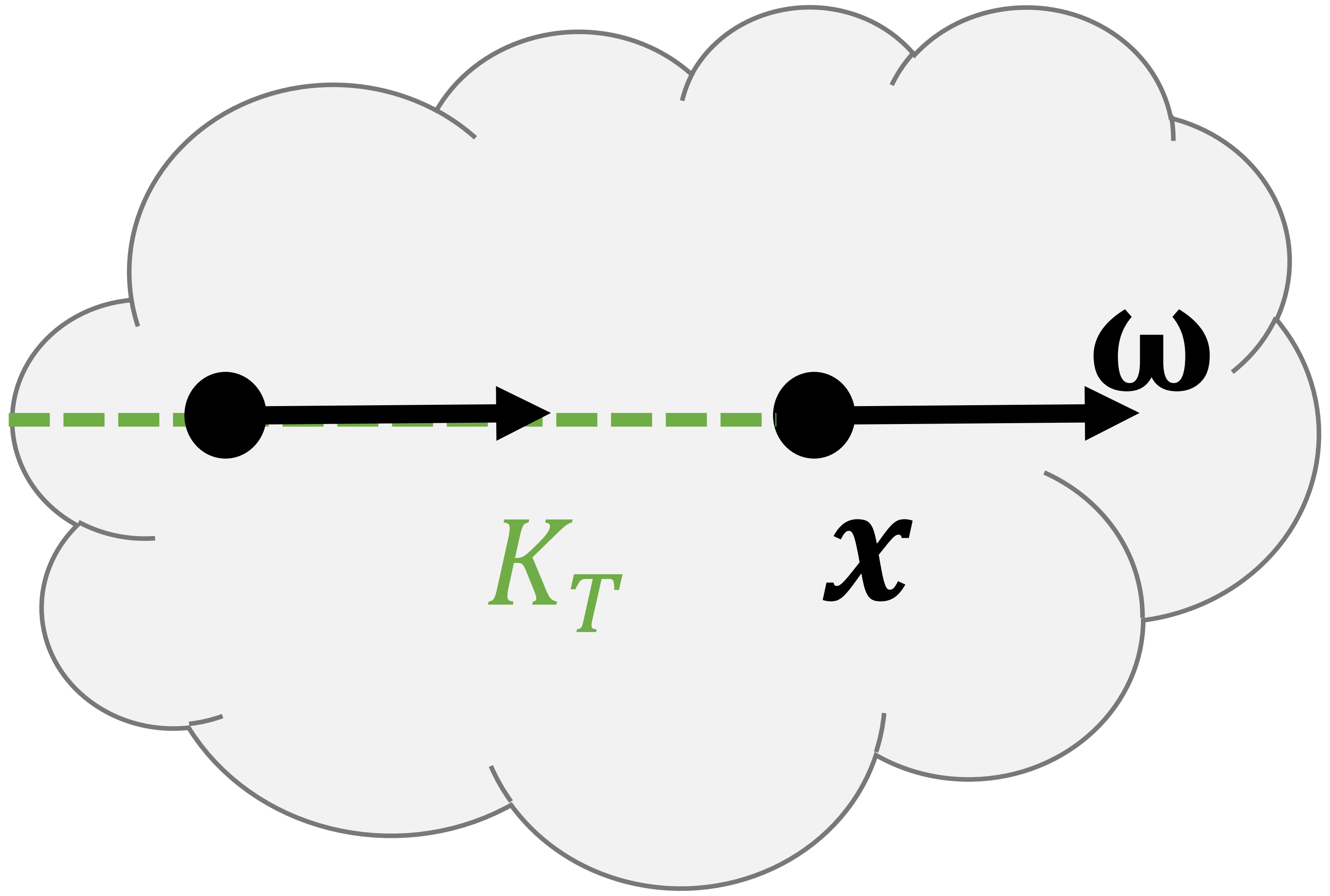
Transport operator — Collision operator — Source

$$L = K_T K_c L + Q$$

Radiative transfer equation (RTE)

in operator form





DIFFERENTIATING THE RTE

$$L = K_T K_c L + Q$$

Differentiating both sides

$$\partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q$$

DIFFERENTIATING THE RTE

$$L = K_T K_c L + Q$$

$$\partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q$$

Differentiating individual operators

DIFFERENTIATING THE COLLISION OPERATOR

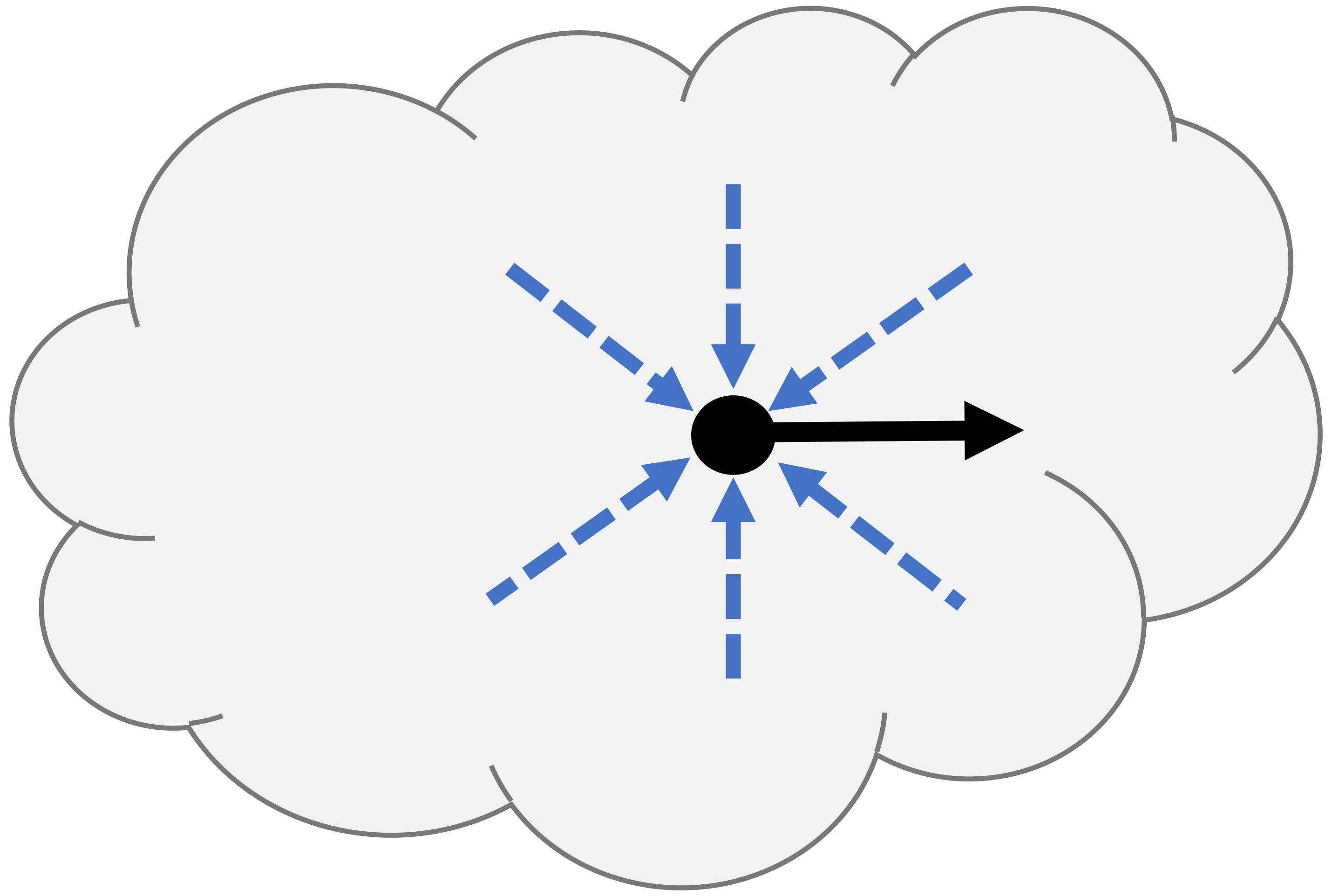
RTE: $L = K_T K_c L + Q$

(**x** omitted for notational simplicity)

$$(KcL)(\boldsymbol{\omega}) = \sigma_s \int_{\mathbb{S}^2} \overbrace{f_p(\boldsymbol{\omega_i}, \boldsymbol{\omega}) L(\boldsymbol{\omega_i})}^{f(\boldsymbol{\omega_i})} \mathrm{d}\boldsymbol{\omega_i}$$

Scattering coefficient

Phase function



$$\partial_\pi \int_{\mathbb{S}^2} f(\boldsymbol{\omega_i}) \mathrm{d}\boldsymbol{\omega_i} = \textbf{?}$$

Requires differentiating a spherical integral

DIFFERENTIATING THE COLLISION OPERATOR

$$(KcL)(\boldsymbol{\omega}) = \sigma_s \int_{\mathbb{S}^2} \overbrace{f_p(\boldsymbol{\omega_i}, \boldsymbol{\omega})\, L(\boldsymbol{\omega_i})}^{f(\boldsymbol{\omega_i})} \mathrm{d}\boldsymbol{\omega_i}$$

$$\partial_\pi \int_{\mathbb{S}^2} f(\boldsymbol{\omega_i}) \mathrm{d}\boldsymbol{\omega_i}$$

DIFFERENTIATING THE COLLISION OPERATOR

$$(KcL)(\boldsymbol{\omega}) = \sigma_s \int_{\mathbb{S}^2} \overbrace{f_p(\boldsymbol{\omega_i}, \boldsymbol{\omega})\, L(\boldsymbol{\omega_i})\mathrm{d}\boldsymbol{\omega_i}}^{f(\boldsymbol{\omega_i})}$$

$$\partial_\pi \int_{\mathbb{S}^2} f(\boldsymbol{\omega_i})\mathrm{d}\boldsymbol{\omega_i} = \int_{\mathbb{S}^2} \partial_\pi f(\boldsymbol{\omega_i})\mathrm{d}\boldsymbol{\omega_i} + \int_{\partial\mathbb{S}^2} \left\langle \boldsymbol{n}, \frac{\partial \boldsymbol{\omega_i}}{\partial \pi} \right\rangle \Delta f(\boldsymbol{\omega_i})\mathrm{d}\boldsymbol{\omega_i}$$

Interior integral

Boundary integral

By applying Reynolds transport theorem

(largely identical to the differentiation of the rendering equation)

OTHER TERMS IN THE RTE

$$L = K_T K_c L + Q$$

Transport operator

$$(K_T K_c L)(x, \omega) = \int_0^D T(x', x)\,(K_c L)(x', \omega)\,d\tau$$

Transmittance

Source

$$Q = T(\boldsymbol{x_0}, \boldsymbol{x}) L_s(\boldsymbol{x_0}, \boldsymbol{\omega})$$

OTHER TERMS IN THE RTE

$$L = K_T K_c L + Q$$

Transport operator (can be differentiated using Leibniz's rule)

$$(K_T K_c L)(x,\omega) = \int_0^D T(x',x)\,(K_c L)(x',\omega)\,d\tau$$

Transmittance

Source

$$Q = T(\boldsymbol{x_0},\boldsymbol{x})L_s(\boldsymbol{x_0},\boldsymbol{\omega})$$

DIFFERENTIAL RADIATIVE TRANSFER EQUATION

$$\dot{L}(\boldsymbol{x},\omega) = \int_0^D T(\boldsymbol{x}',\boldsymbol{x})\Big[\sigma_{\mathrm{s}}(\boldsymbol{x}')\,\dot{L}^{\mathrm{ins}}(\boldsymbol{x}',\omega) + \big(\dot{\sigma}_{\mathrm{s}}(\boldsymbol{x}') - \Sigma_{\mathrm{t}}(\boldsymbol{x},\omega,\tau)\,\sigma_{\mathrm{s}}(\boldsymbol{x}')\big)\,L^{\mathrm{ins}}(\boldsymbol{x}',\omega)\Big]\,\mathrm{d}\tau$$
$$+\, T(\boldsymbol{x}_0,\boldsymbol{x})\Big[-\big(\Sigma_{\mathrm{t}}(\boldsymbol{x},\omega,D) + \dot{D}\,\sigma_{\mathrm{t}}(\boldsymbol{x}_0)\big)\,L_{\mathrm{s}}(\boldsymbol{x}_0,\omega) + \dot{L}_{\mathrm{s}}(\boldsymbol{x}_0,\omega) + \dot{D}\,\sigma_{\mathrm{s}}(\boldsymbol{x}_0)\,L^{\mathrm{ins}}(\boldsymbol{x}_0,\omega)\Big],$$

where Σ_{t} is defined in Eq. (17), $\dot{L}^{\mathrm{ins}}$ follows Eq. (22), and $\dot{L}_{\mathrm{s}} = \dot{L}_{\mathrm{s}}^{\mathrm{r}} + \dot{L}_{\mathrm{s}}^{\mathrm{e}}$ with $\dot{L}_{\mathrm{s}}^{\mathrm{r}}$ given by Eq. (29)

This is Eq. (32) of the work by Zhang et al. [2019]

DIFFERENTIAL RTE, OPERATOR FORM

$$L = K_T K_c L + Q$$

$$\partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q$$

DIFFERENTIAL RTE, OPERATOR FORM

$$L = K_T K_c L + Q$$

$$\partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q$$

$$\begin{pmatrix} \partial_\pi L \\ L \end{pmatrix} = \begin{pmatrix} K_T K_c & K_* \\ 0 & K_T K_c \end{pmatrix} \begin{pmatrix} \partial_\pi L \\ L \end{pmatrix} + \begin{pmatrix} \partial_\pi Q \\ Q \end{pmatrix}$$

Differential radiative transfer equation

DIFFERENTIAL RTE, OPERATOR FORM

$$L = K_T K_c L + Q$$

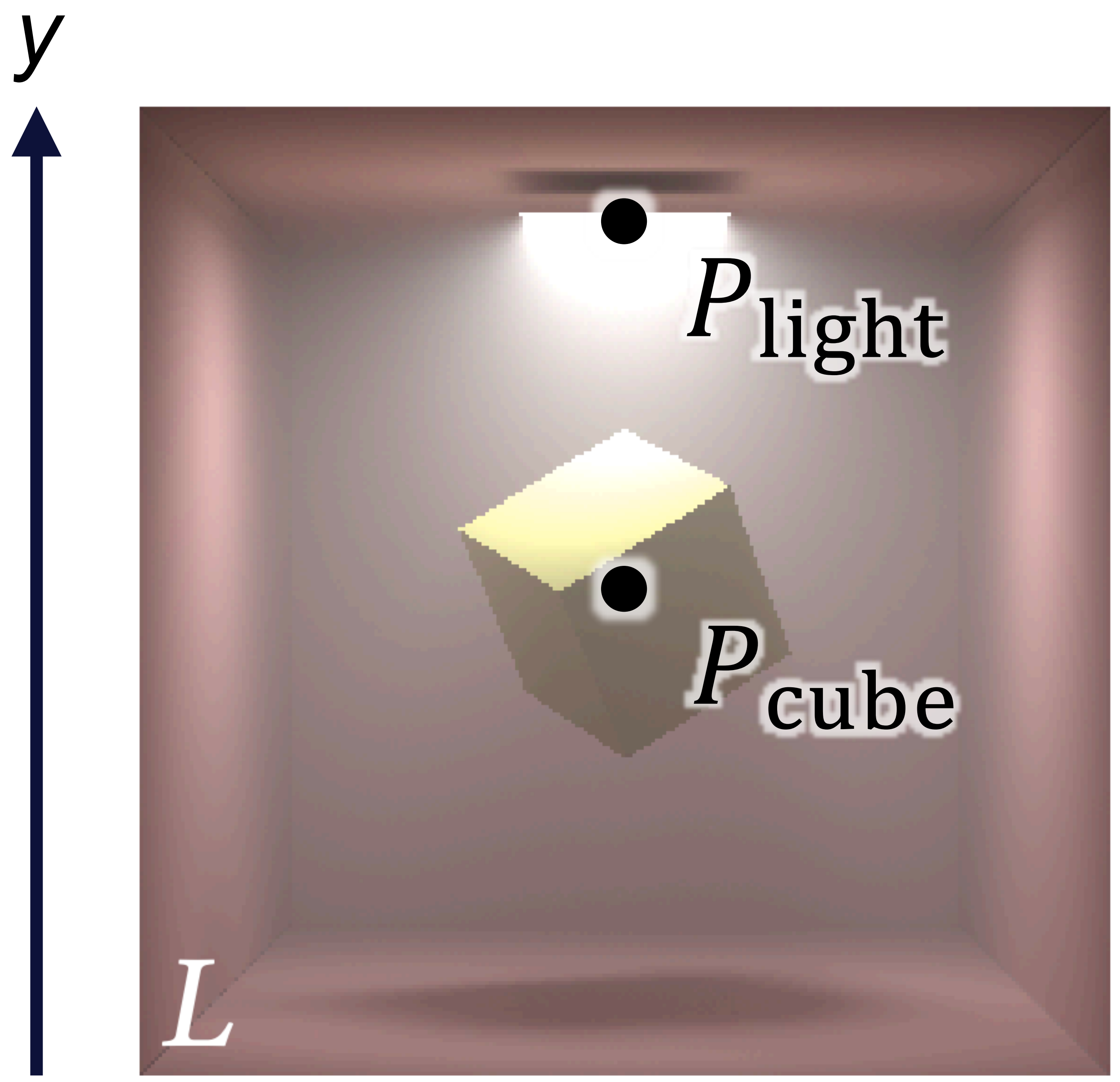
$$\partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q$$

Captures the boundary integrals

$$\begin{pmatrix} \partial_\pi L \\ L \end{pmatrix} = \begin{pmatrix} K_T K_c & K_* \\ 0 & K_T K_c \end{pmatrix} \begin{pmatrix} \partial_\pi L \\ L \end{pmatrix} + \begin{pmatrix} \partial_\pi Q \\ Q \end{pmatrix}$$

Differential radiative transfer equation

SIGNIFICANCE OF THE BOUNDARY INTEGRAL



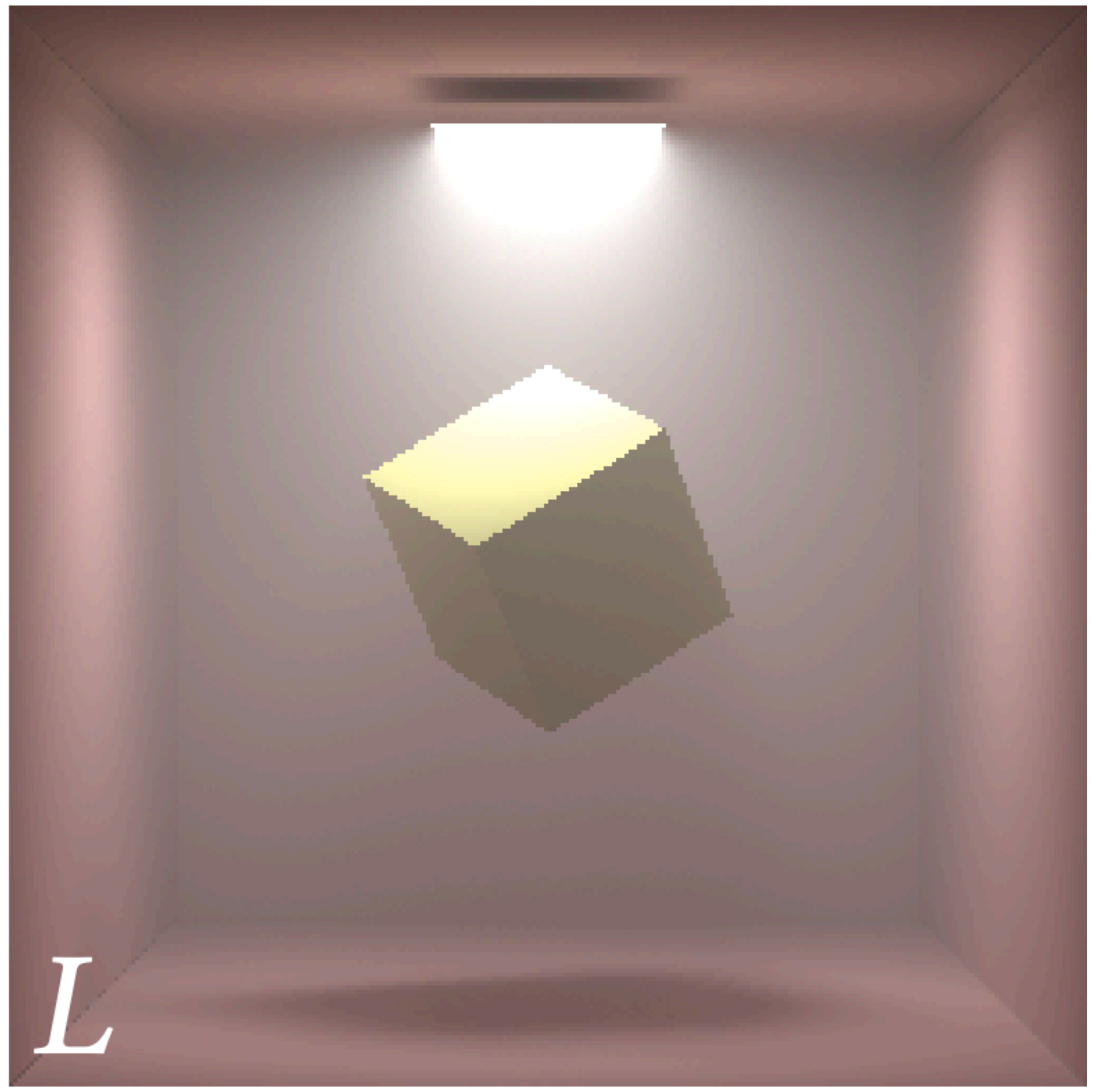
Original image

$$\mathbf{P}_{\text{light}}(\pi) = \mathbf{P}_0 + \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix}$$

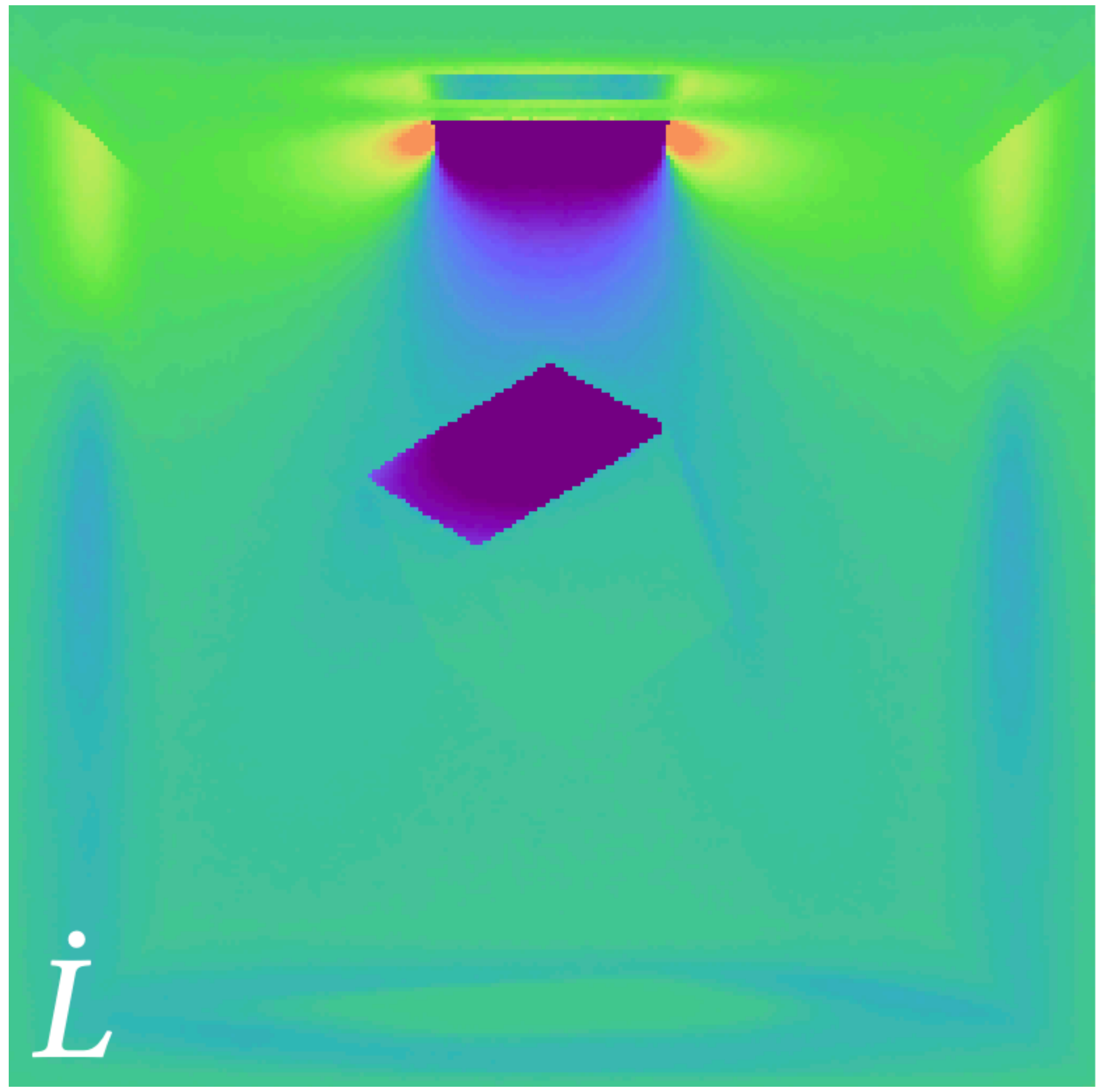
$$\mathbf{P}_{\text{cube}}(\pi) = \mathbf{P}_1 + \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix}$$

Constant initial positions

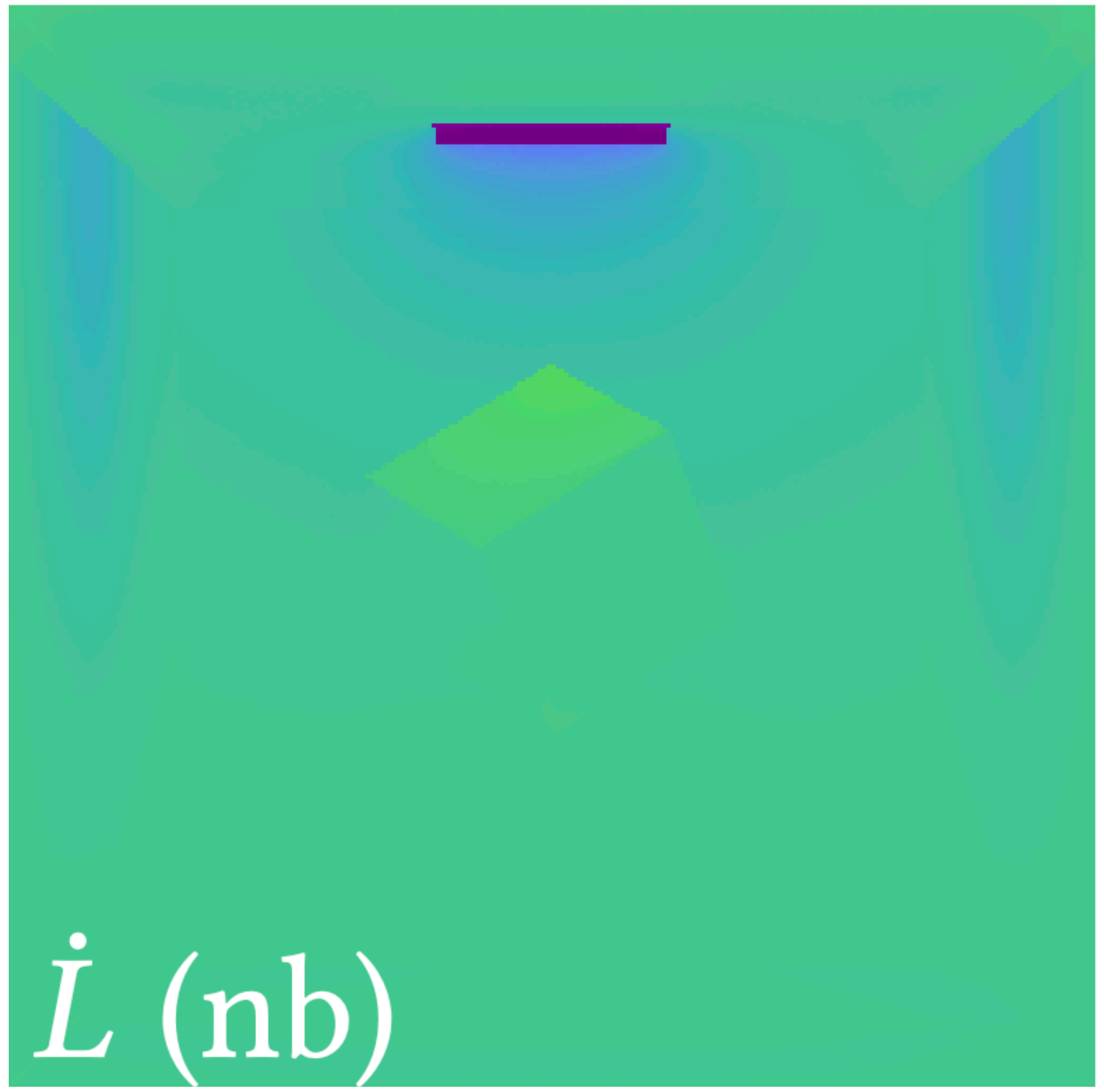
SIGNIFICANCE OF THE BOUNDARY INTEGRAL



Original image

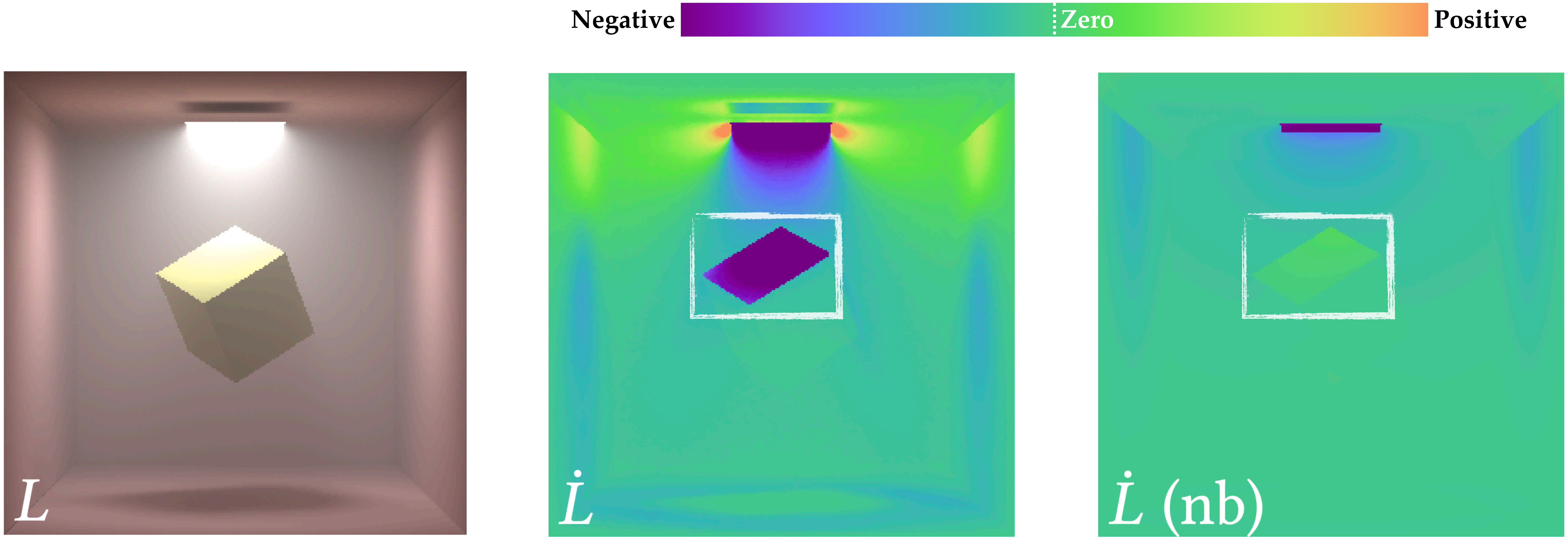


Derivative image



Derivative image
(w/o boundary integral)

SIGNIFICANCE OF THE BOUNDARY INTEGRAL

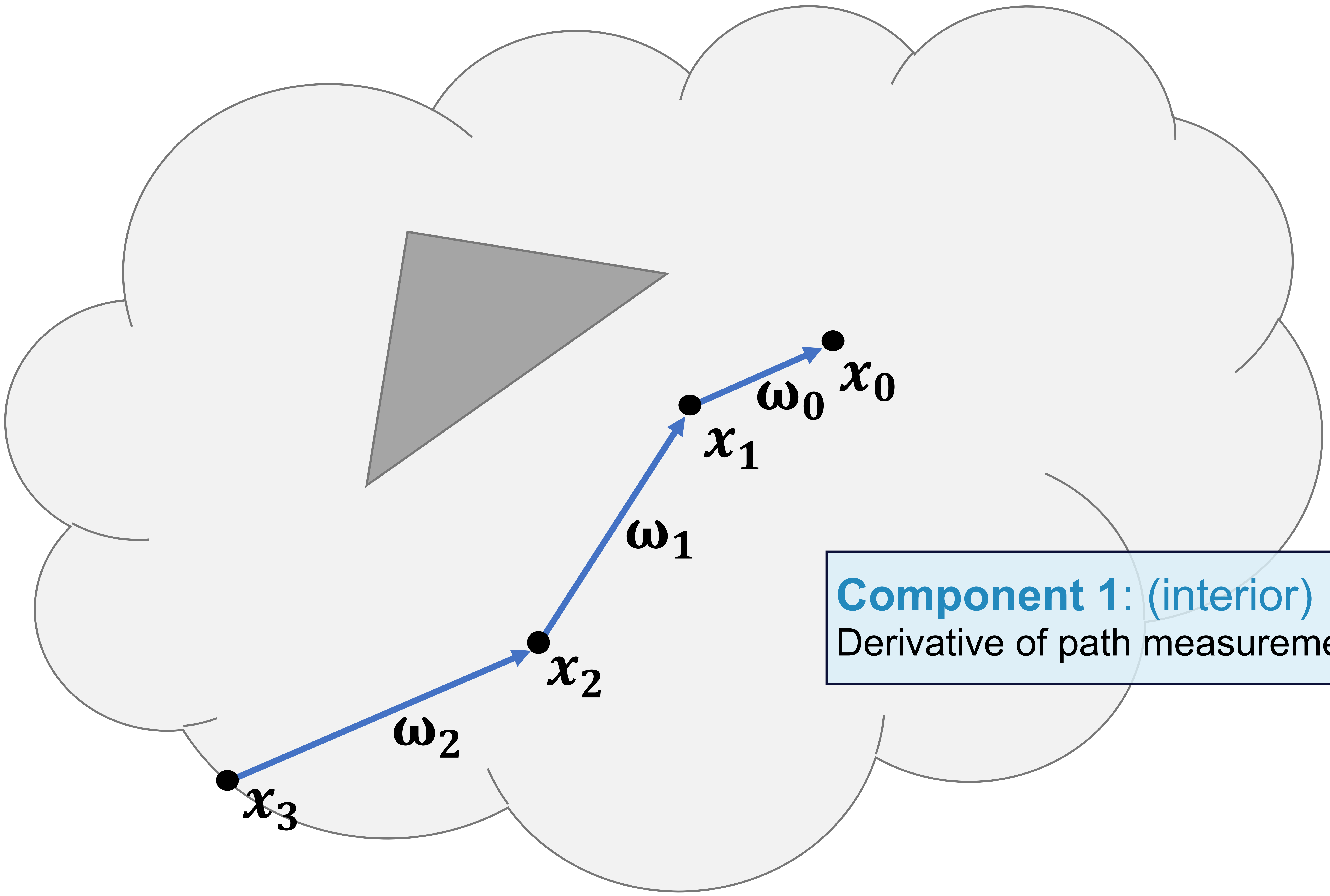


Original image

Derivative image

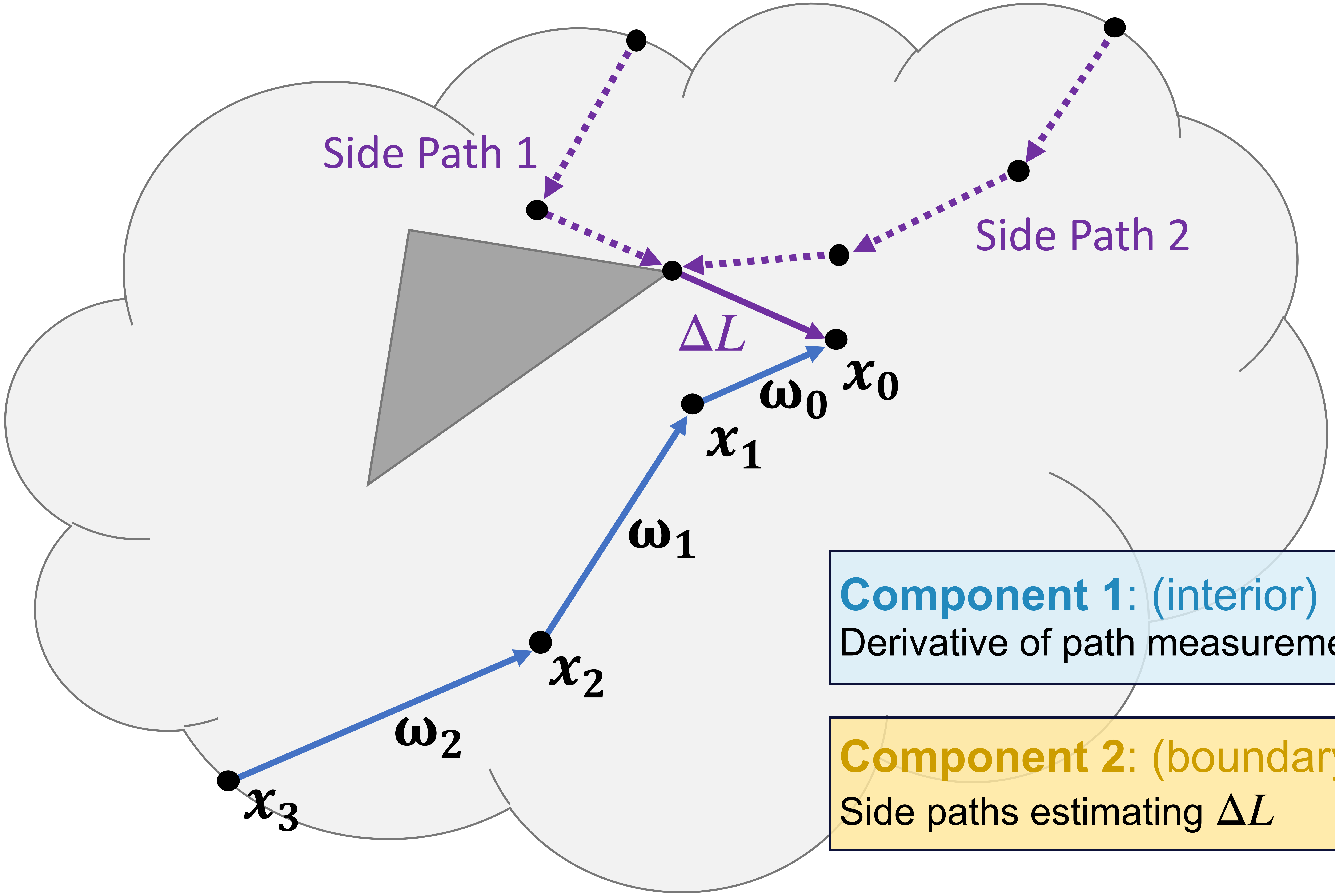
Derivative image
(w/o boundary integral)

DIFFERENTIABLE VOLUMETRIC PATH TRACING



Component 1: (interior)
Derivative of path measurement contribution

DIFFERENTIABLE VOLUMETRIC PATH TRACING



Component 1: (interior)
Derivative of path measurement contribution

Component 2: (boundary)
Side paths estimating ΔL

INVERSE-RENDERING RESULTS

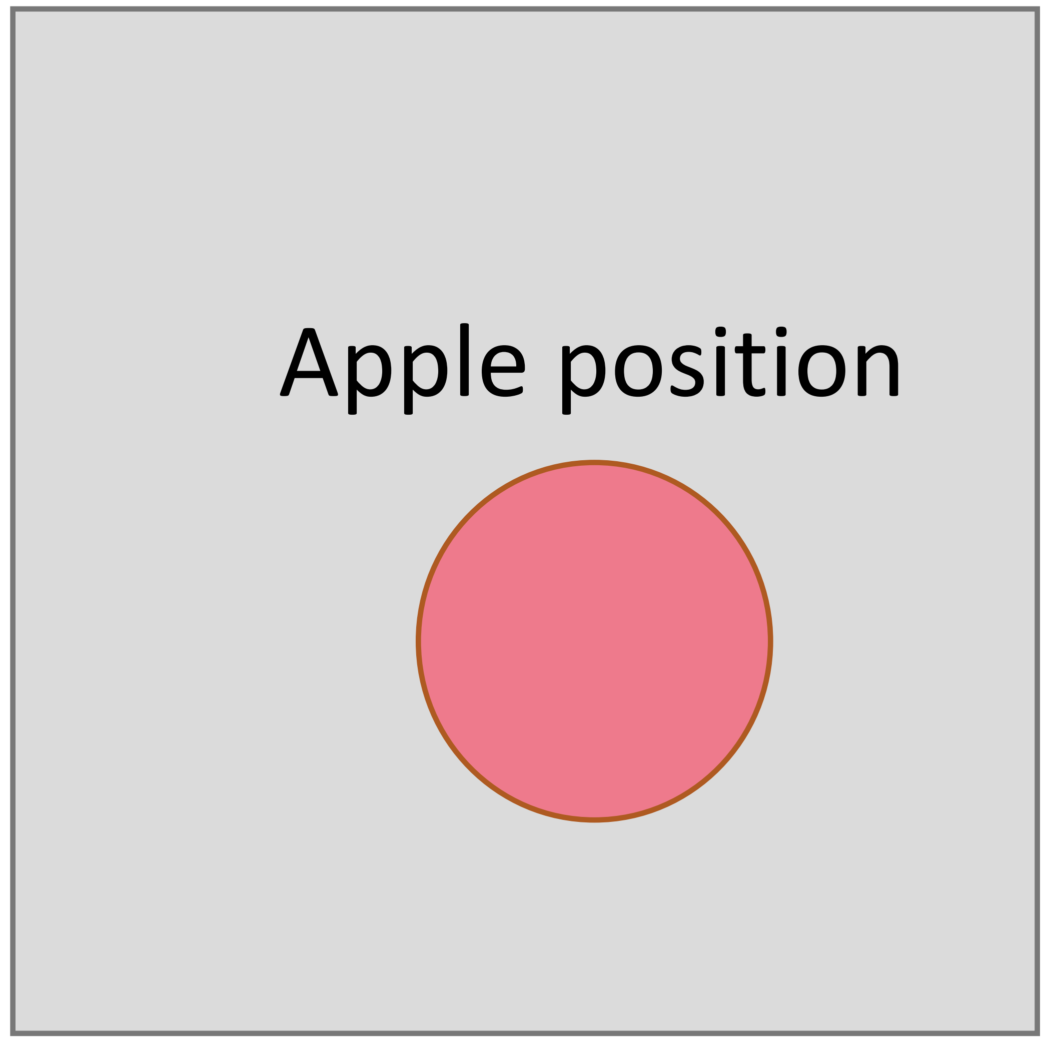
- Scene configurations
  - Participating media
  - Changing geometry

- Optimization
  - Using only image loss (L2)

INVERSE-RENDERING RESULTS

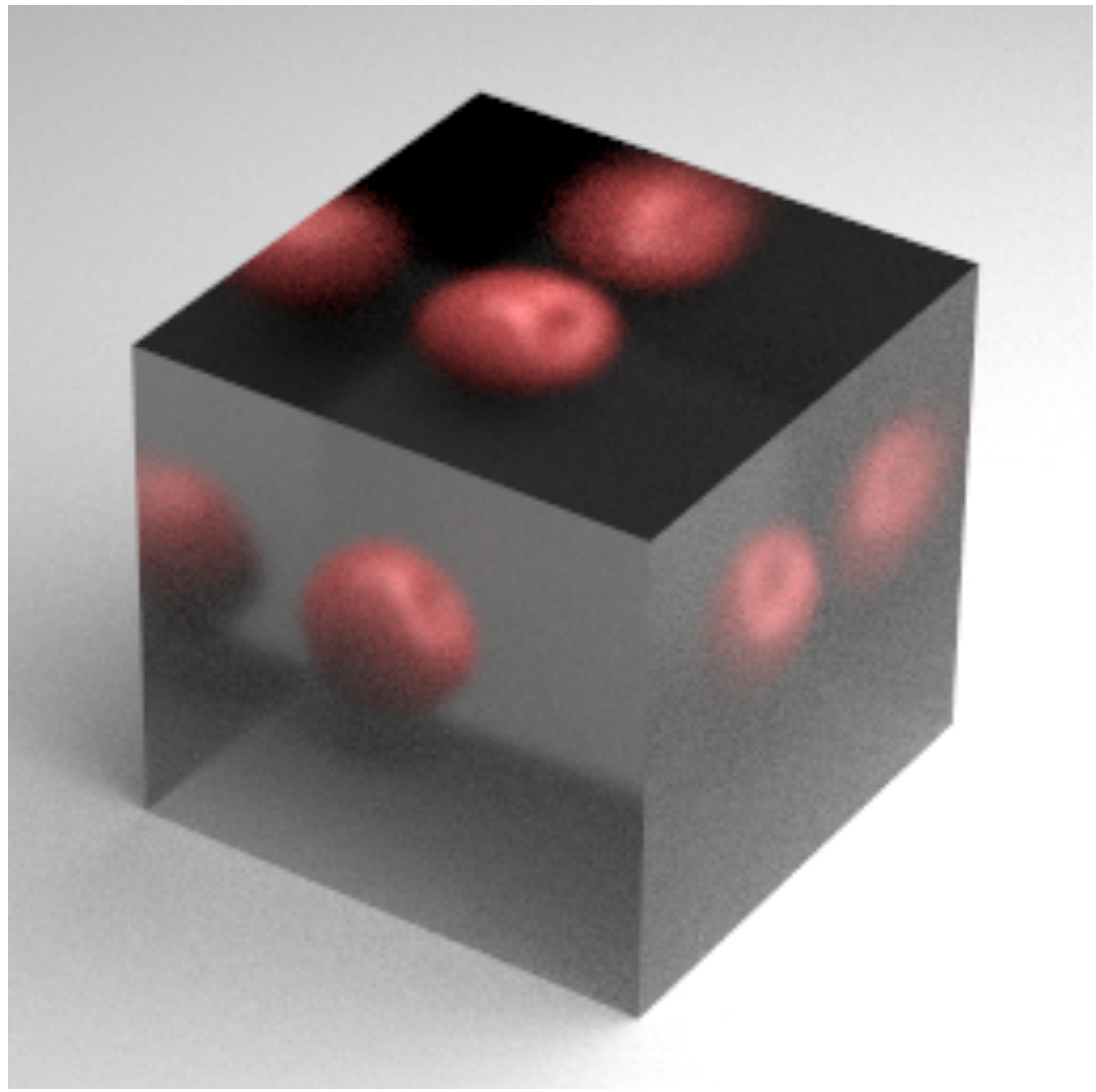
Apple in a box

Parameters

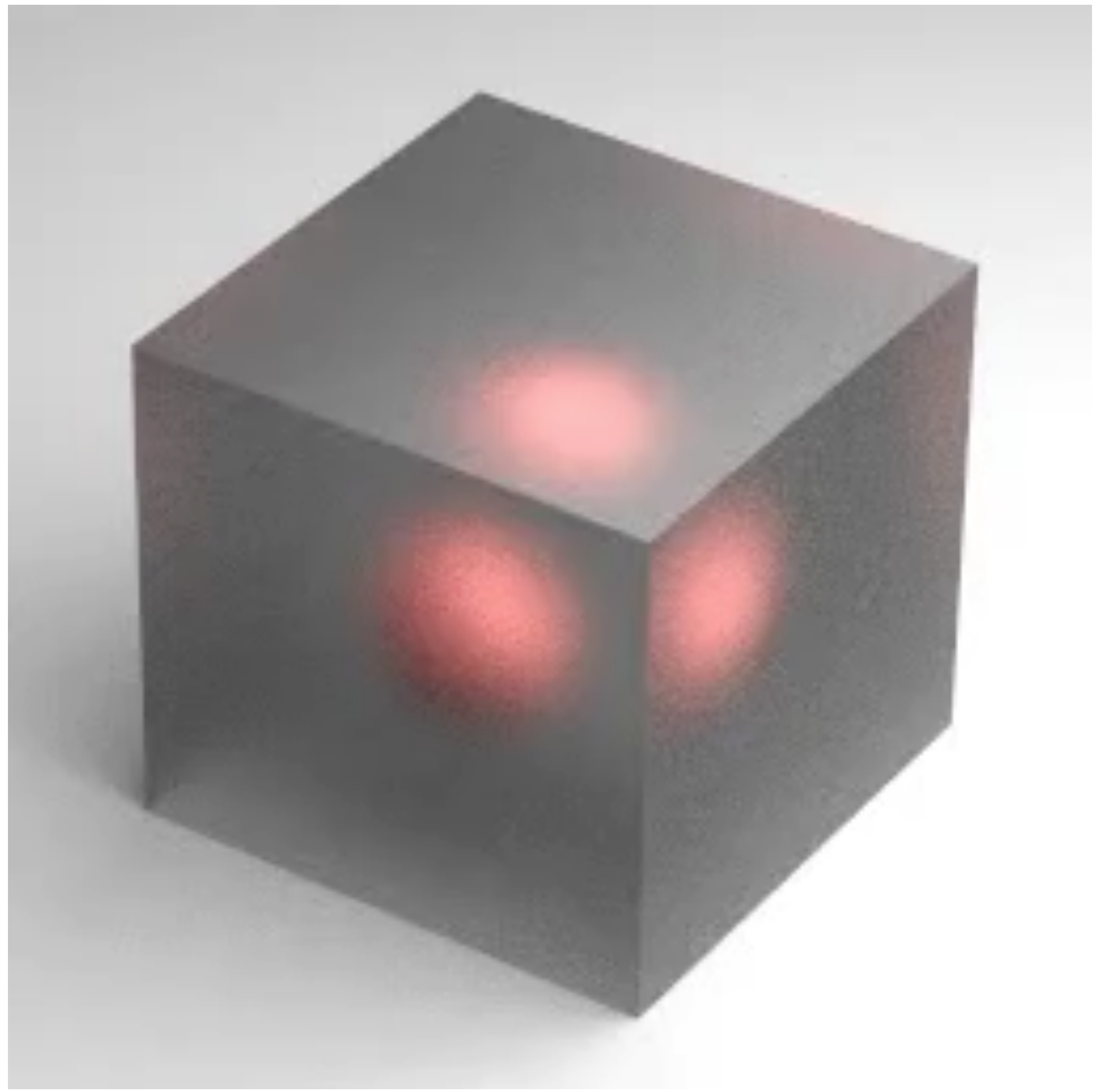


Cube roughness

Target



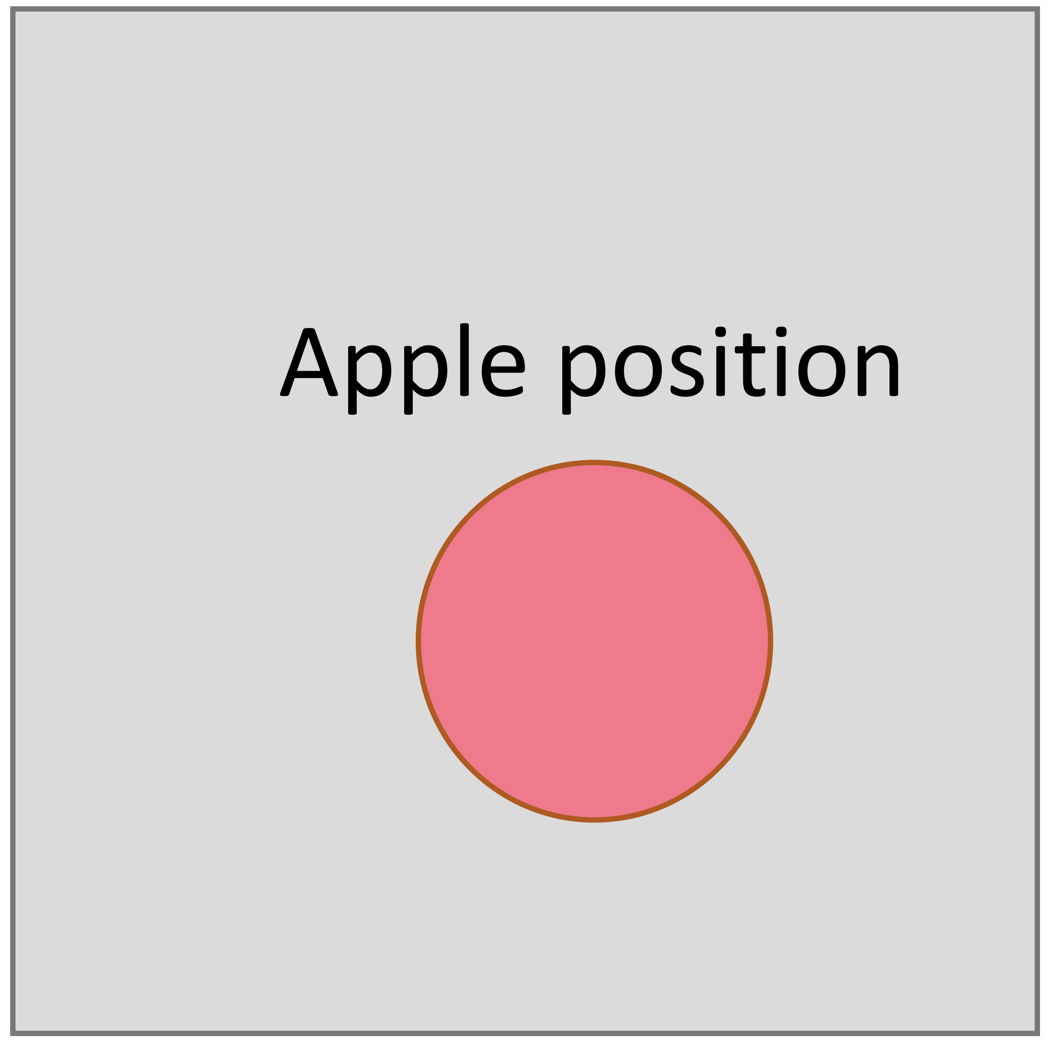
Optimization process



INVERSE-RENDERING RESULTS

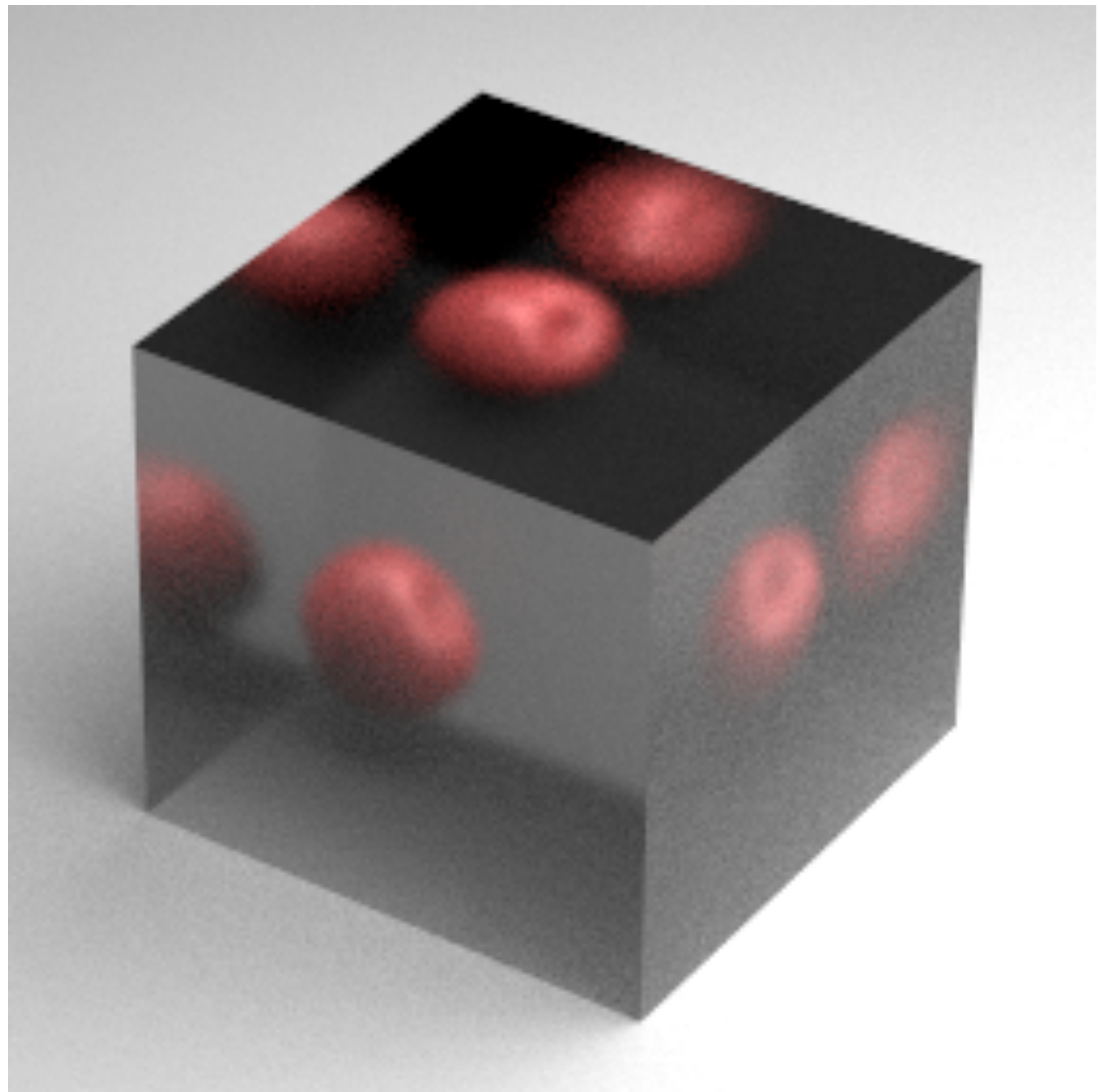
Apple in a box

Parameters

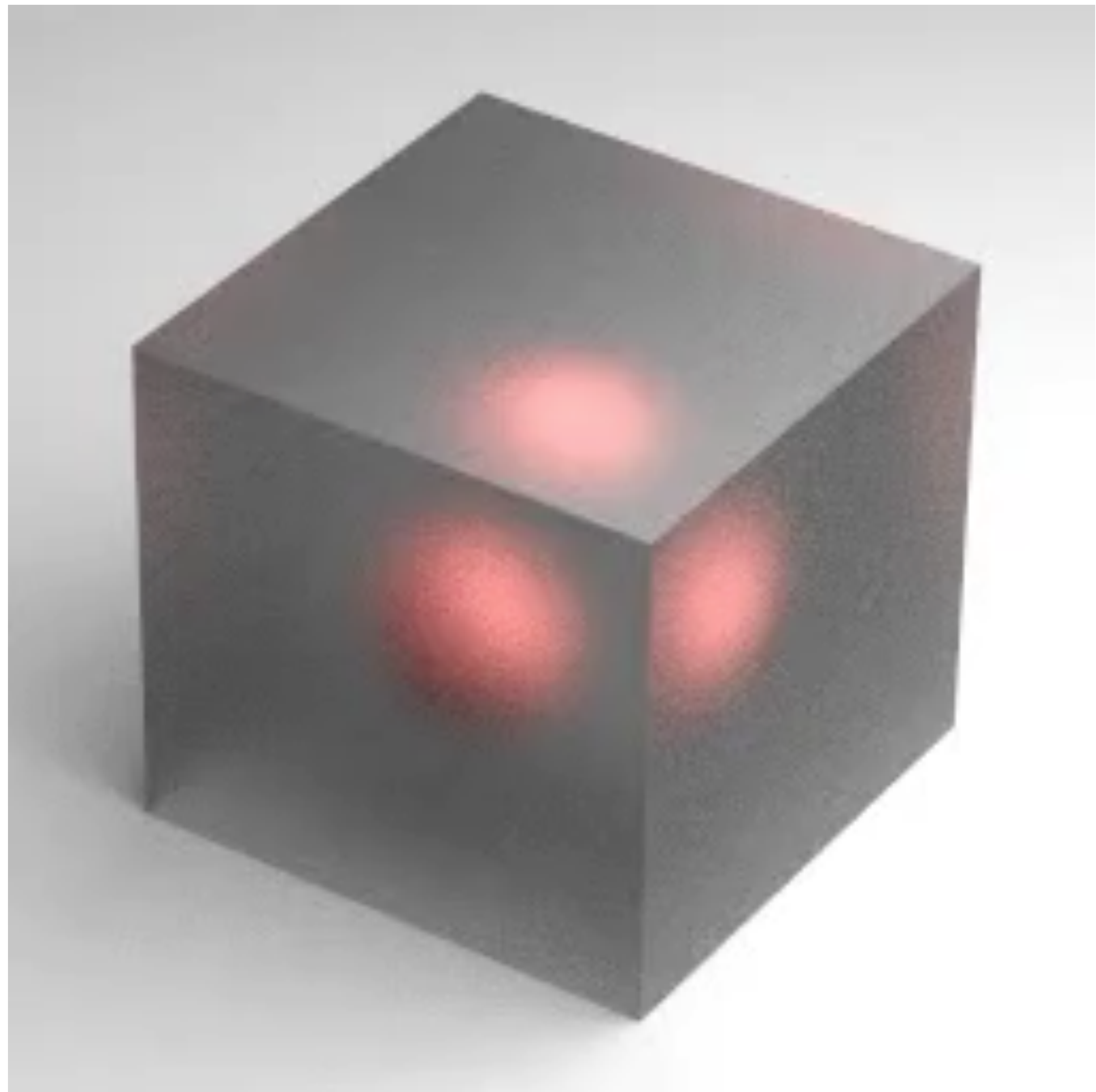


Cube roughness

Target

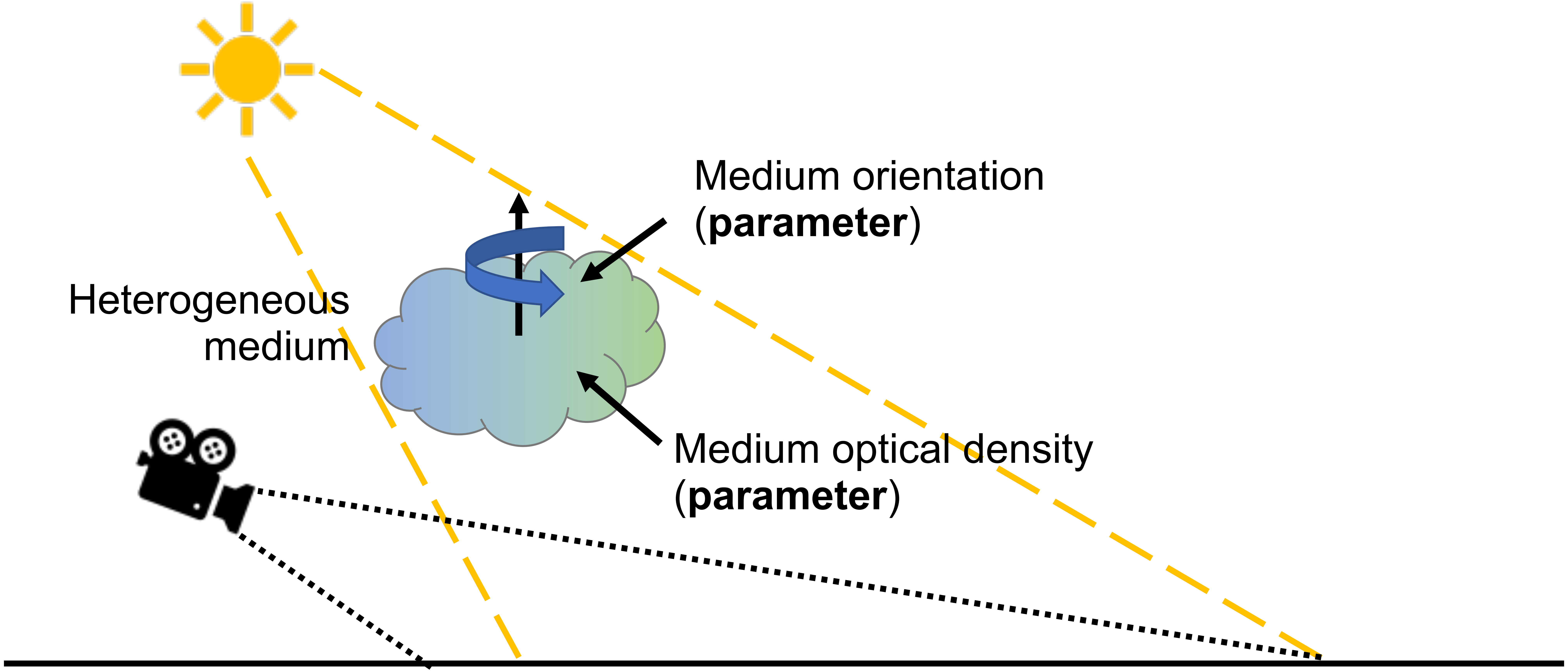


Optimization process



INVERSE-RENDERING RESULTS

Non-line-of-sight inverse rendering



INVERSE-RENDERING RESULTS

Non-line-of-sight inverse rendering

Target

Optimization process

Different view

INVERSE-RENDERING RESULTS

Non-line-of-sight inverse rendering

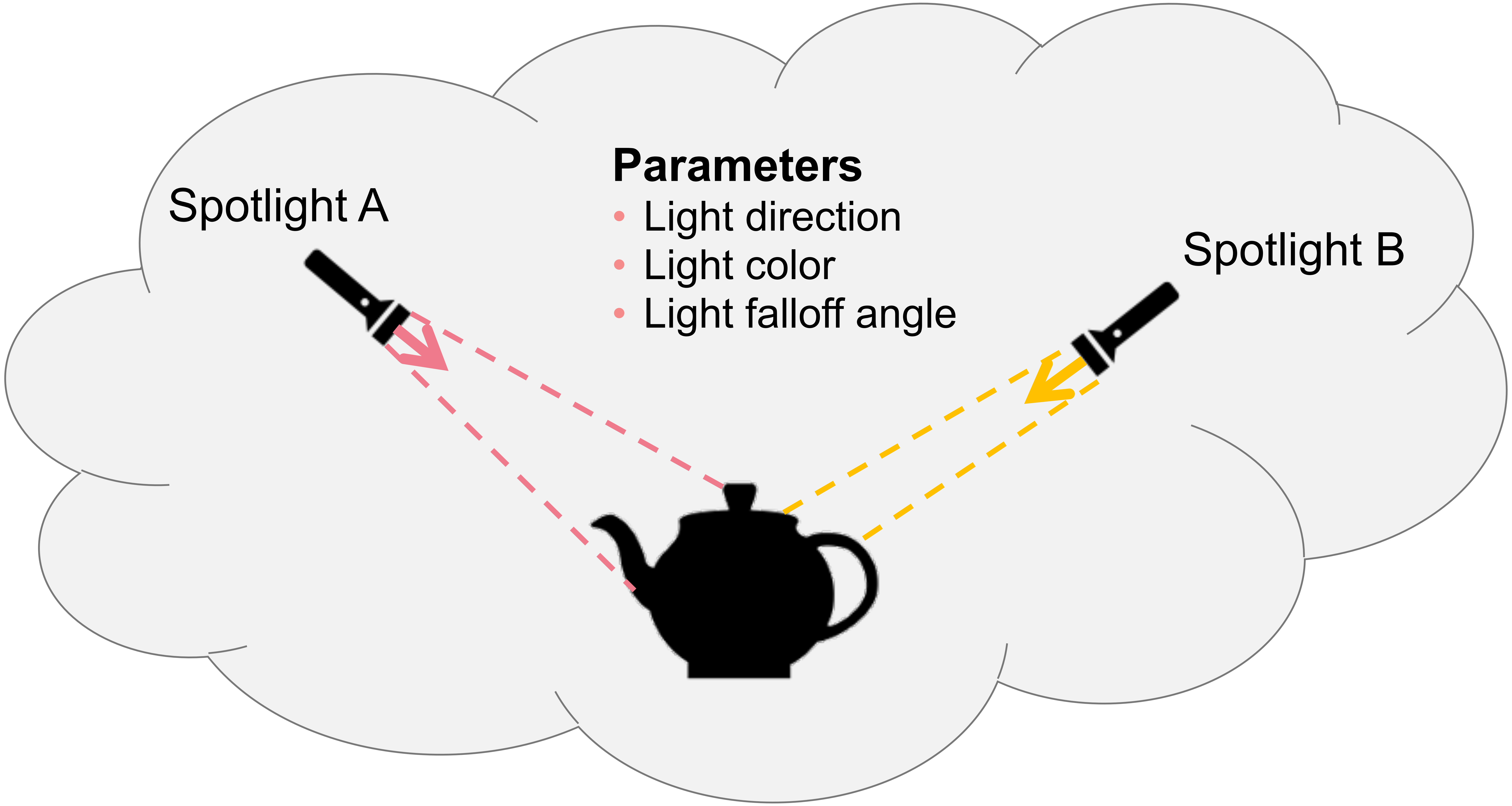
Target

Optimization process

Different view

INVERSE-RENDERING RESULTS

Design-inspired inverse rendering

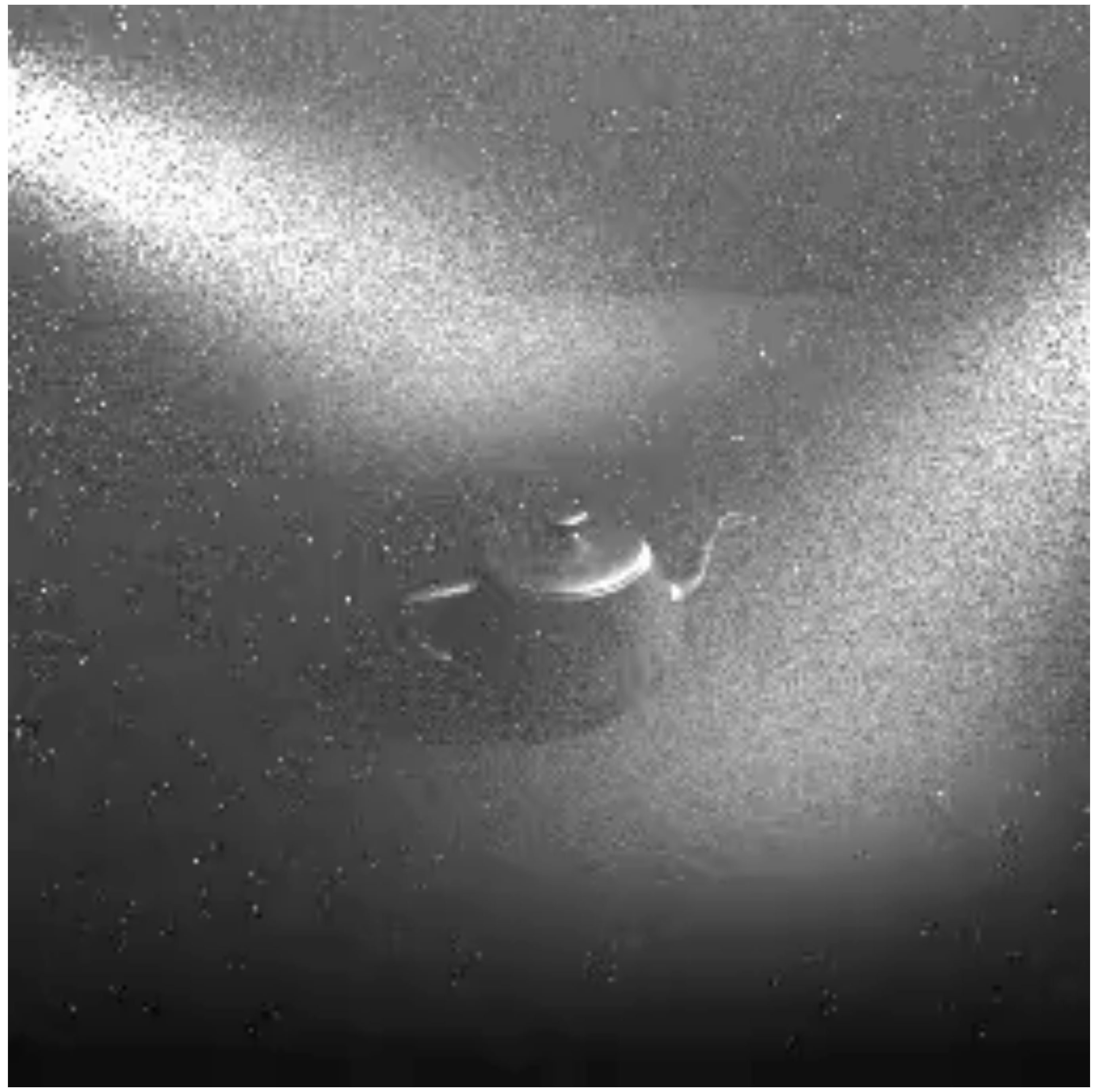


INVERSE-RENDERING RESULTS

Design-inspired inverse rendering

Target

Optimization process

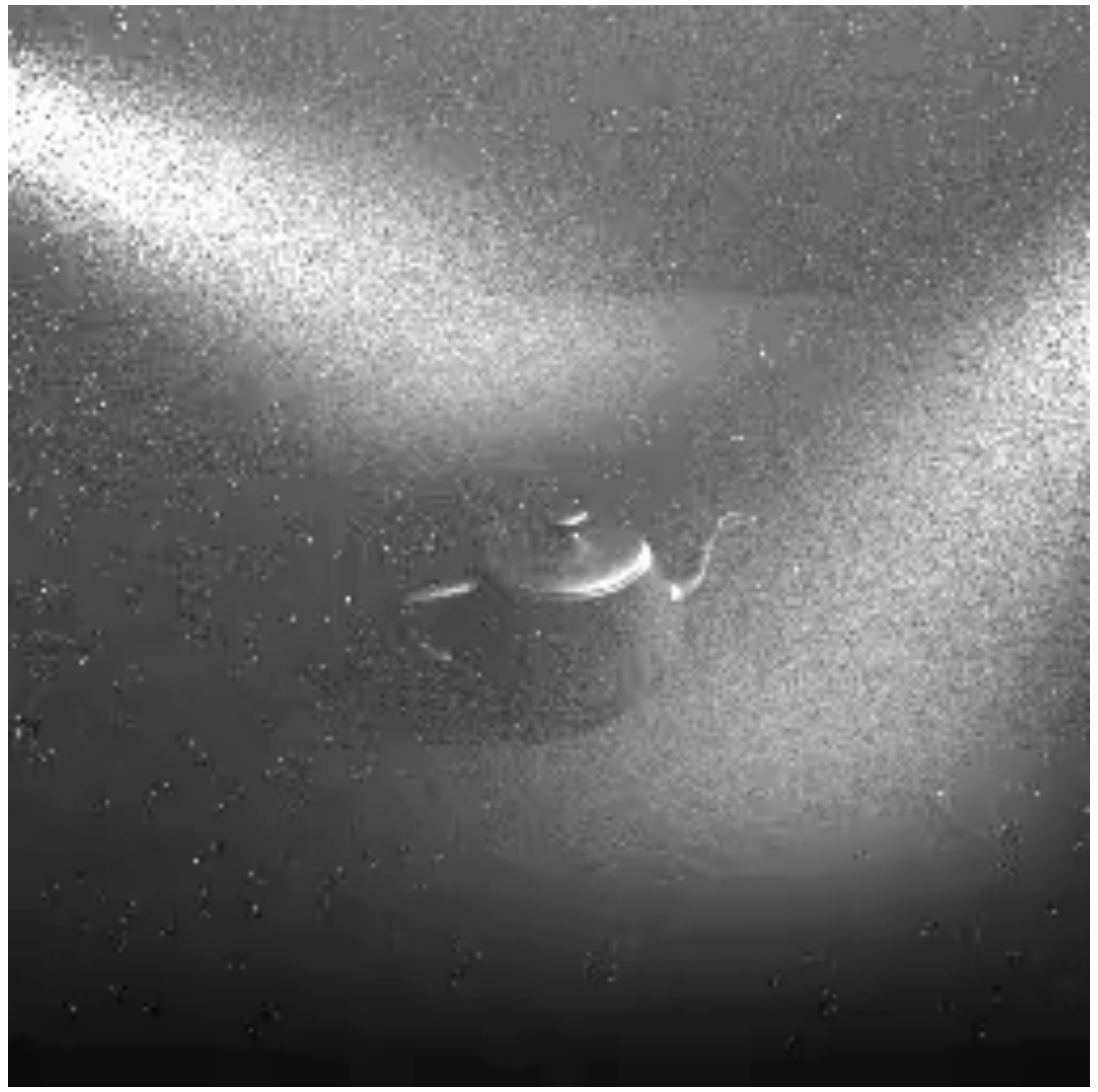


INVERSE-RENDERING RESULTS

Design-inspired inverse rendering

Target

Optimization process



CHALLENGES

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i}) f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})} \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

Interior integral

Boundary integral

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i}) \, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

CHALLENGES

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i}) f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})} \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

Differential rendering equation

Interior integral — Boundary integral

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i}) \, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

- Complex scenes
  - Discontinuity points (e.g., $\partial\mathbb{S}^2$) can be expensive to detect

CHALLENGES

Rendering equation

$$L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \overbrace{L_\mathrm{i}(\boldsymbol{\omega}_\mathrm{i}) f_s(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})}^{f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i})} \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

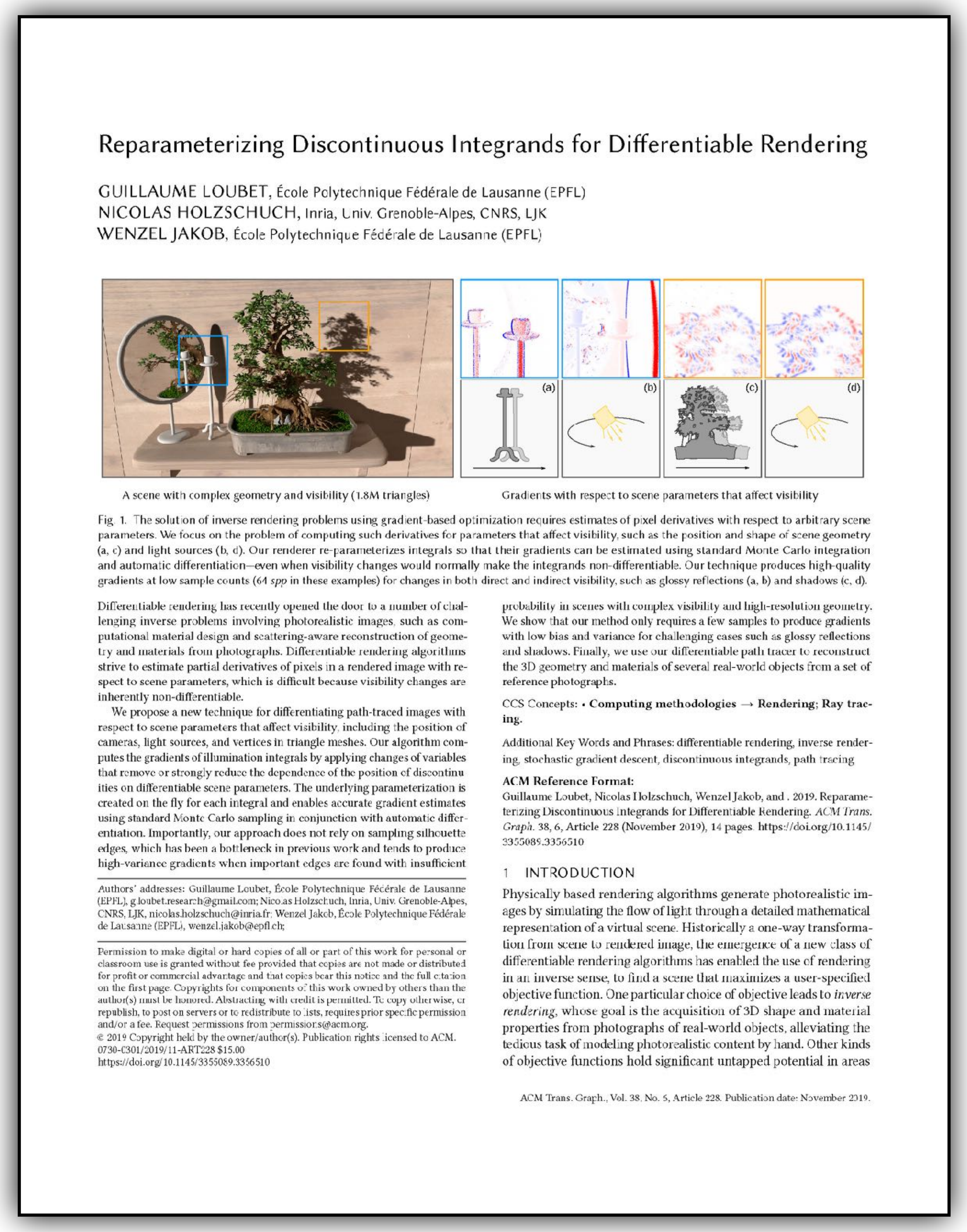
Differential rendering equation

Interior integral | Boundary integral

$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i}) \, \Delta f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\ell(\boldsymbol{\omega}_\mathrm{i}) + \frac{\mathrm{d}}{\mathrm{d}\pi} L_\mathrm{e}(\boldsymbol{\omega}_\mathrm{o})$$

- Complex scenes
  - Discontinuity points (e.g., $\partial\mathbb{S}^2$) can be expensive to detect
- Scaling out to millions of parameters

ANOTHER WAY OF DEALING WITH EDGES



Reparameterizing Discontinuous Integrands for Differentiable Rendering

GUILLAUME LOUBET, École Polytechnique Fédérale de Lausanne (EPFL)
NICOLAS HOLZSCHUCH, Inria, Univ. Grenoble-Alpes, CNRS, LJK
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL)

(a) (b) (c) (d)

A scene with complex geometry and visibility (1.8M triangles)

Gradients with respect to scene parameters that affect visibility

Fig. 1. The solution of inverse rendering problems using gradient-based optimization requires estimates of pixel derivatives with respect to arbitrary scene parameters. We focus on the problem of computing such derivatives for parameters that affect visibility, such as the position and shape of scene geometry (a, c) and light sources (b, d). Our renderer re-parameterizes integrals so that their gradients can be estimated using standard Monte Carlo integration and automatic differentiation—even when visibility changes would normally make the integrands non-differentiable. Our technique produces high-quality gradients at low sample counts (64 *spp* in these examples) for changes in both direct and indirect visibility, such as glossy reflections (a, b) and shadows (c, d).

Differentiable rendering has recently opened the door to a number of challenging inverse problems involving photorealistic images, such as computational material design and scattering-aware reconstruction of geometry and materials from photographs. Differentiable rendering algorithms strive to estimate partial derivatives of pixels in a rendered image with respect to scene parameters, which is difficult because visibility changes are inherently non-differentiable.

We propose a new technique for differentiating path-traced images with respect to scene parameters that affect visibility, including the position of cameras, light sources, and vertices in triangle meshes. Our algorithm computes the gradients of illumination integrals by applying changes of variables that remove or strongly reduce the dependence of the position of discontinuities on differentiable scene parameters. The underlying parameterization is created on the fly for each integral and enables accurate gradient estimates using standard Monte Carlo sampling in conjunction with automatic differentiation. Importantly, our approach does not rely on sampling silhouette edges, which has been a bottleneck in previous work and tends to produce high-variance gradients when important edges are found with insufficient probability in scenes with complex visibility and high-resolution geometry. We show that our method only requires a few samples to produce gradients with low bias and variance for challenging cases such as glossy reflections and shadows. Finally, we use our differentiable path tracer to reconstruct the 3D geometry and materials of several real-world objects from a set of reference photographs.

CCS Concepts: • **Computing methodologies → Rendering; Ray tracing.**

Additional Key Words and Phrases: differentiable rendering, inverse rendering, stochastic gradient descent, discontinuous integrands, path tracing

ACM Reference Format:
Guillaume Loubet, Nicolas Holzschuch, Wenzel Jakob, and . 2019. Reparameterizing Discontinuous Integrands for Differentiable Rendering. *ACM Trans. Graph.* 38, 6, Article 228 (November 2019), 14 pages. https://doi.org/10.1145/3355089.3356510

Authors' addresses: Guillaume Loubet, École Polytechnique Fédérale de Lausanne (EPFL), g.loubet.research@gmail.com; Nicolas Holzschuch, Inria, Univ. Grenoble-Alpes, CNRS, LJK, nicolas.holzschuch@inria.fr; Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL), wenzel.jakob@epfl.ch;

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2019/11-ART228 $15.00
https://doi.org/10.1145/3355089.3356510

1 INTRODUCTION

Physically based rendering algorithms generate photorealistic images by simulating the flow of light through a detailed mathematical representation of a virtual scene. Historically a one-way transformation from scene to rendered image, the emergence of a new class of differentiable rendering algorithms has enabled the use of rendering in an inverse sense, to find a scene that maximizes a user-specified objective function. One particular choice of objective leads to *inverse rendering*, whose goal is the acquisition of 3D shape and material properties from photographs of real-world objects, alleviating the tedious task of modeling photorealistic content by hand. Other kinds of objective functions hold significant untapped potential in areas

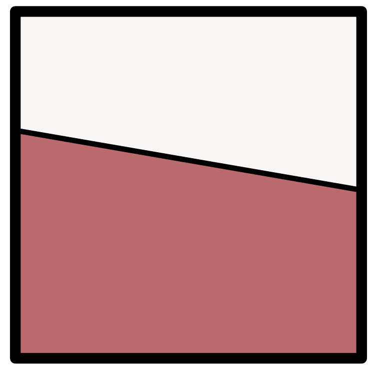
ACM Trans. Graph., Vol. 38, No. 6, Article 228. Publication date: November 2019.

Reparameterizing Discontinuous Integrals for Differentiable Rendering

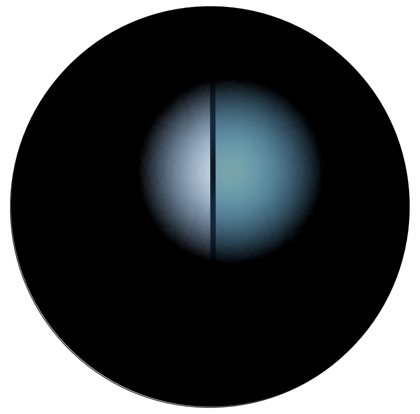
Guillaume Loubet, Nicolas Holzschuch, Wenzel Jakob

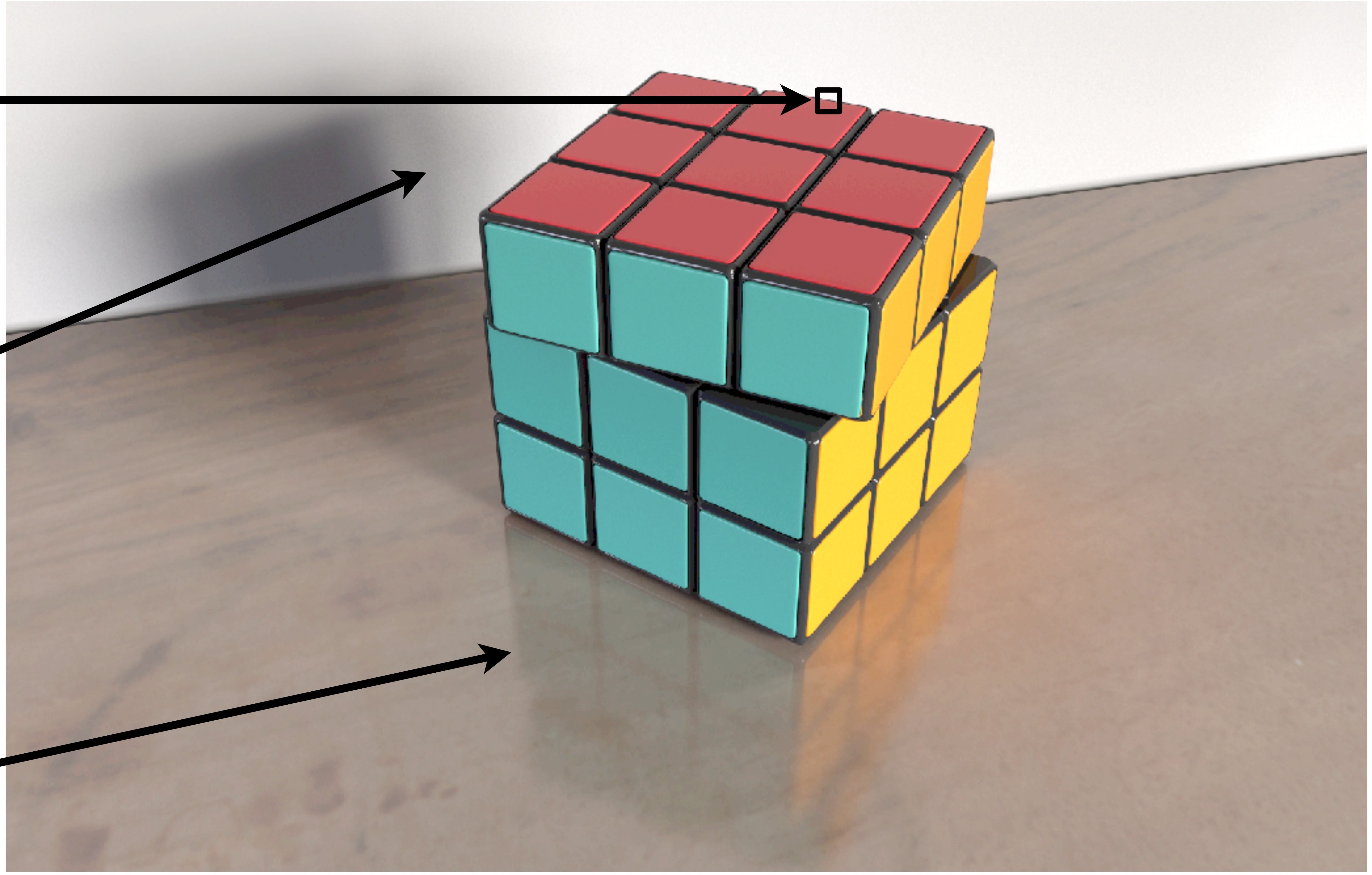
SIGGRAPH Asia 2019

MOVING DISCONTINUITIES

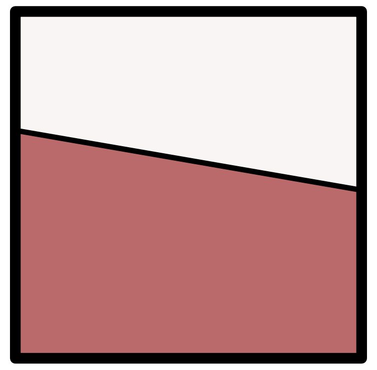
Pixel integrals $\iint$  $\mathrm{d}x\,\mathrm{d}y$

Light integrals $\int$  $\mathrm{d}\omega$

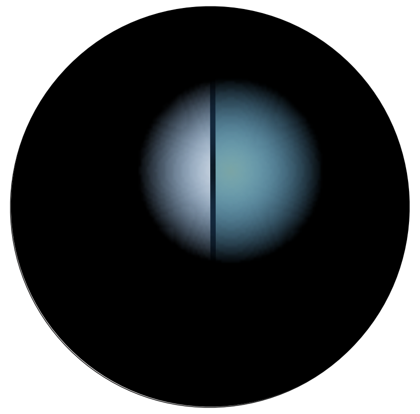
BSDF integrals $\int$  $\mathrm{d}\omega$

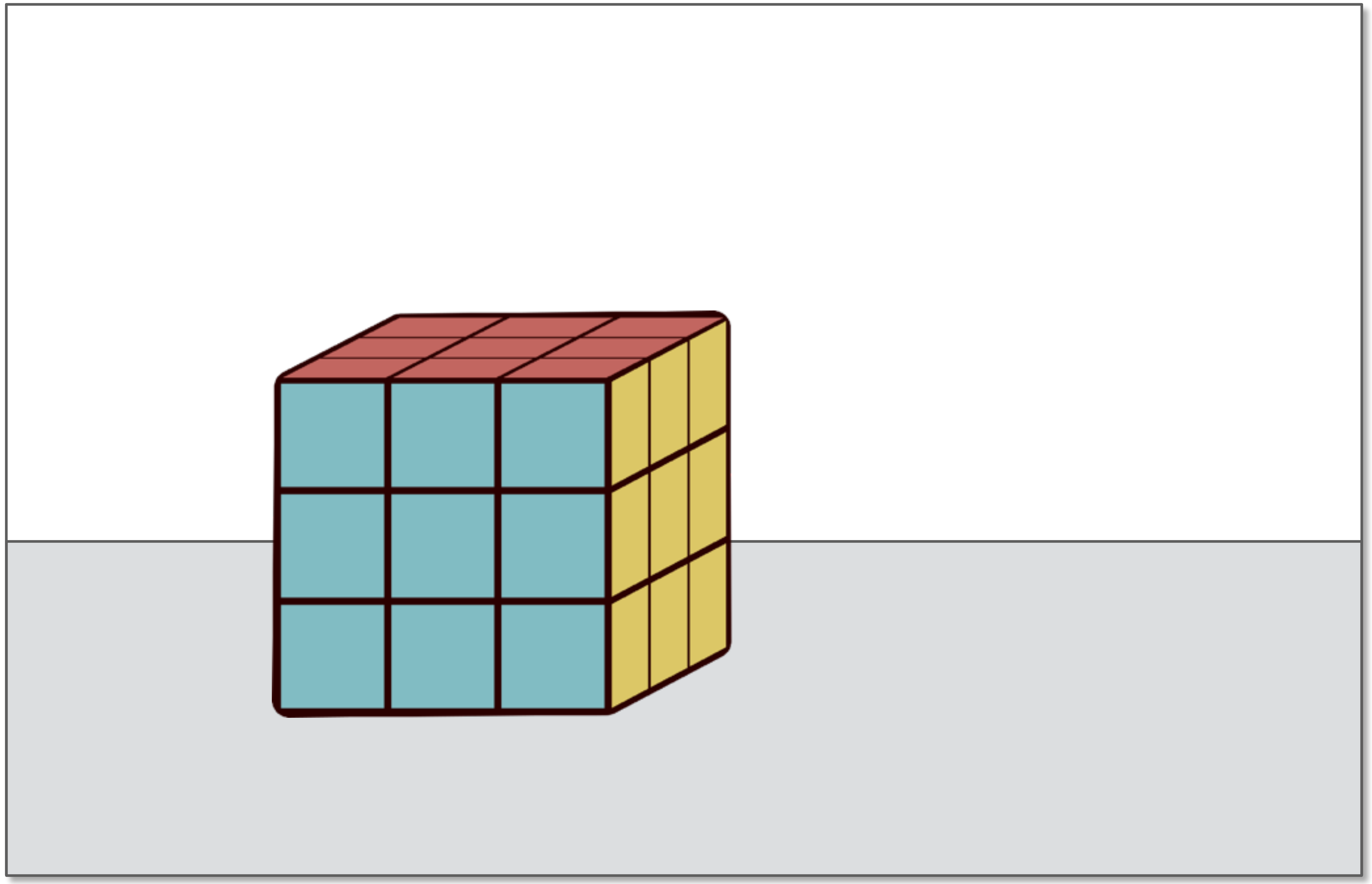


MOVING DISCONTINUITIES

Pixel integrals $\iint$  $\mathrm{d}x\,\mathrm{d}y$

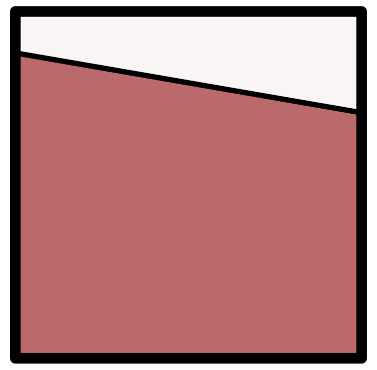
Light integrals $\int$  $\mathrm{d}\omega$

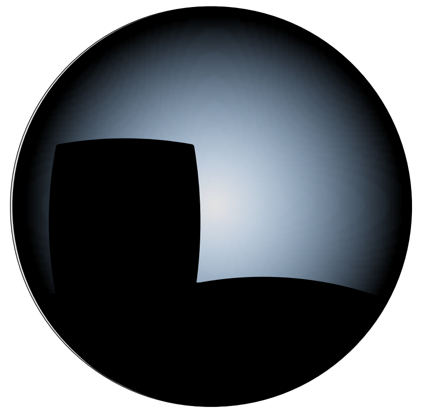
BSDF integrals $\int$  $\mathrm{d}\omega$

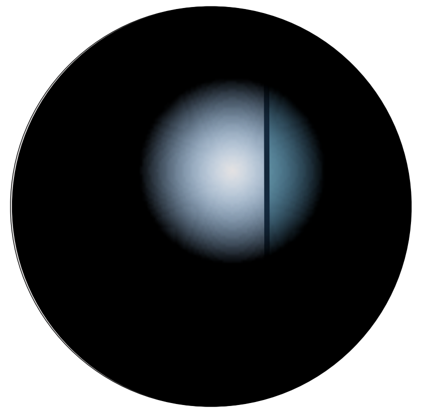


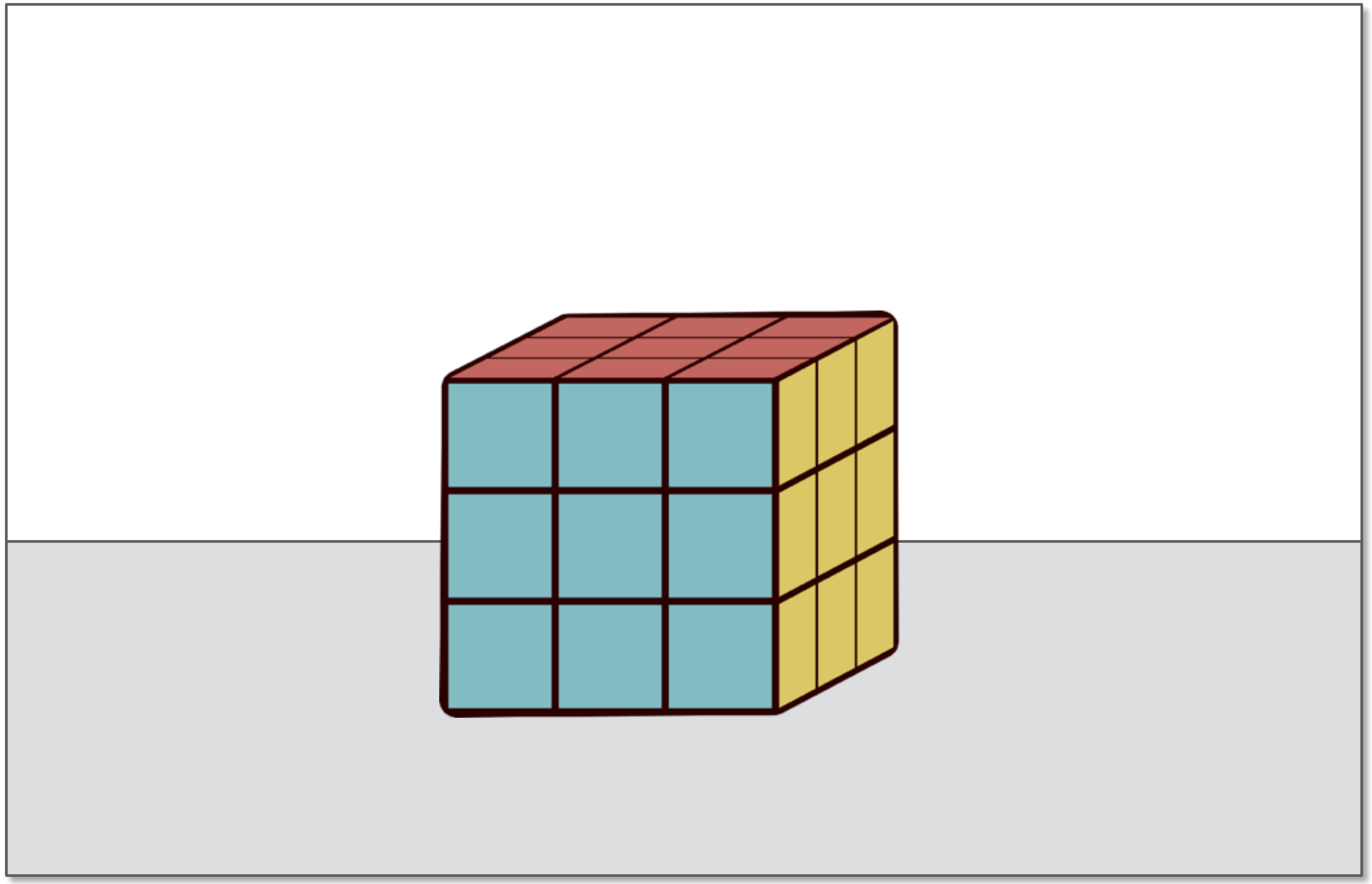
Scene parameter x_i

MOVING DISCONTINUITIES

Pixel integrals $\iint$  $\mathrm{d}x\,\mathrm{d}y$

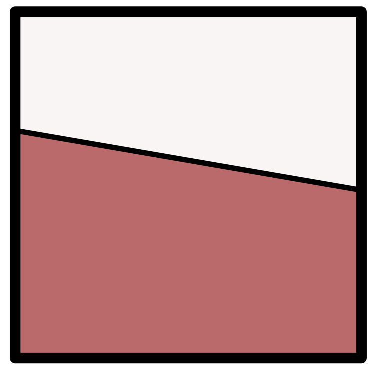
Light integrals $\int$  $\mathrm{d}\omega$

BSDF integrals $\int$  $\mathrm{d}\omega$

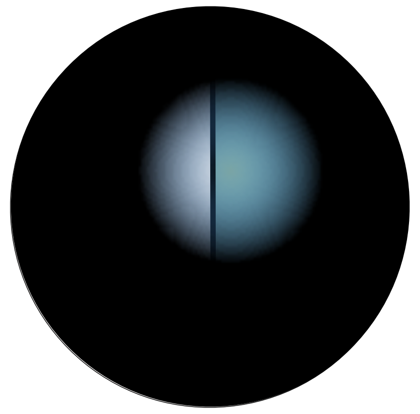


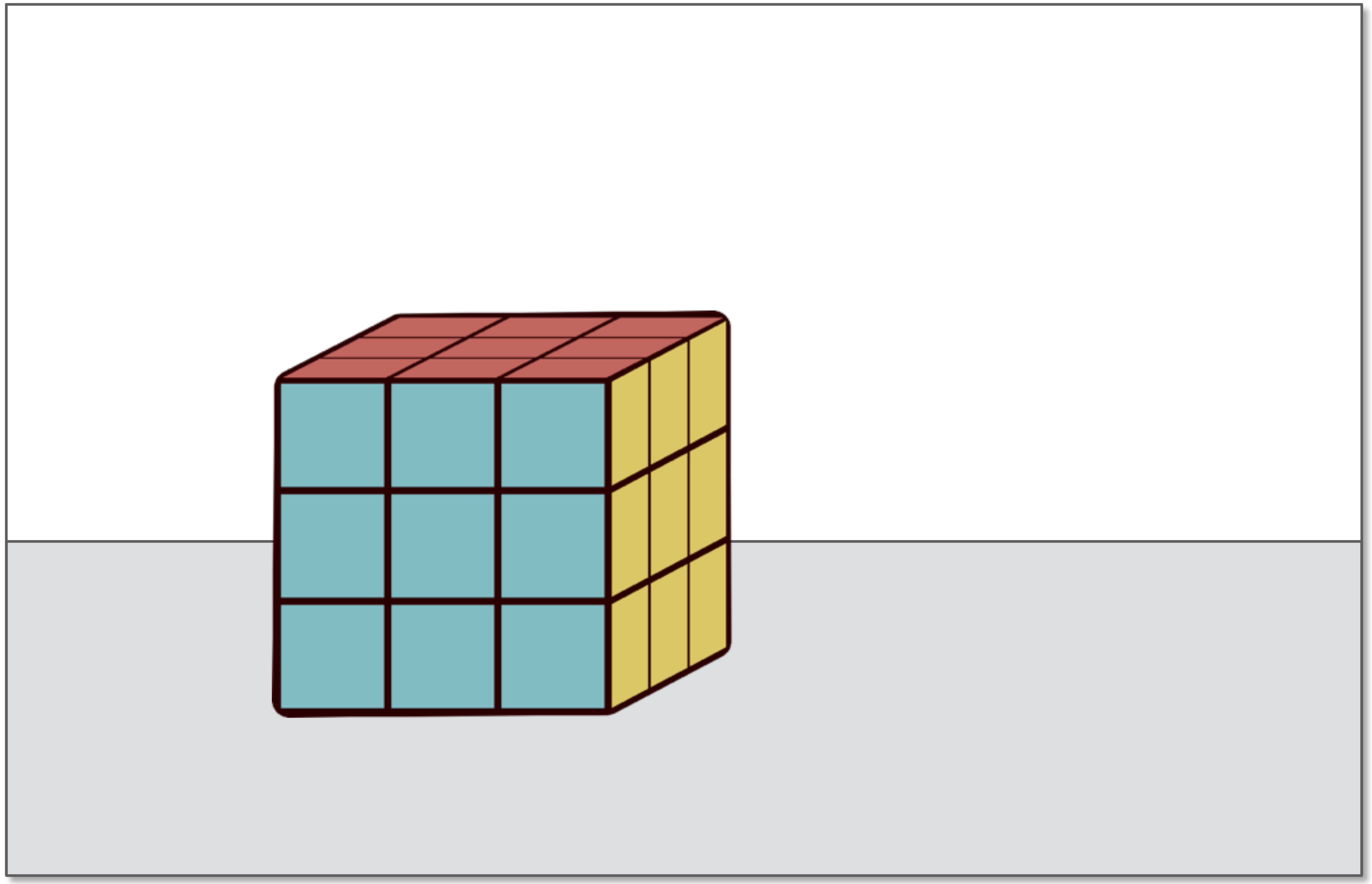
Scene parameter x_i

MOVING DISCONTINUITIES

Pixel integrals $\iint$  $\mathrm{d}x\,\mathrm{d}y$

Light integrals $\int$  $\mathrm{d}\omega$

BSDF integrals $\int$  $\mathrm{d}\omega$

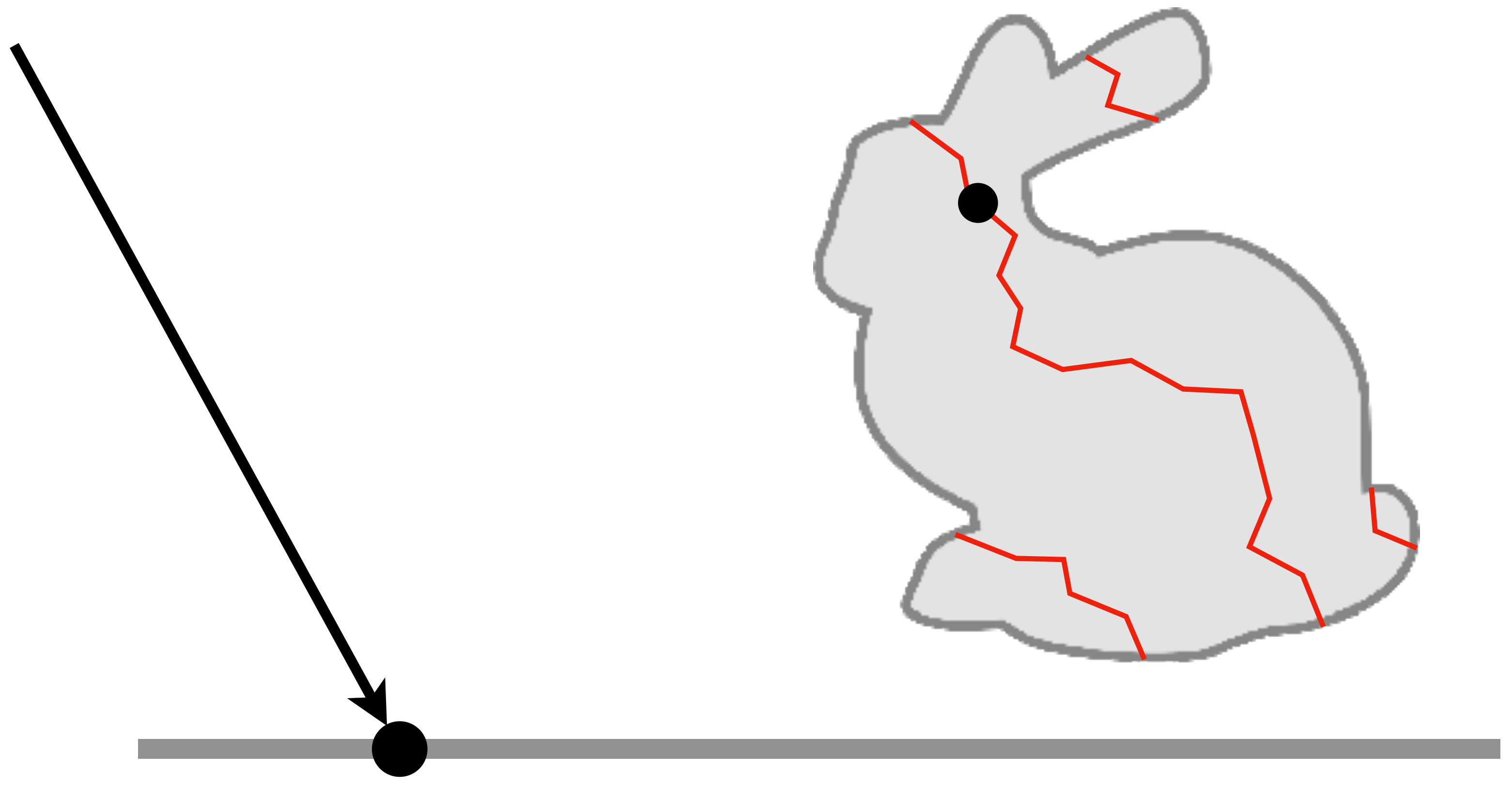


Scene parameter x_i

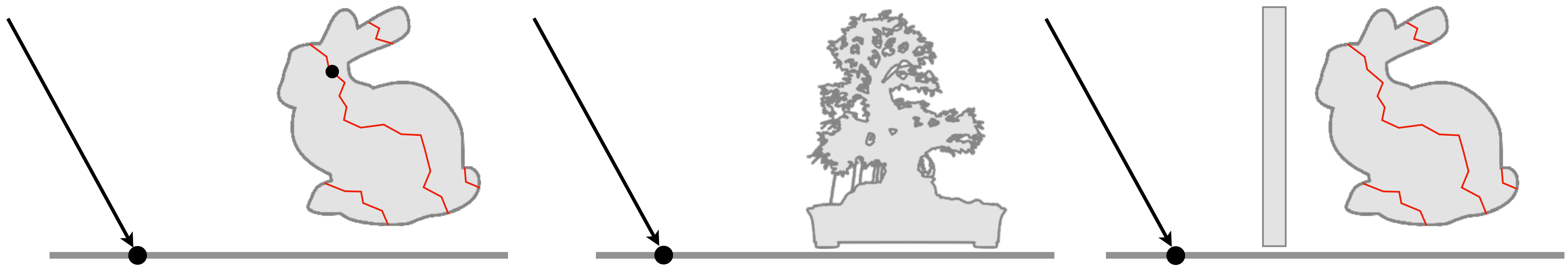
Cannot differentiate standard Monte Carlo estimates

EDGE SAMPLING

EDGE SAMPLING

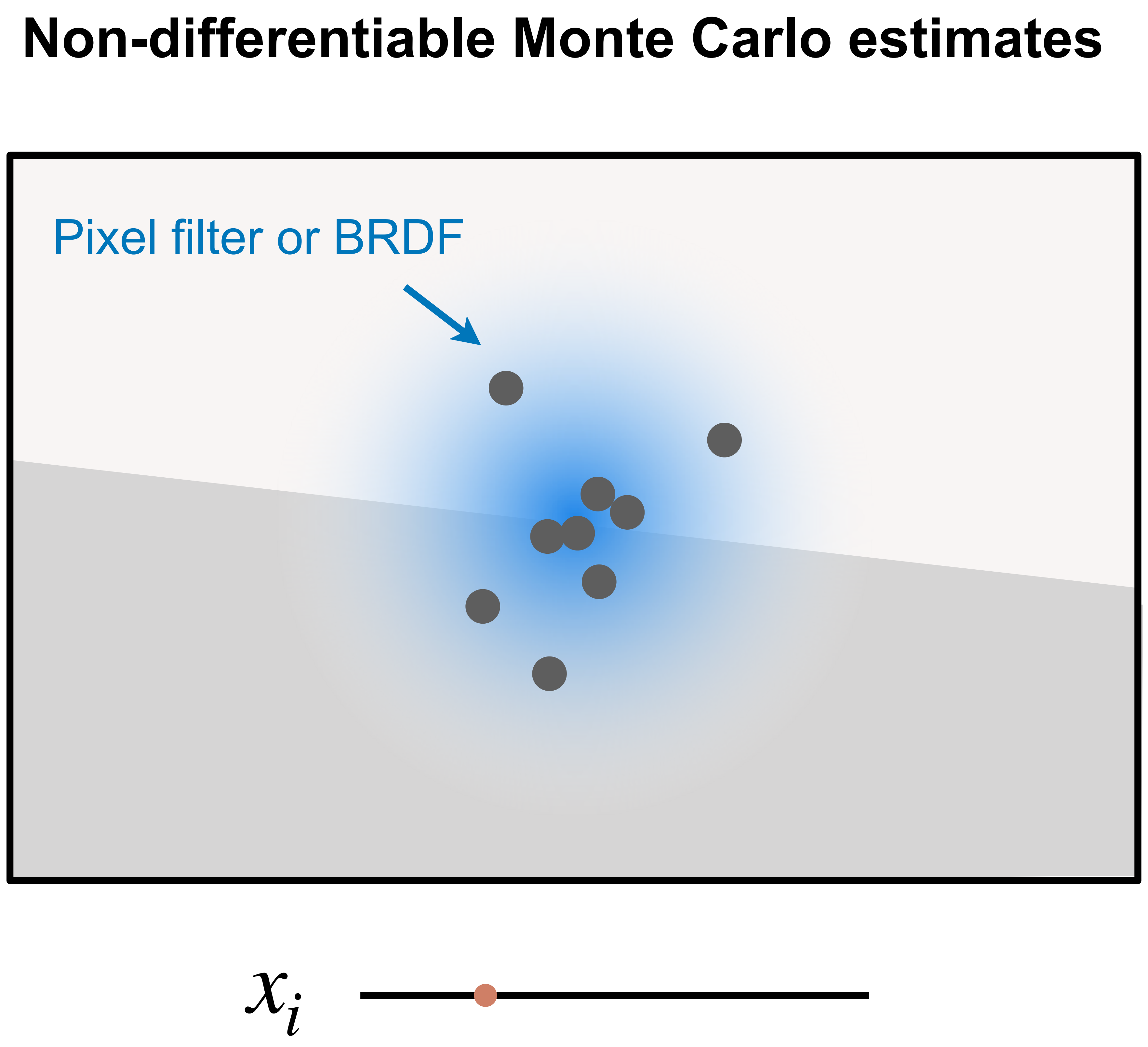


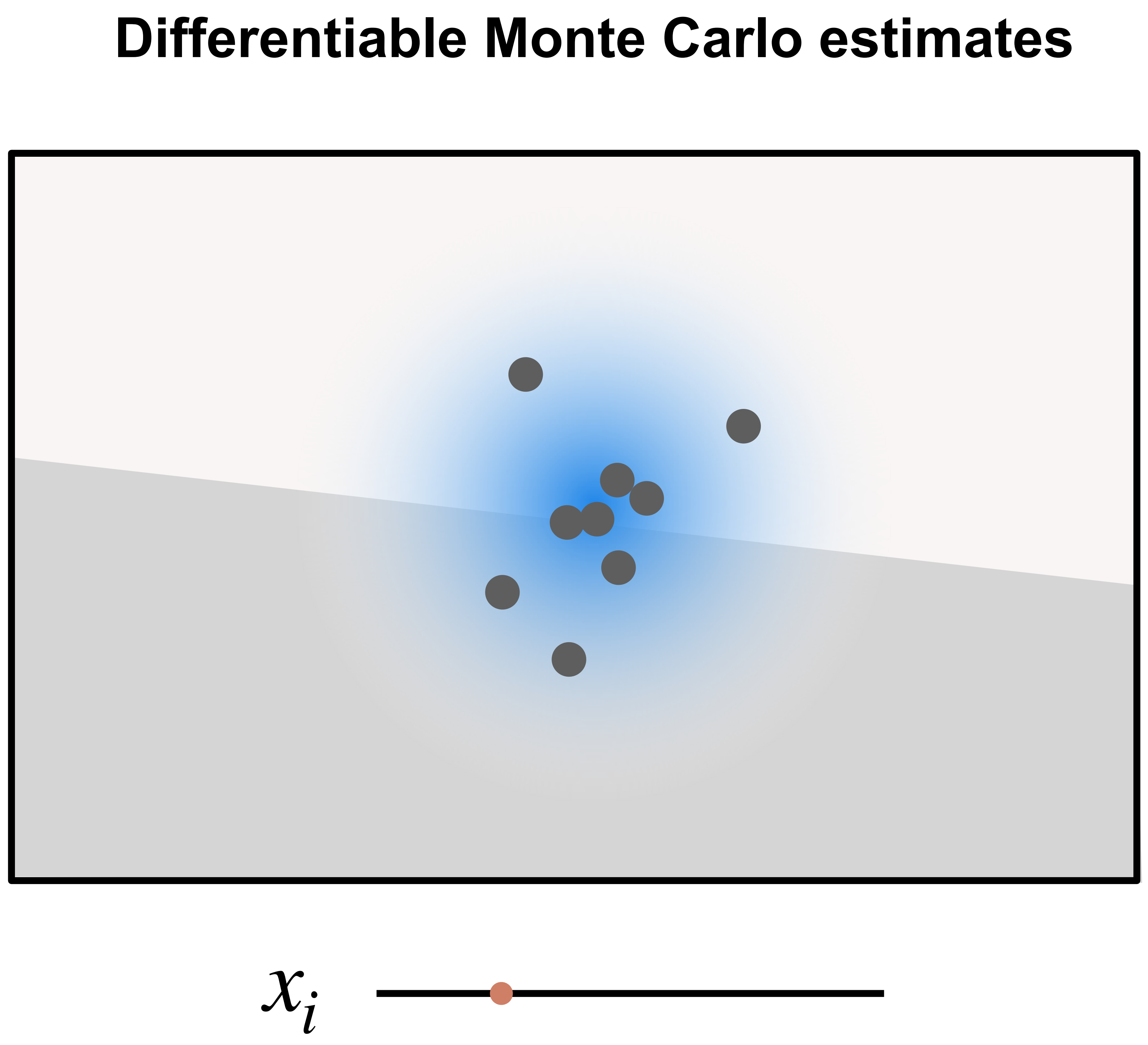
EDGE SAMPLING



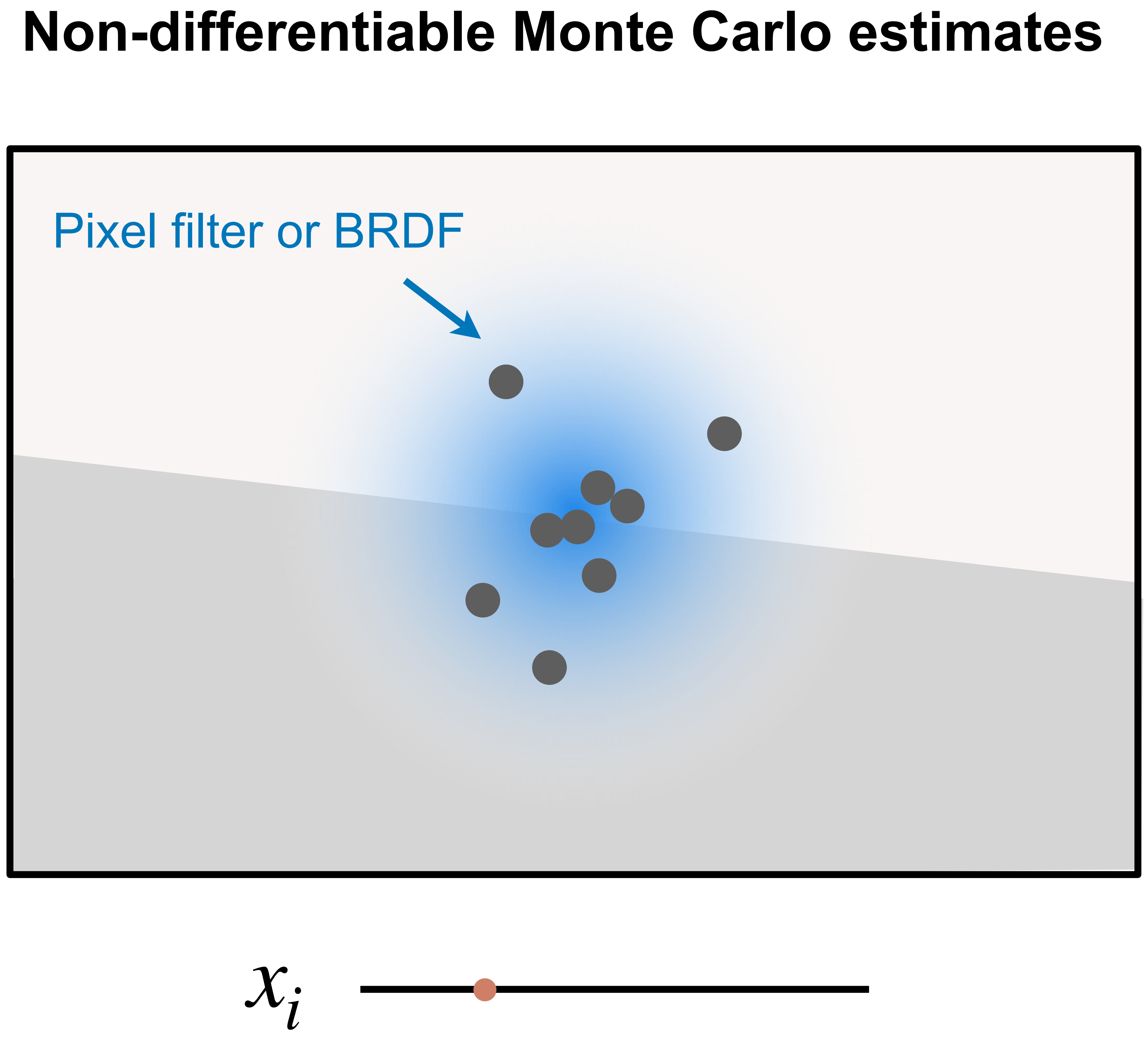
We currently don't have good acceleration data structures for this operation.

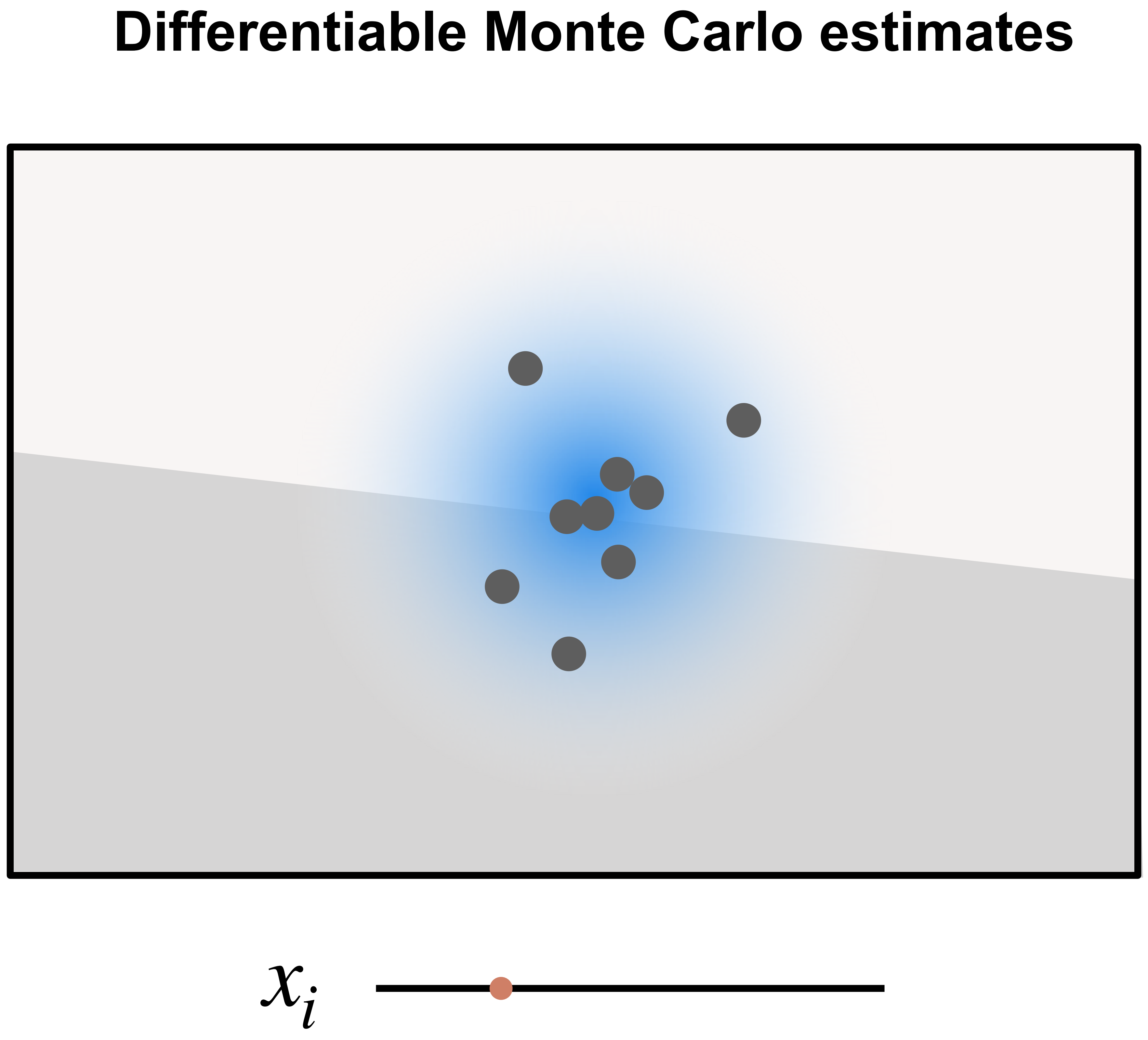
REPARAMETERIZE INTEGRALS?





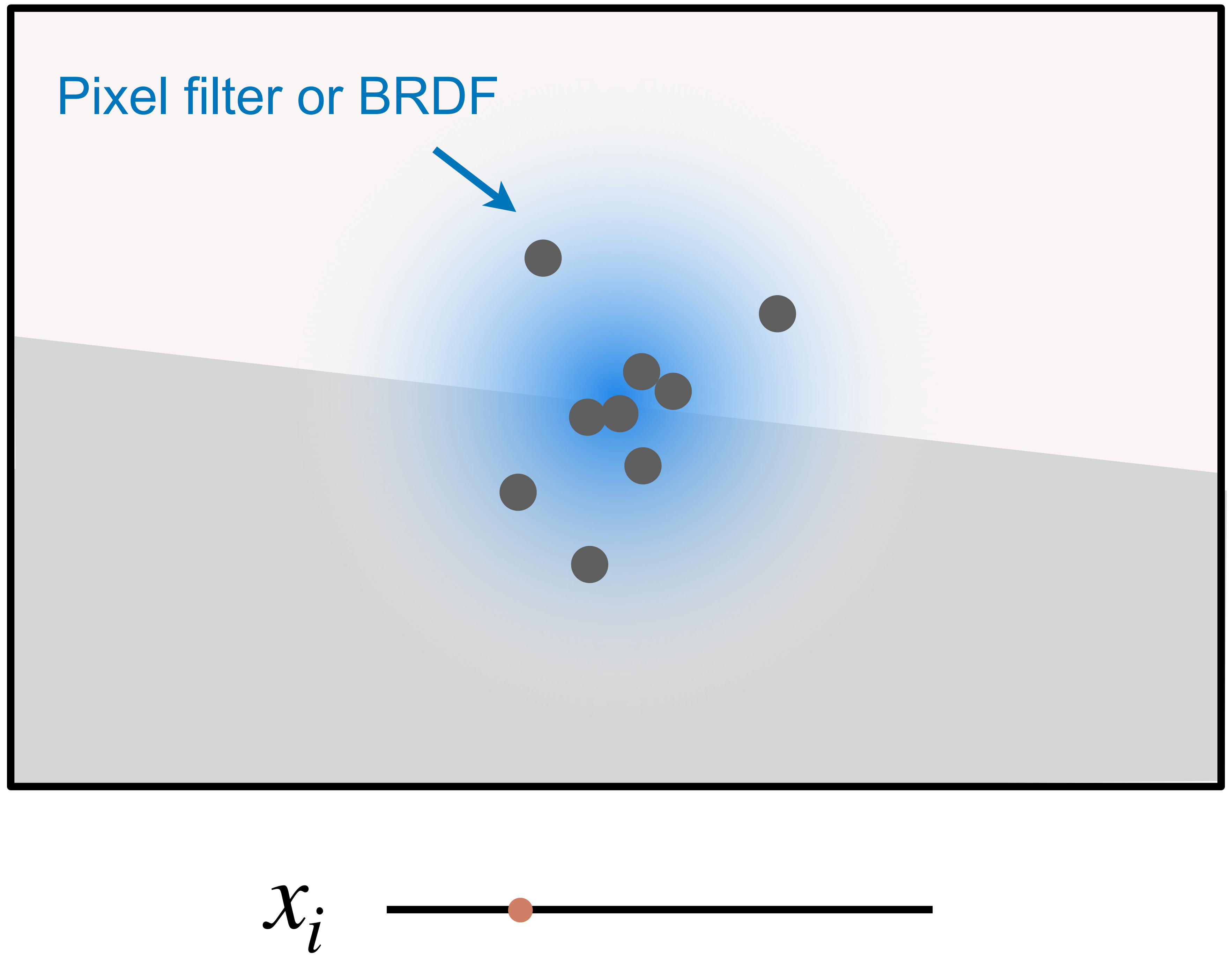
REPARAMETERIZE INTEGRALS?



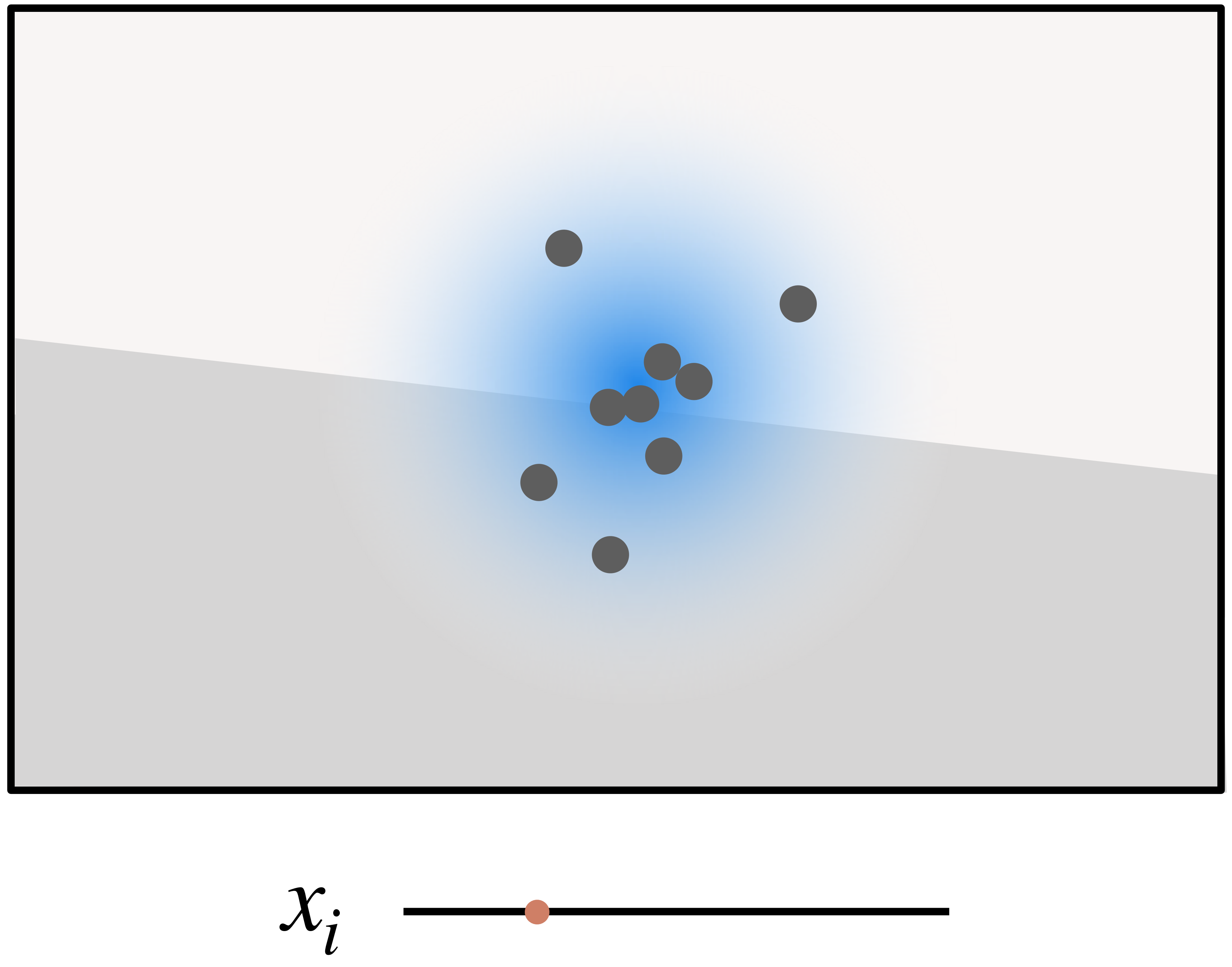


REPARAMETERIZE INTEGRALS?

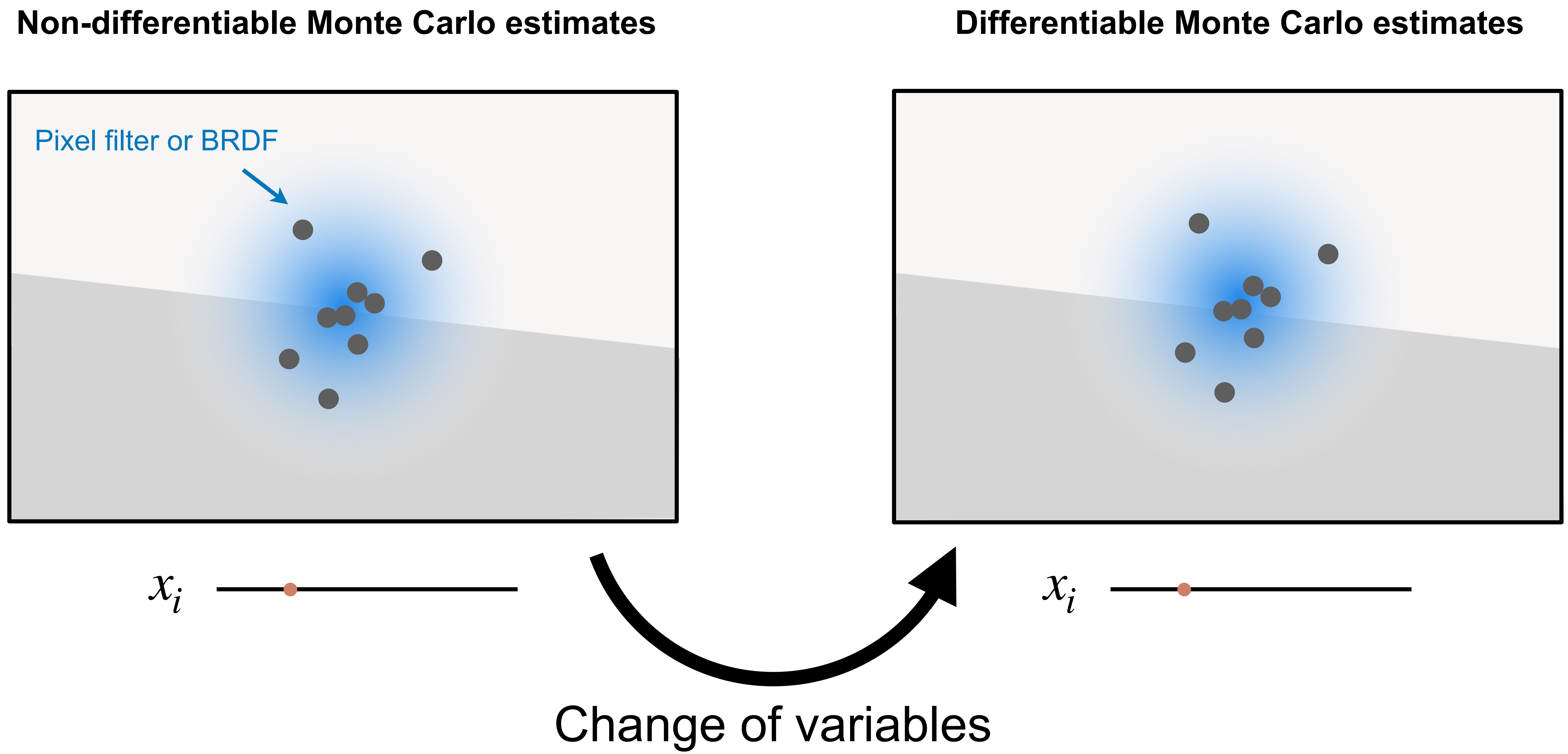
Non-differentiable Monte Carlo estimates



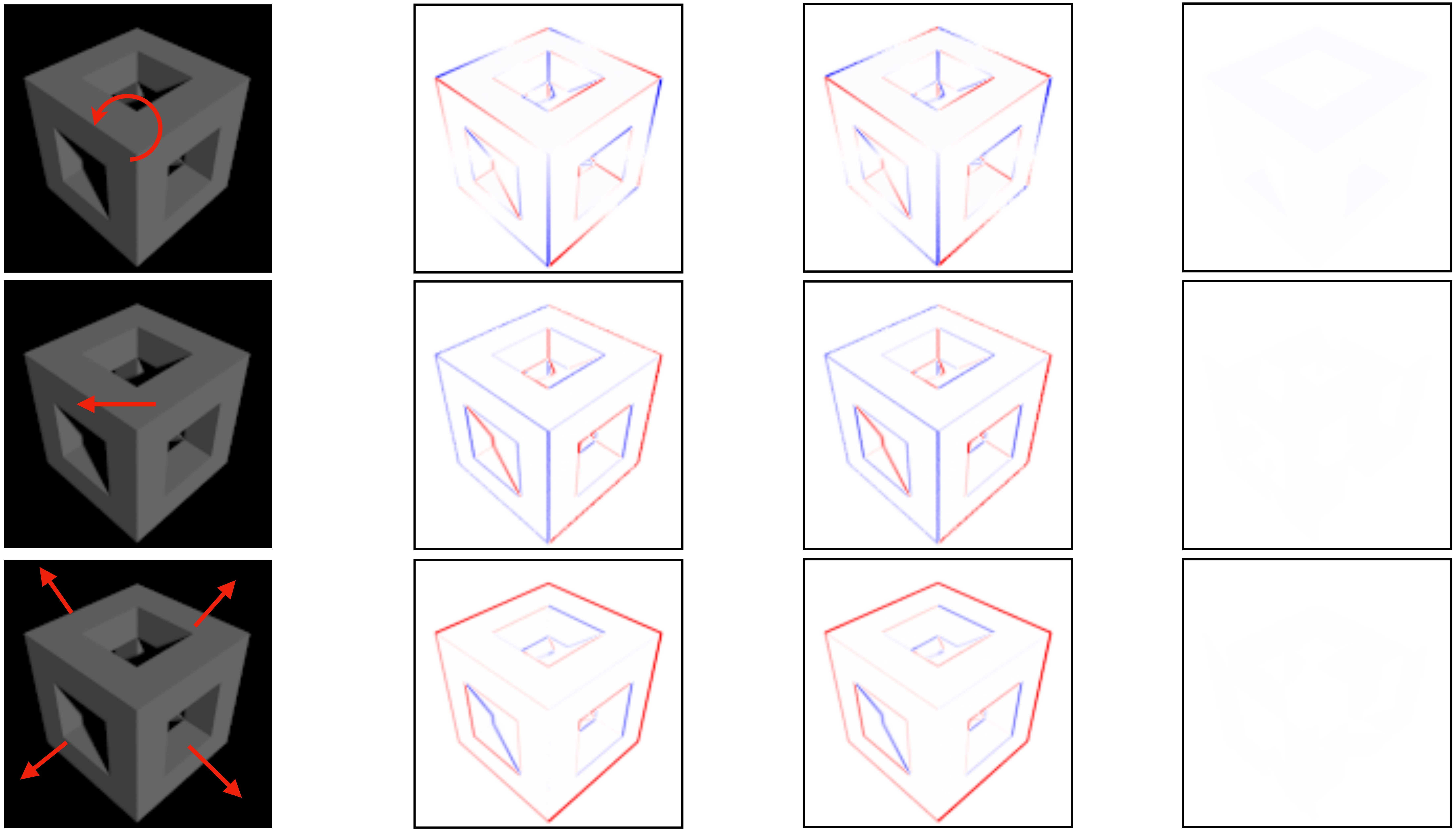
Differentiable Monte Carlo estimates



REPARAMETERIZE INTEGRALS?



RESULTS

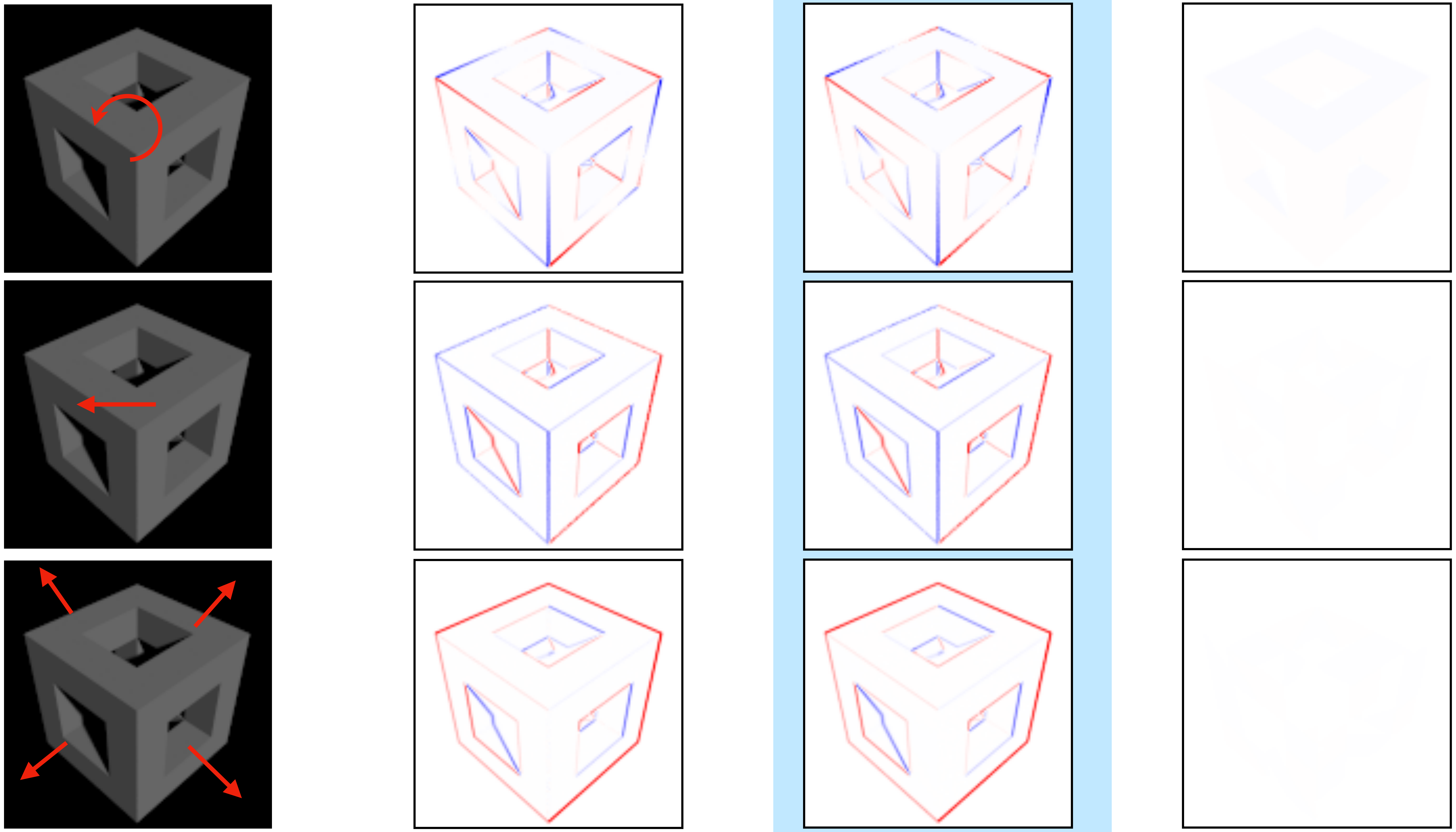


Ours

Reference
(Finite differences)

Without
changes of variables

RESULTS

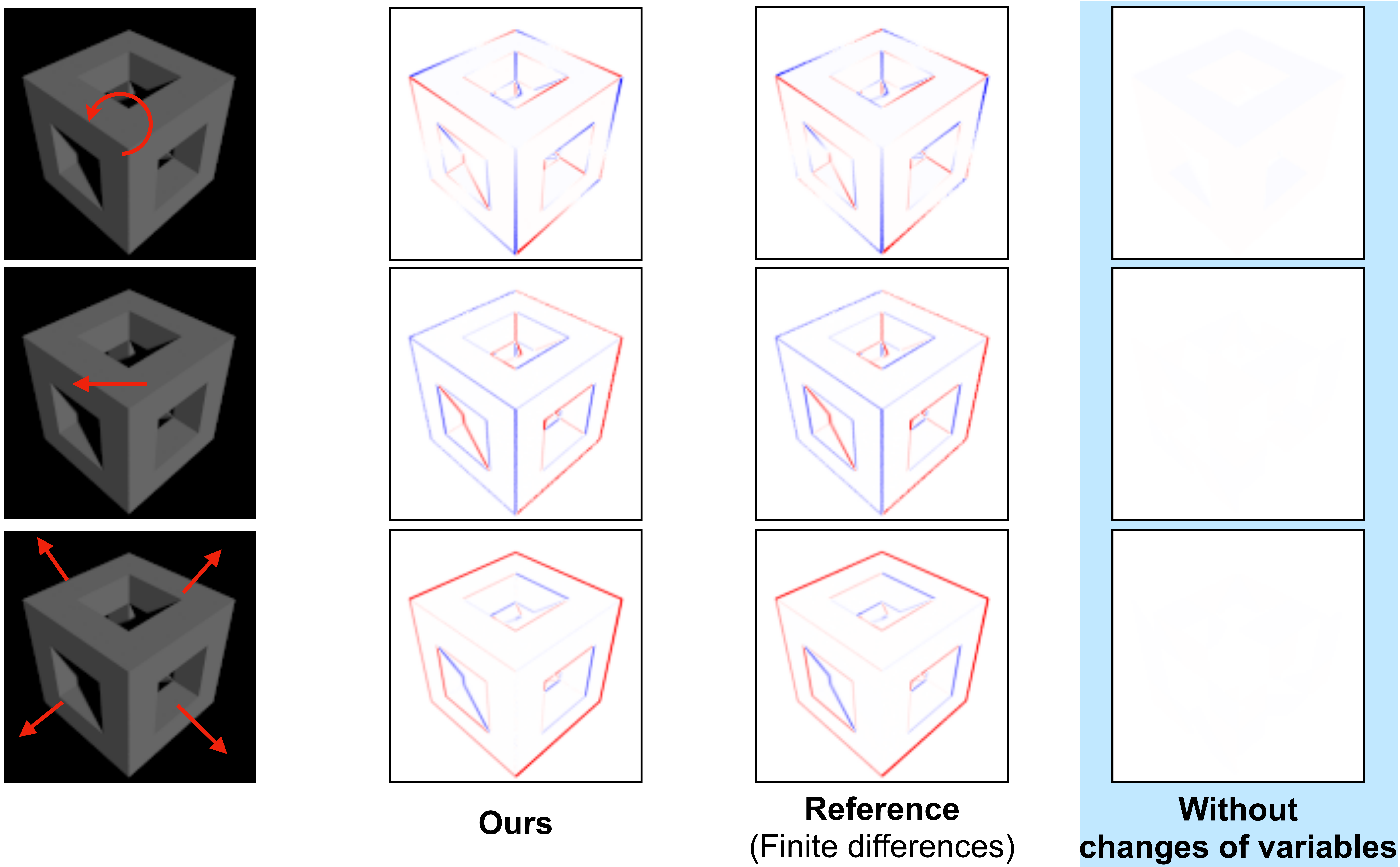


Ours

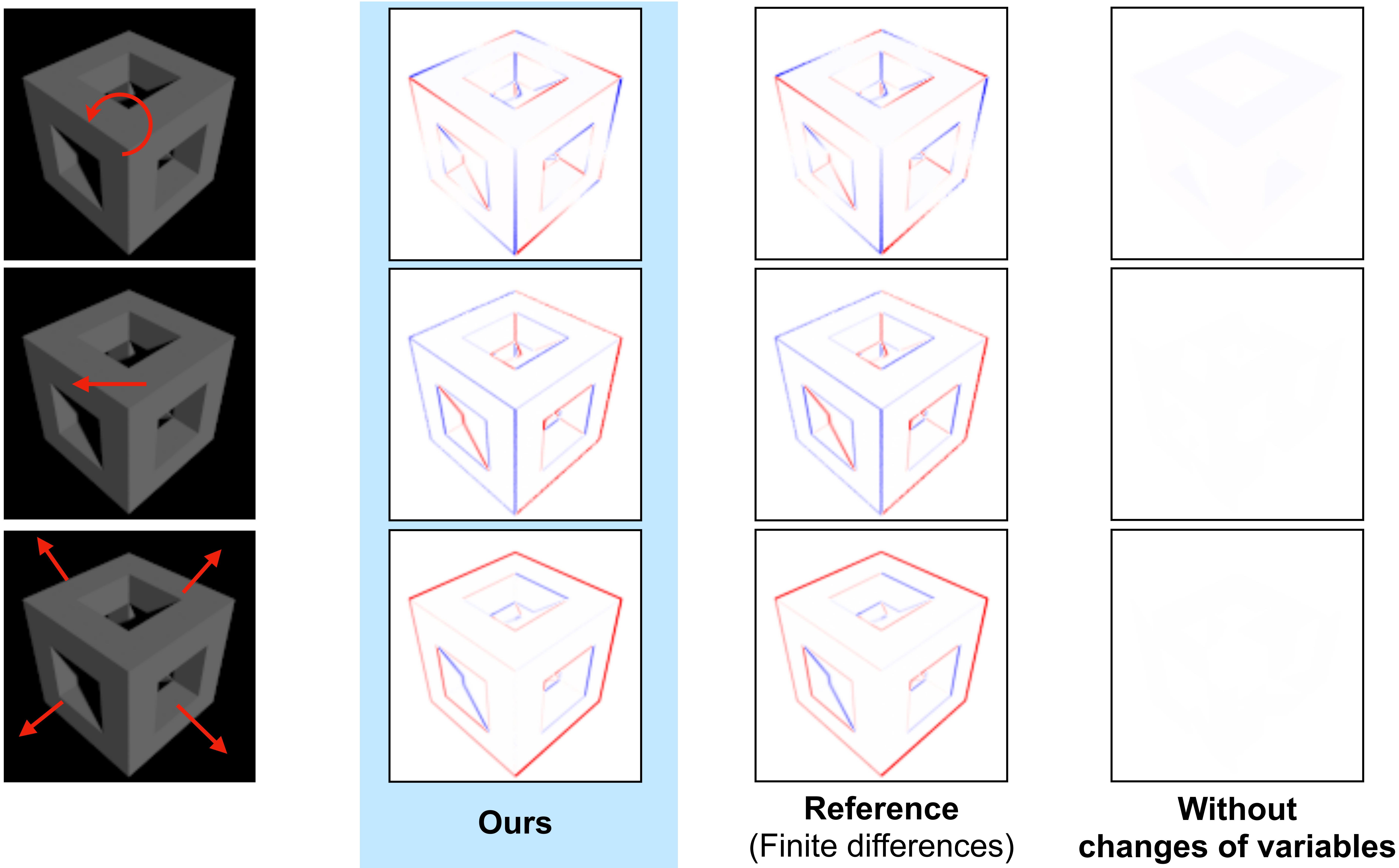
Reference
(Finite differences)

Without changes of variables

RESULTS

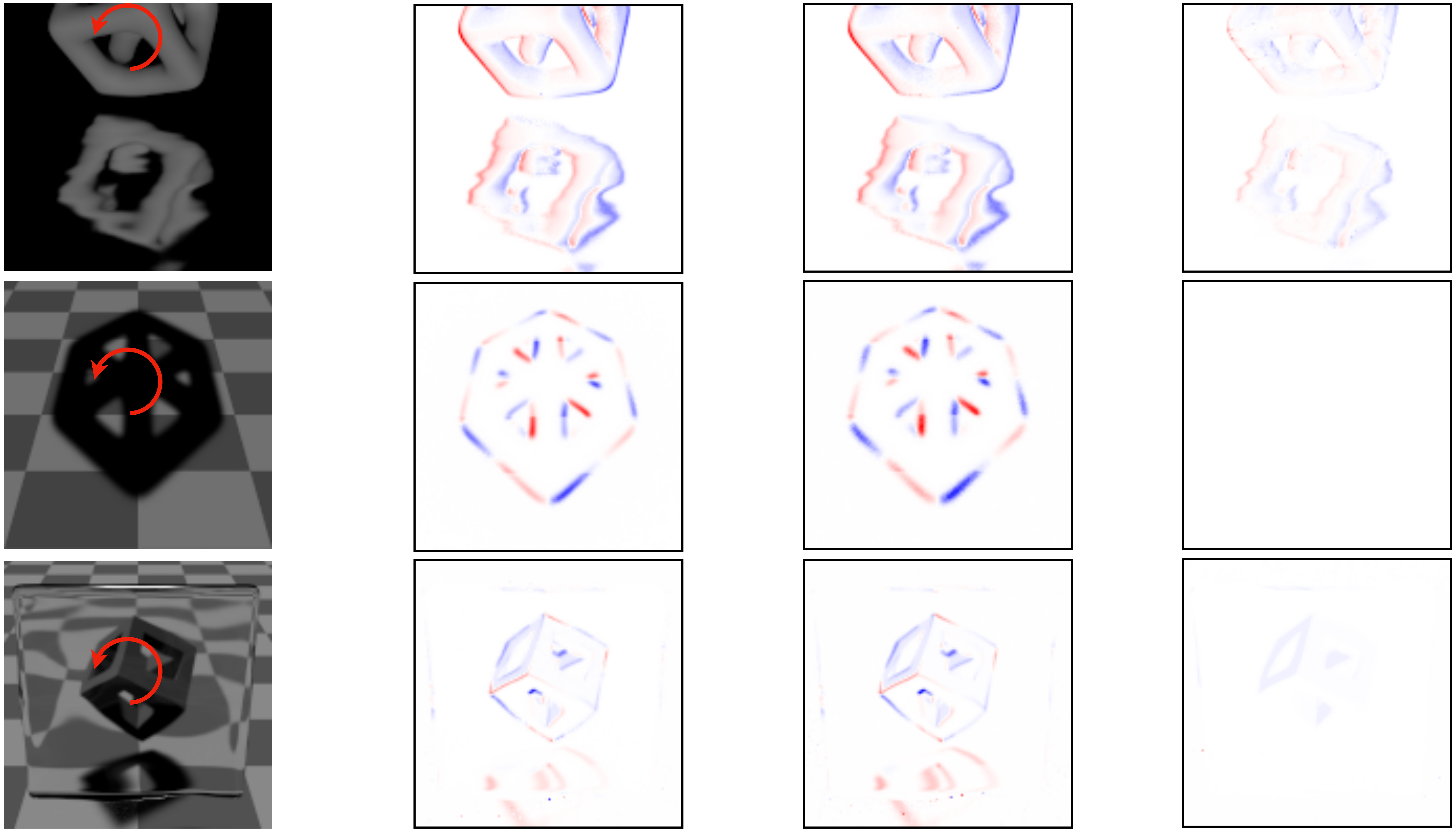

Ours
Reference
(Finite differences)
Without
changes of variables

RESULTS



Ours
Reference
(Finite differences)
Without
changes of variables

RESULTS



Glossy reflection

Shadows

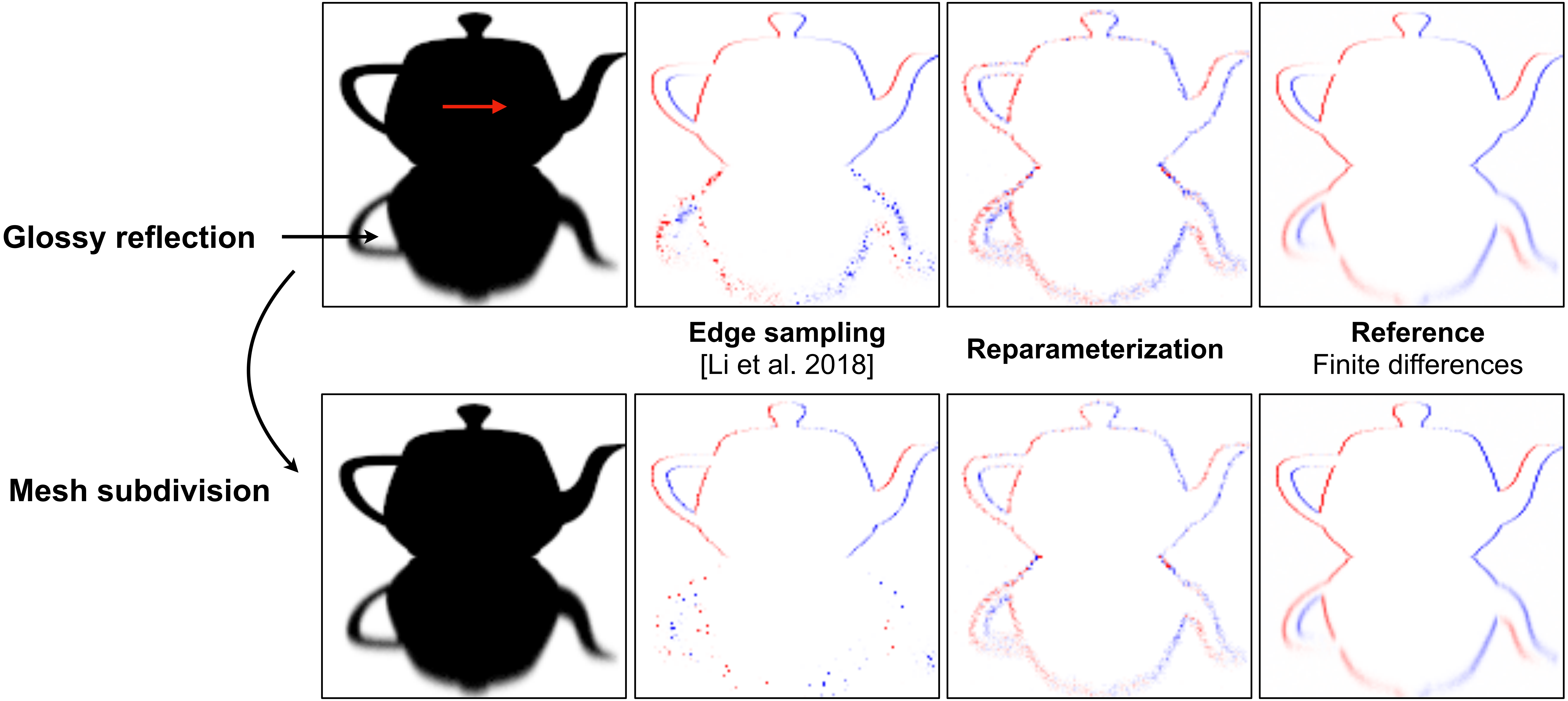
Refraction

Ours

Reference
(Finite differences)

Without changes of variables

RESULTS


Glossy reflection
Mesh subdivision
Edge sampling
[Li et al. 2018]
Reparameterization
Reference
Finite differences

FAST INTEGRATION

Dealing with discontinuities is not enough.

Want to propagate derivative information through complex simulations with **millions** of differentiable parameters.

DIFFERENTIAL MONTE CARLO

“Monte-Carlo calculation of derivatives of functionals from the solution of the transfer equation according to the parameters of the system”

G. A. Mikhailov, Novosibirsk, **July 1966**

“Monte Carlo Analysis of Reactivity Coefficients in Fast Reactors, General Theory and Applications”

L.B. Miller, Argonne Natl. Laboratory, **March 1967**

ВЫЧИСЛЕНИЕ МЕТОДОМ МОНТЕ-КАРЛО ПРОИЗВОДНЫХ ФУНКЦИОНАЛОВ ОТ РЕШЕНИЯ УРАВНЕНИЯ ПЕРЕНОСА ПО ПАРАМЕТРАМ СИСТЕМ

Г. А. МИХАЙЛОВ

(Новосибирск)

§ 1. Оценка функционалов от решения уравнения переноса методом Монте-Карло. Метод зависимых испытаний

Интегральное уравнение переноса (см., например, [1]) можно записать в виде

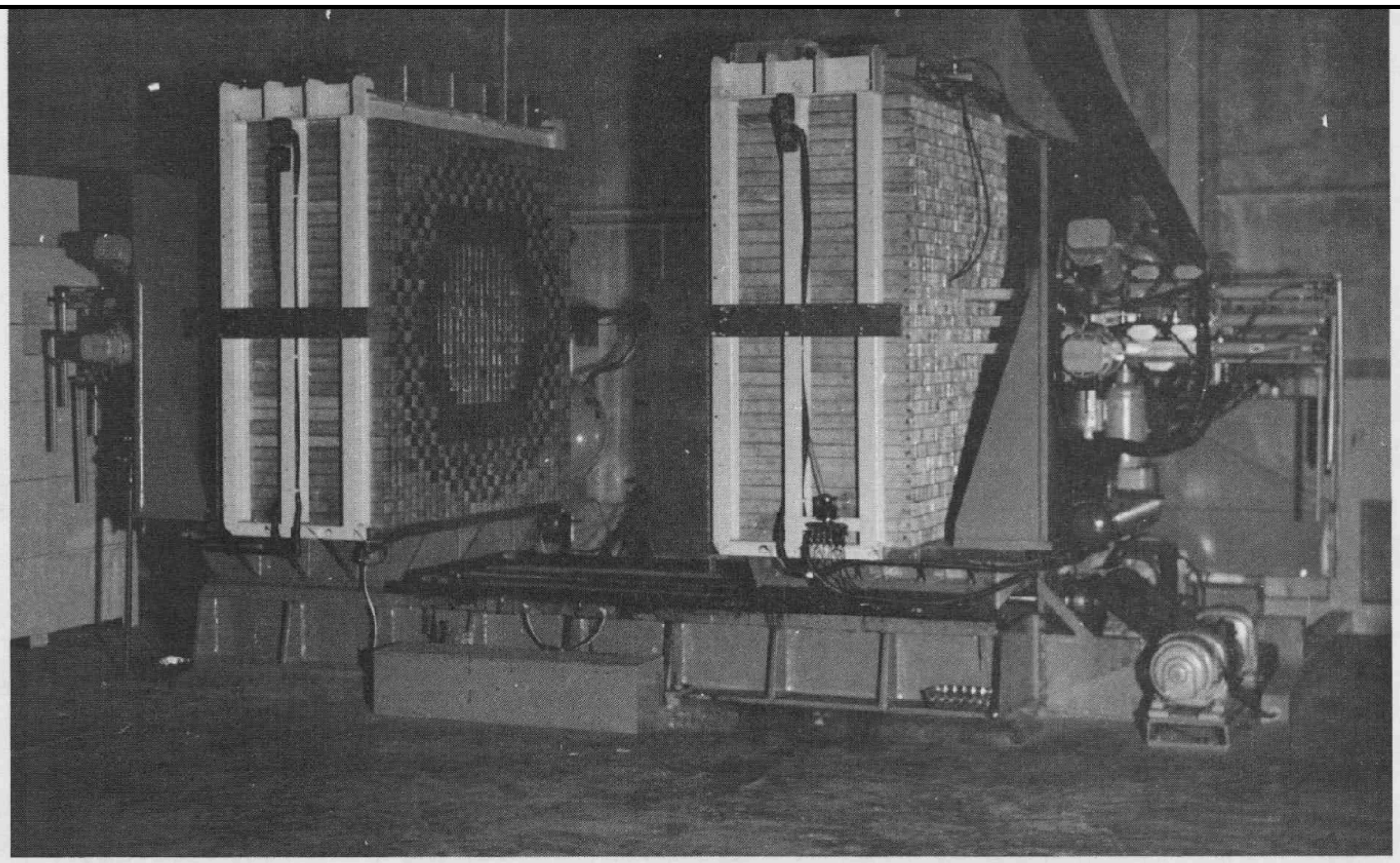
$$F(x) = \int_X k(x' \to x) F(x')\, dx' + f(x), \qquad (1)$$

или

$$F = KF + f,$$

где X — фазовое пространство координат и скоростей, $F(x)$ — плотность столкновений в точке $x \in X$; $k(x' \to x)$ — плотность «первичных» столкновений в точке x от «одного» столкновения в точке x'; $x, x' \in X$, $f(x)$ — плотность источников.

Мы будем предполагать, что решение уравнения (1) можно представить в виде ряда Неймана

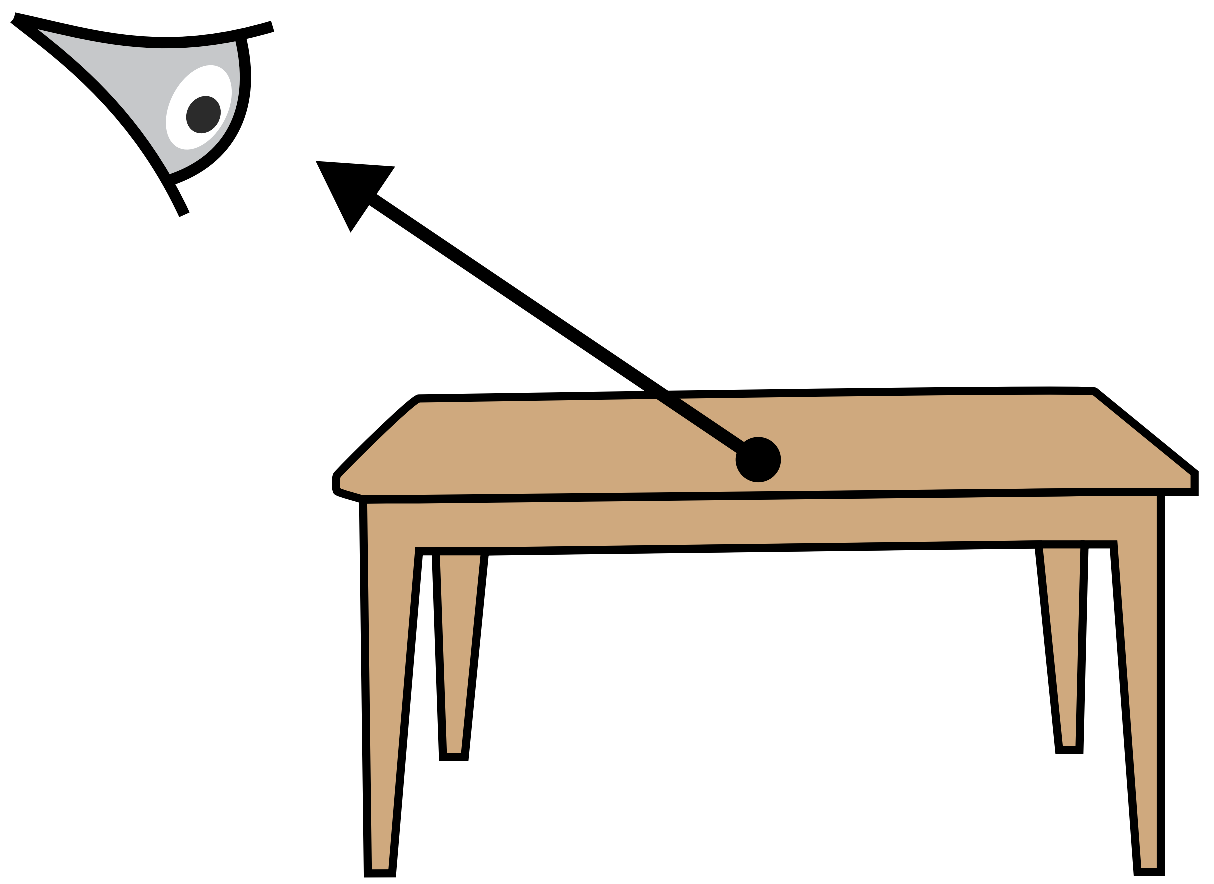


103-298

Fig. 2. ZPR-3 Critical Facility

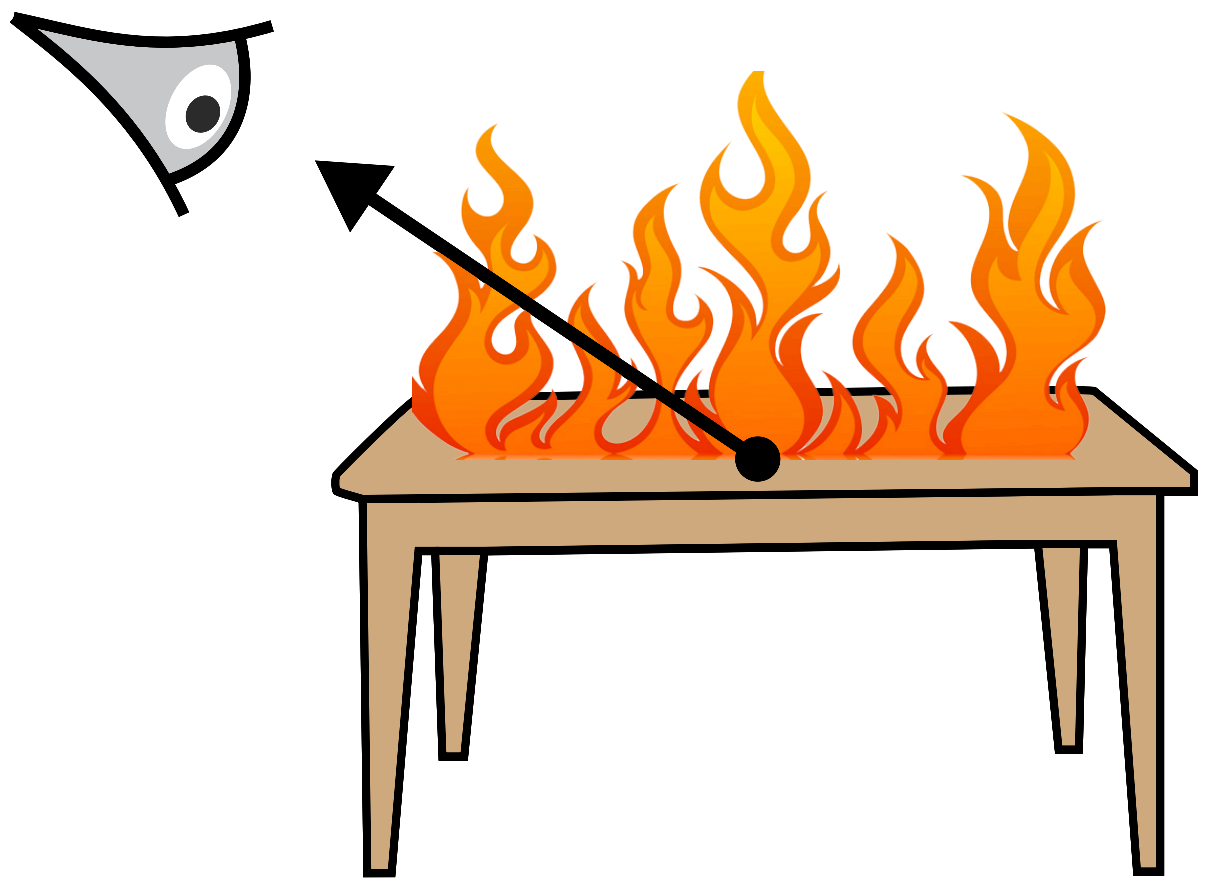
DIFFERENTIATING THE RENDERING EQN

$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_i(\mathbf{x},\omega')\, f_s(\mathbf{x},\omega,\omega')\, \cos\theta \, \mathrm{d}\omega'$$



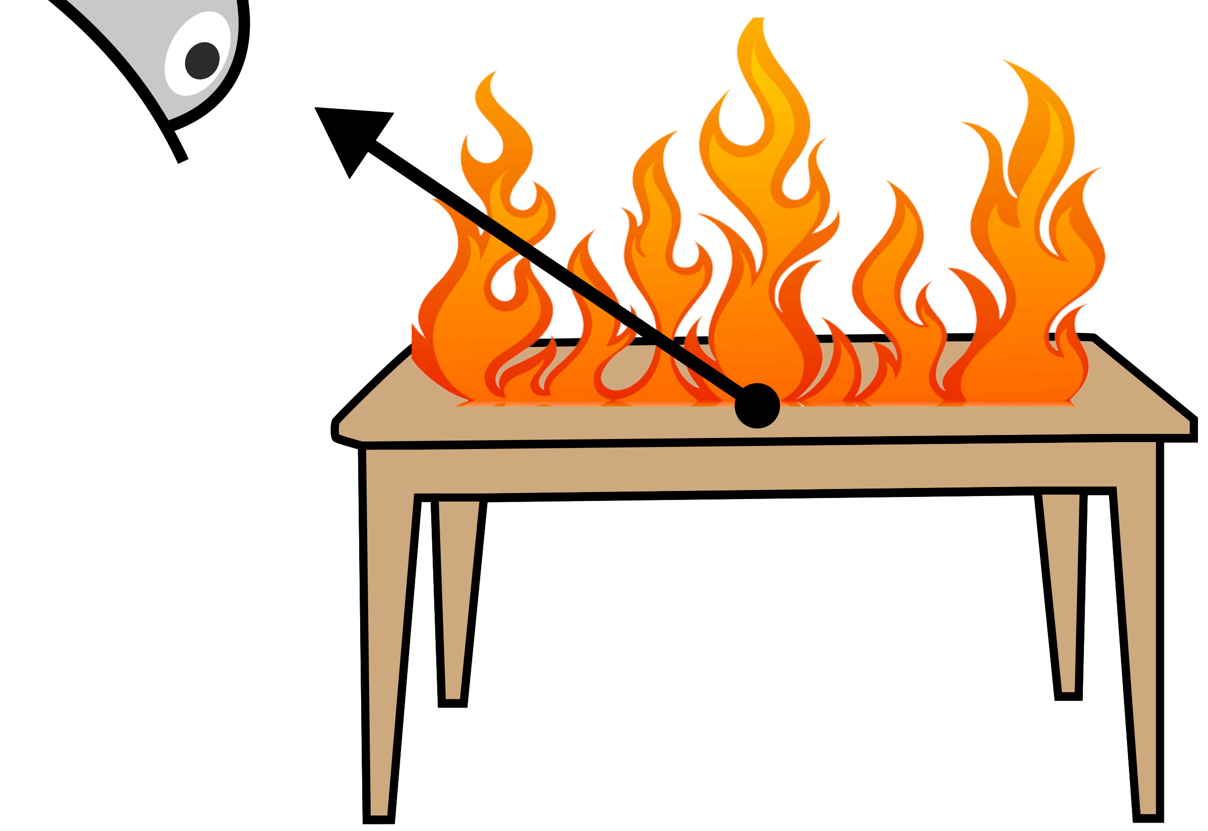
DIFFERENTIATING THE RENDERING EQN

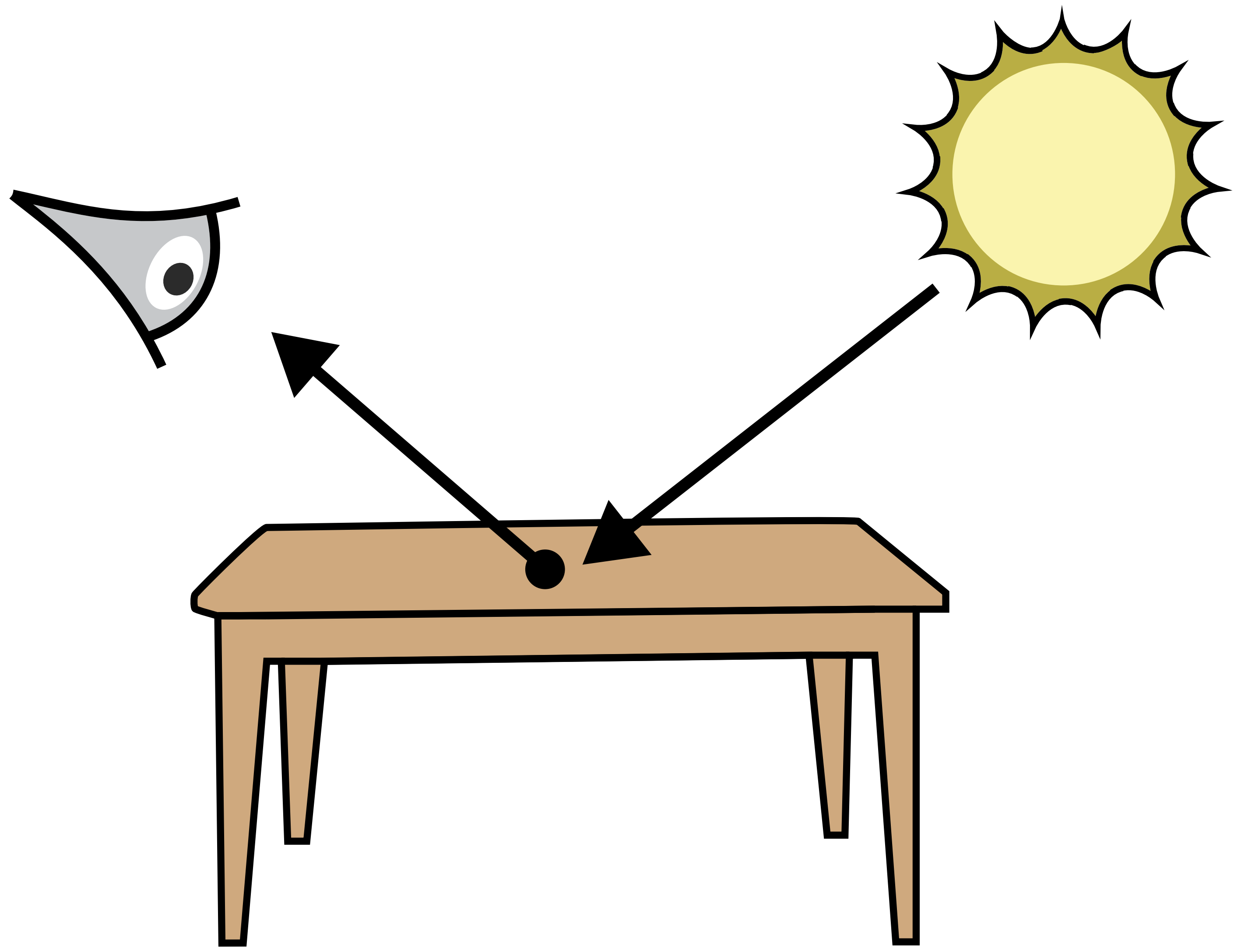
$$L_o(\mathbf{x},\omega) = L_e(\mathbf{x},\omega) + \int_{S^2} L_i(\mathbf{x},\omega')\, f_s(\mathbf{x},\omega,\omega')\, \cos\theta \,\mathrm{d}\omega'$$



DIFFERENTIATING THE RENDERING EQN

$$L_o(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{S^2} L_i(\mathbf{x}, \omega') \, f_s(\mathbf{x}, \omega, \omega') \, \cos\theta \, \mathrm{d}\omega'$$





DIFFERENTIATING THE RENDERING EQN

$$L_o(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{S^2} L_i(\mathbf{x}, \omega') \, f_s(\mathbf{x}, \omega, \omega') \, \cos\theta \, \mathrm{d}\omega'$$

$\partial_{\mathbf{x}}$ derivative wrt. scene parameters

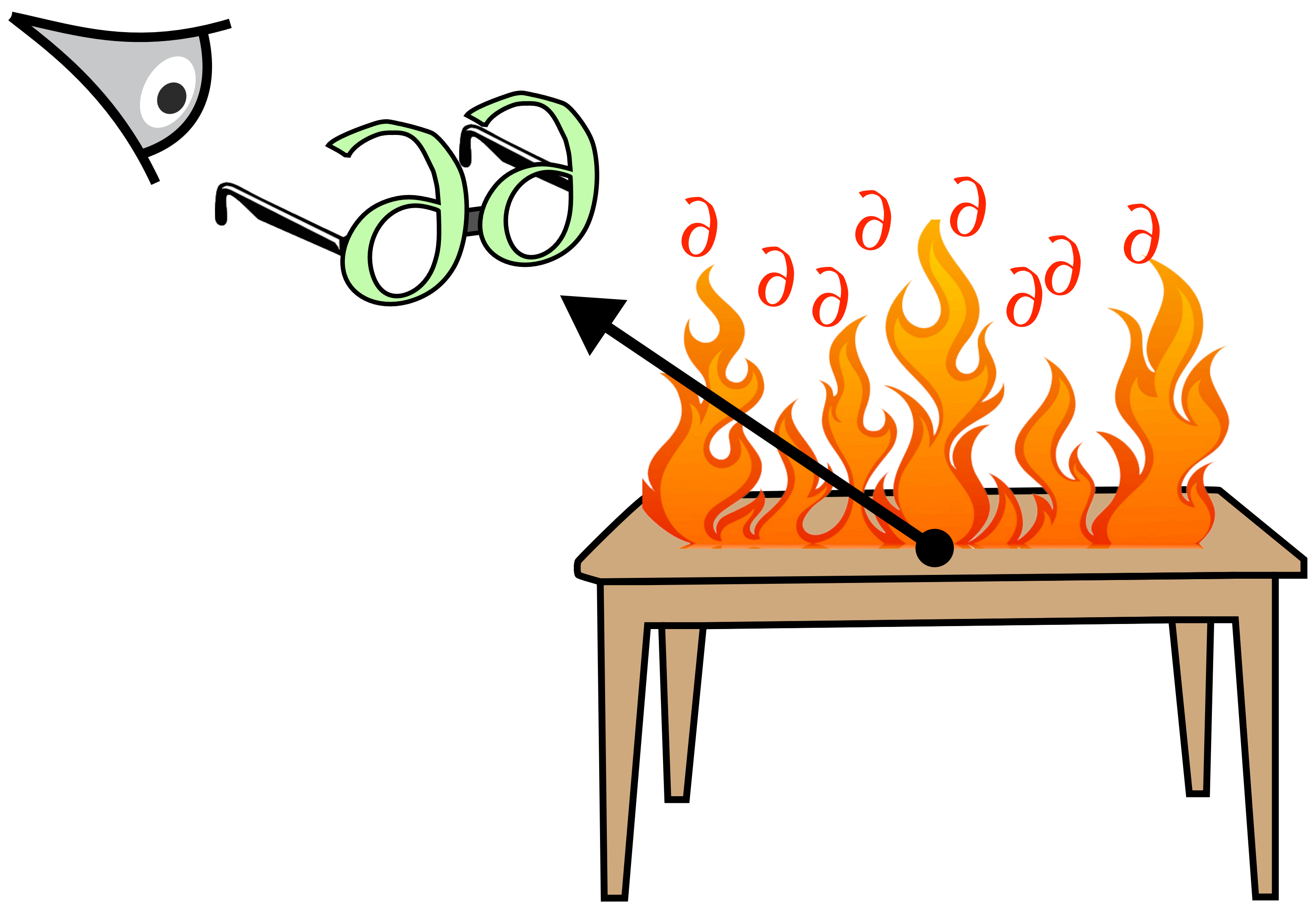
DIFFERENTIATING THE RENDERING EQN

$$\partial_{\mathbf{x}} L_o(\mathbf{x},\omega) = \partial_{\mathbf{x}} L_e(\mathbf{x},\omega) + \int_{S^2} L_i(\mathbf{x},\omega')\, f_s(\mathbf{x},\omega,\omega')\, \cos\theta\, \mathrm{d}\omega'$$

$\partial_{\mathbf{x}}$ derivative wrt. scene parameters

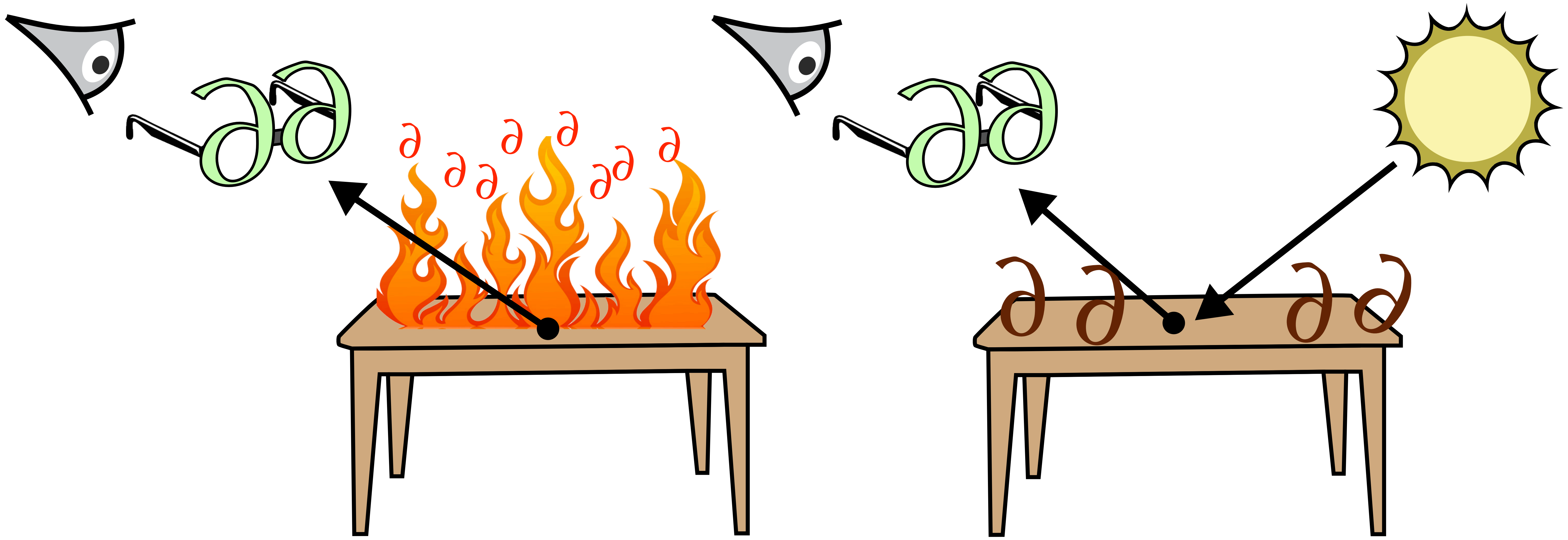
DIFFERENTIATING THE RENDERING EQN

$$\partial_{\mathbf{x}} L_o(\mathbf{x}, \omega) = \partial_{\mathbf{x}} L_e(\mathbf{x}, \omega) + \int_{S^2} L_i(\mathbf{x}, \omega') \, f_s(\mathbf{x}, \omega, \omega') \cos\theta \, \mathrm{d}\omega'$$



DIFFERENTIATING THE RENDERING EQN

$$\partial_{\mathbf{x}} L_o(\mathbf{x}, \omega) = \partial_{\mathbf{x}} L_e(\mathbf{x}, \omega) + \int_{S^2} \left[L_i(\mathbf{x}, \omega') \, \partial_{\mathbf{x}} f_s(\mathbf{x}, \omega, \omega') \right.$$



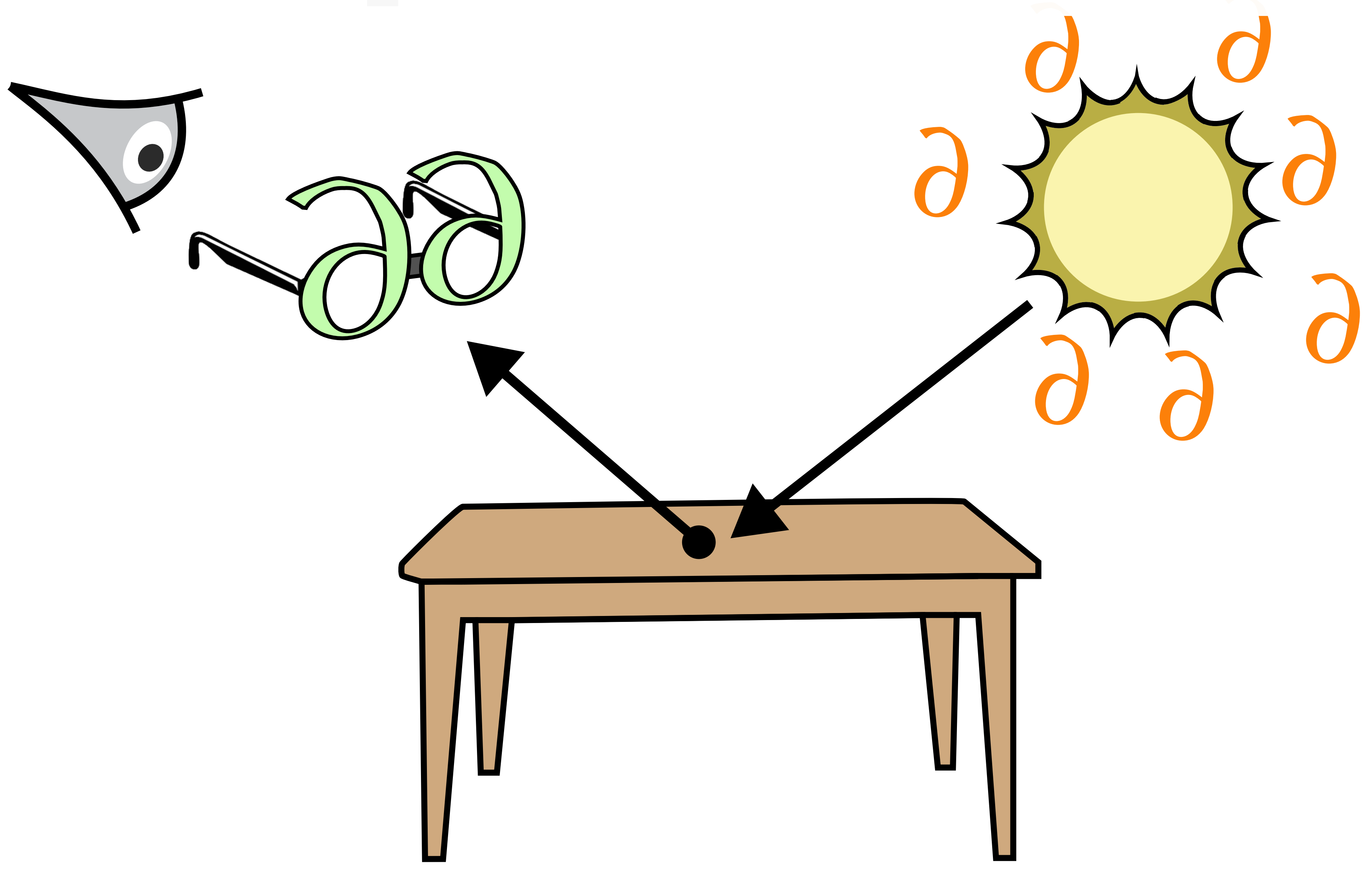
DIFFERENTIATING THE RENDERING EQN

$$\partial_{\mathbf{x}} L_o(\mathbf{x}, \omega) = \partial_{\mathbf{x}} L_e(\mathbf{x}, \omega) + \int_{S^2} \Big[L_i(\mathbf{x}, \omega')\, \partial_{\mathbf{x}} f_s(\mathbf{x}, \omega, \omega') + \partial_{\mathbf{x}} L_i(\mathbf{x}, \omega')\, f_s(\mathbf{x}, \omega, \omega') \Big] \cos\theta \, \mathrm{d}\omega'$$

DIFFERENTIATING THE RENDERING EQN

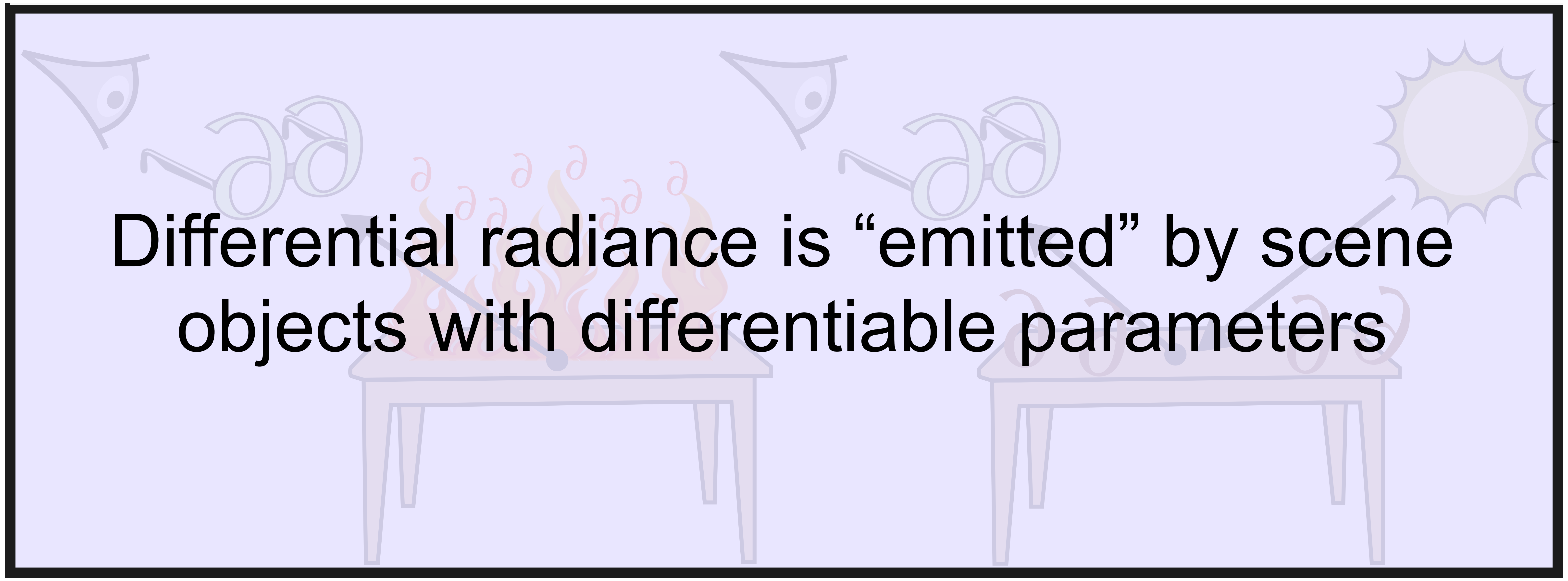
TL;DR

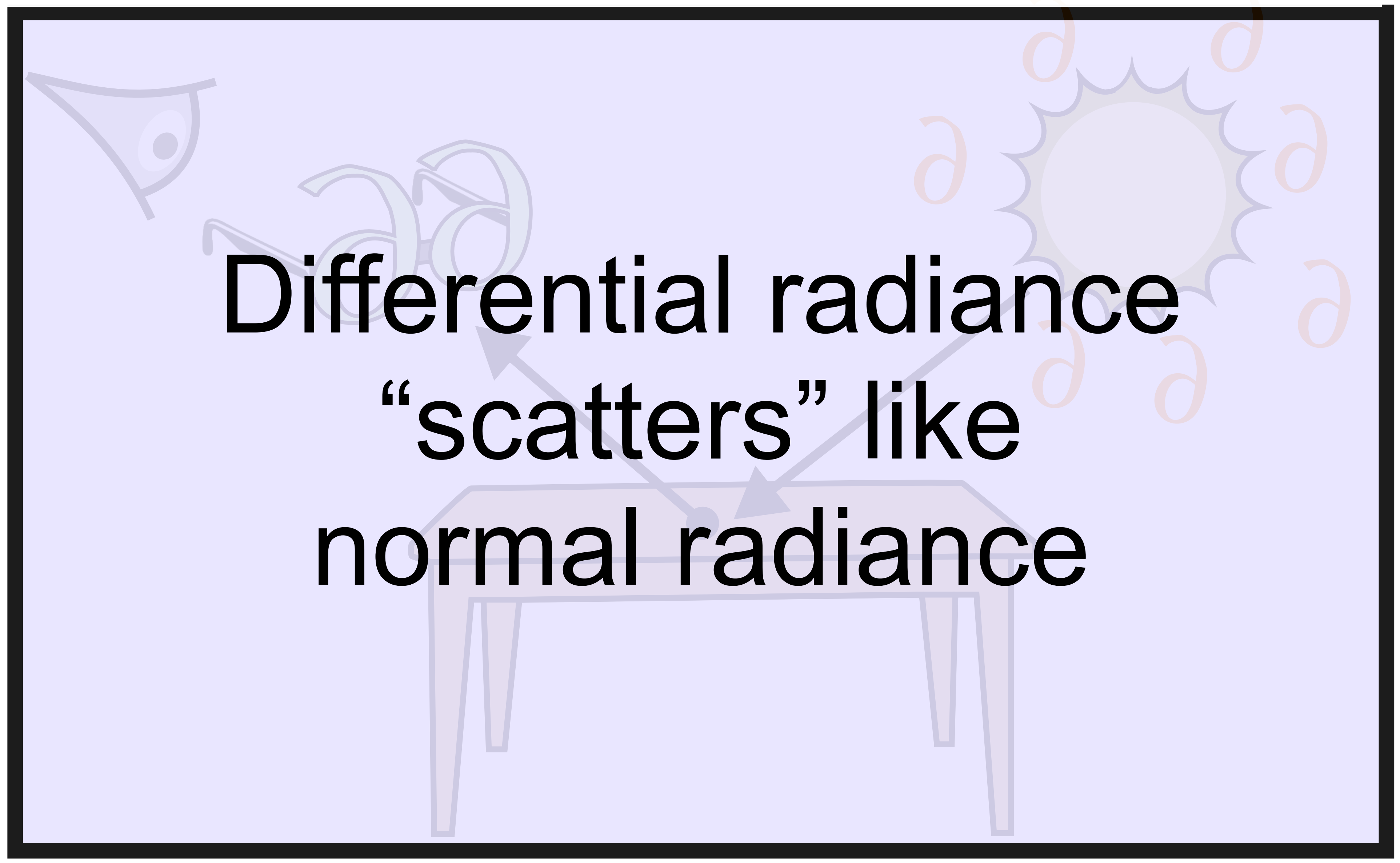
Differential radiance is “emitted” by scene objects with differentiable parameters



DIFFERENTIATING THE RENDERING EQN

TL;DR





$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$


$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$
$\frac{\partial z}{\partial y}$


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)


Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)

Gradients
Derivative wrt. parameters
Derivative wrt. objective
Dot product (discrete)

WHAT'S WRONG WITH THIS?

1MPix rendering &
1M parameters:

WHAT'S WRONG WITH THIS?

1MPix rendering &
1M parameters:

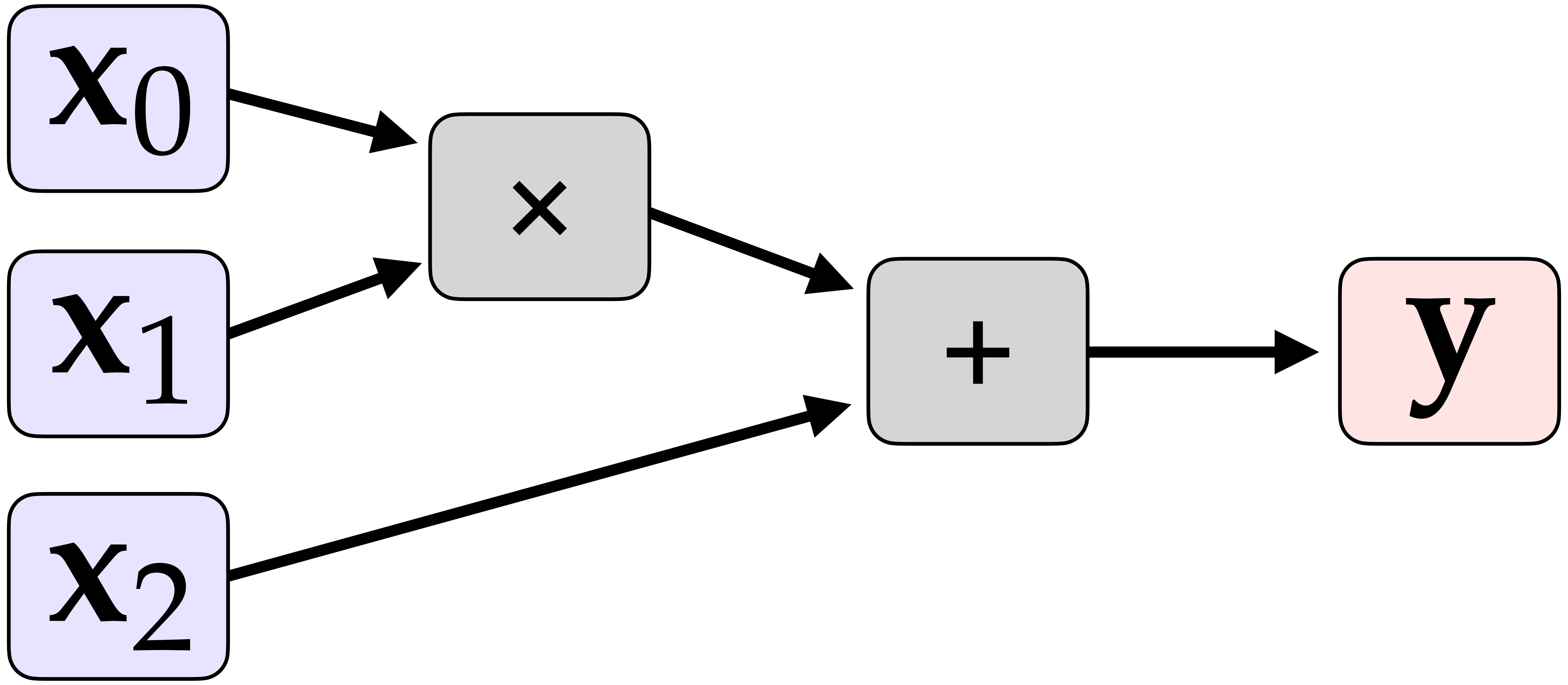
$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \in \mathbb{R}^{1000000 \times 1000000}$$

(~3.6 TiB)

DIRECTIONALITY OF DIFFERENTIATION

Forward mode

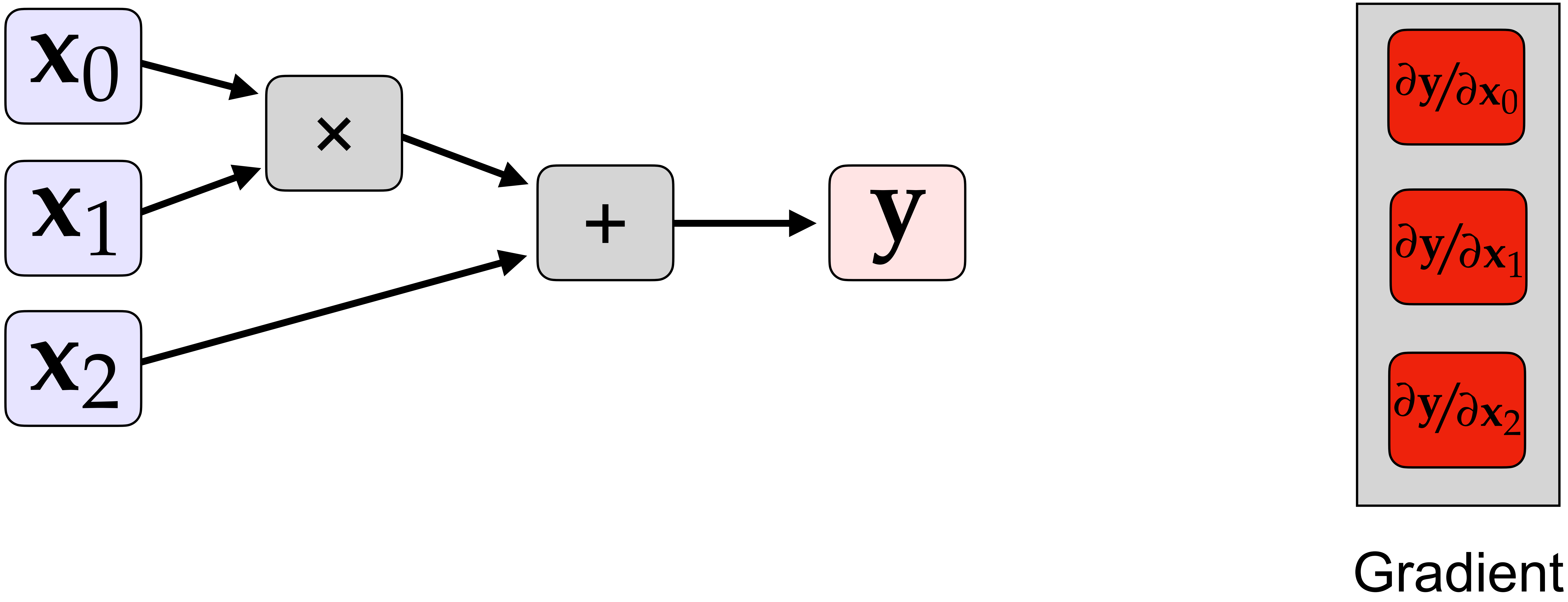
$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



DIRECTIONALITY OF DIFFERENTIATION

Forward mode

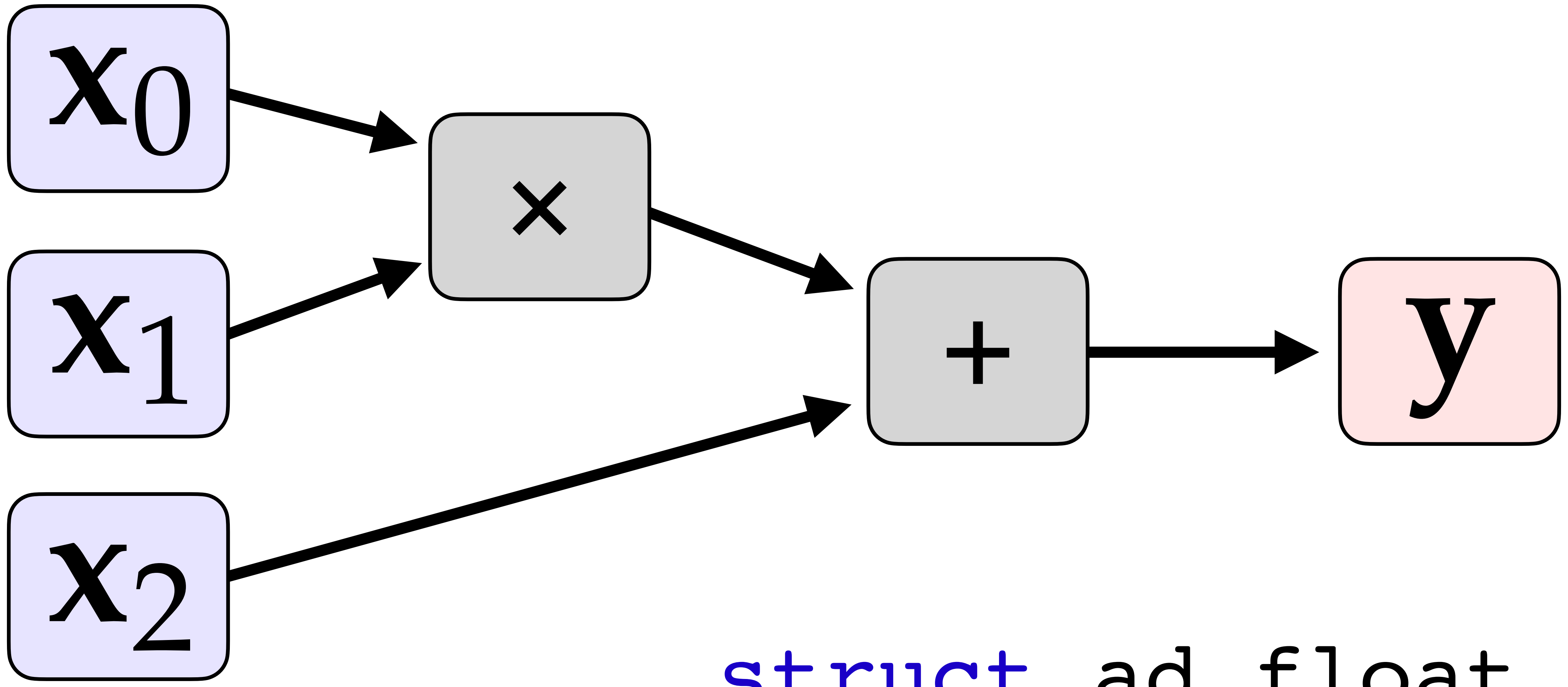
$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



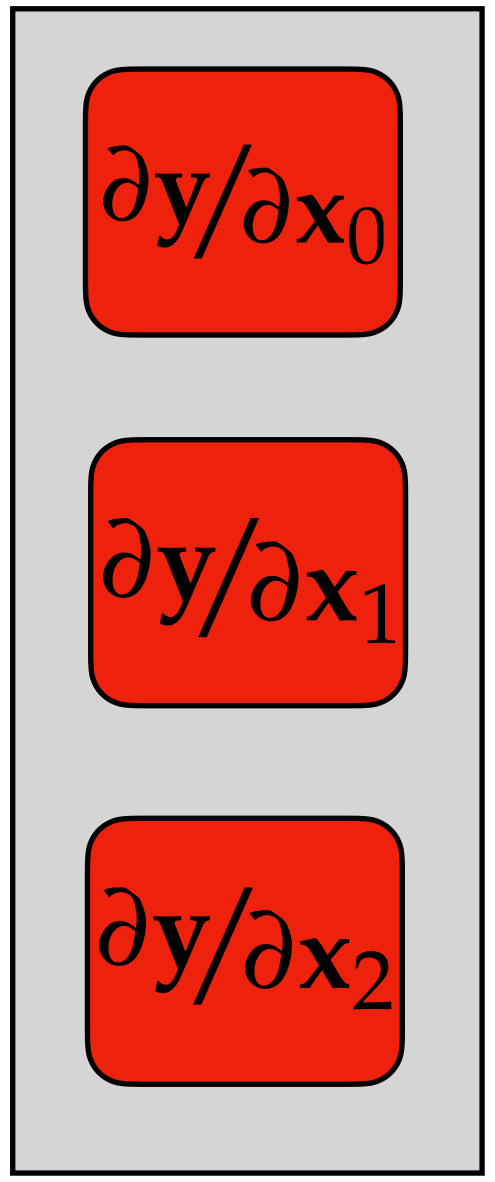
DIRECTIONALITY OF DIFFERENTIATION

Forward mode

$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



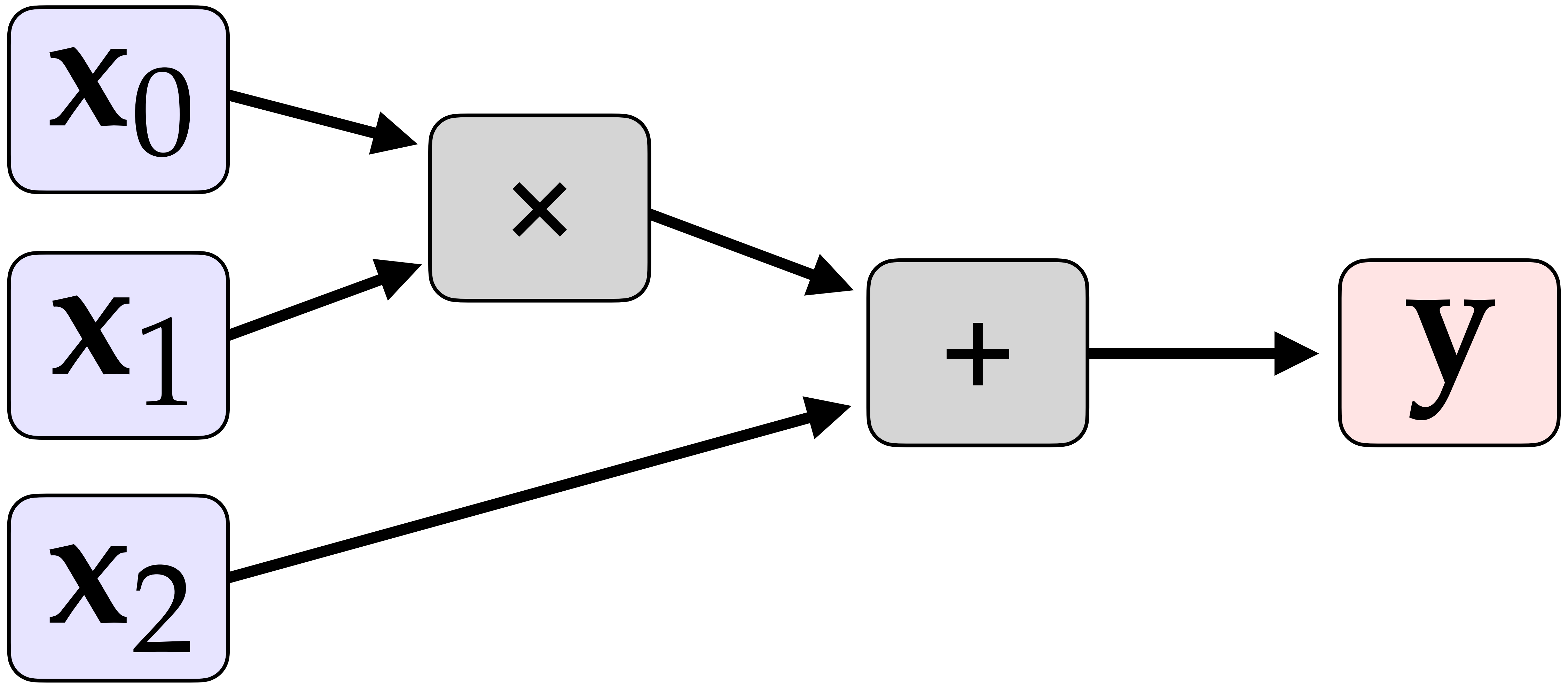

```
struct ad_float {
    float value;
    float derivative;
};
```



Gradient

DIRECTIONALITY OF DIFFERENTIATION

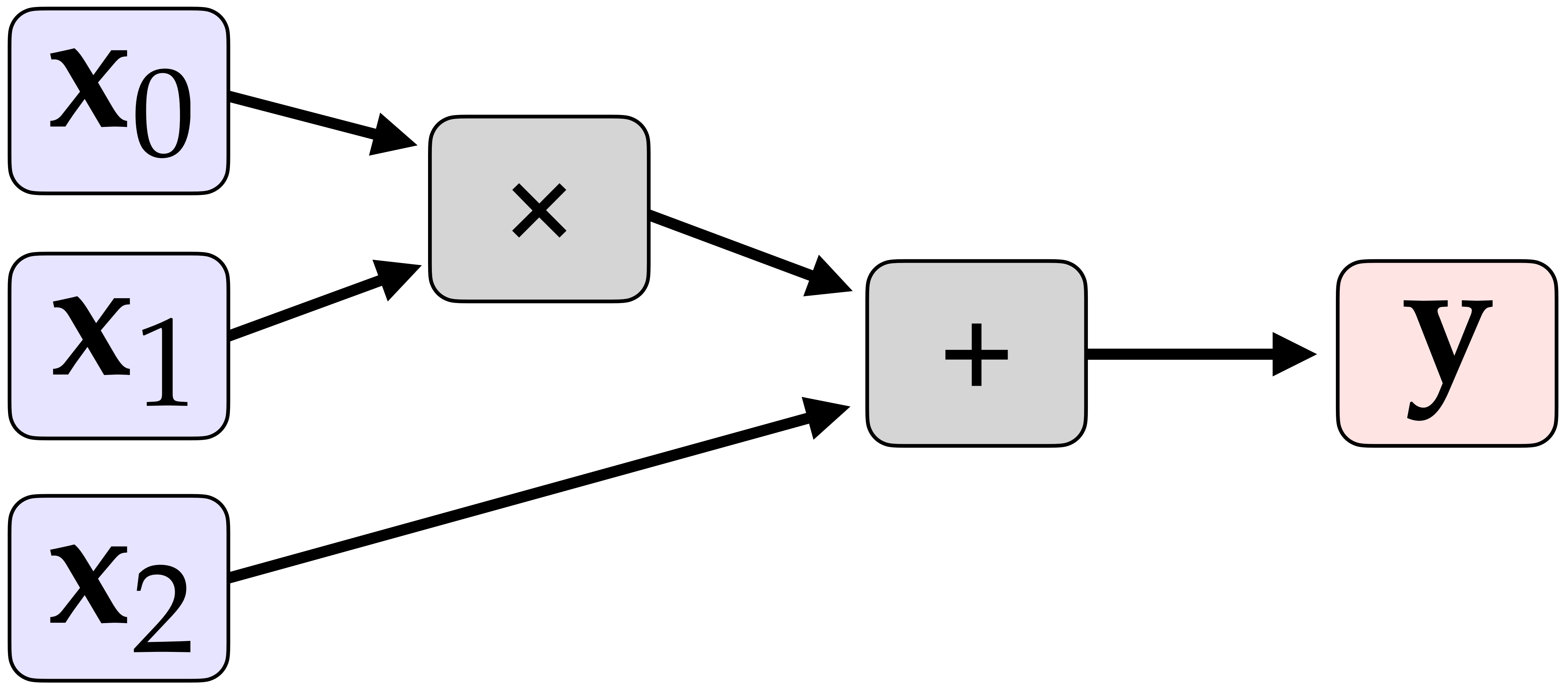
$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

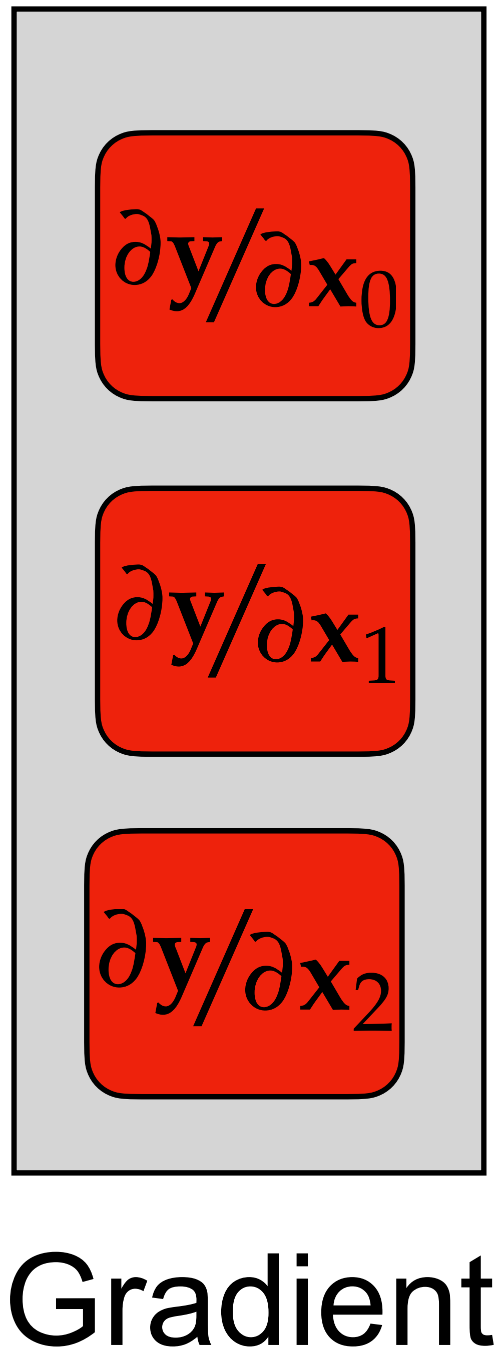
$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$

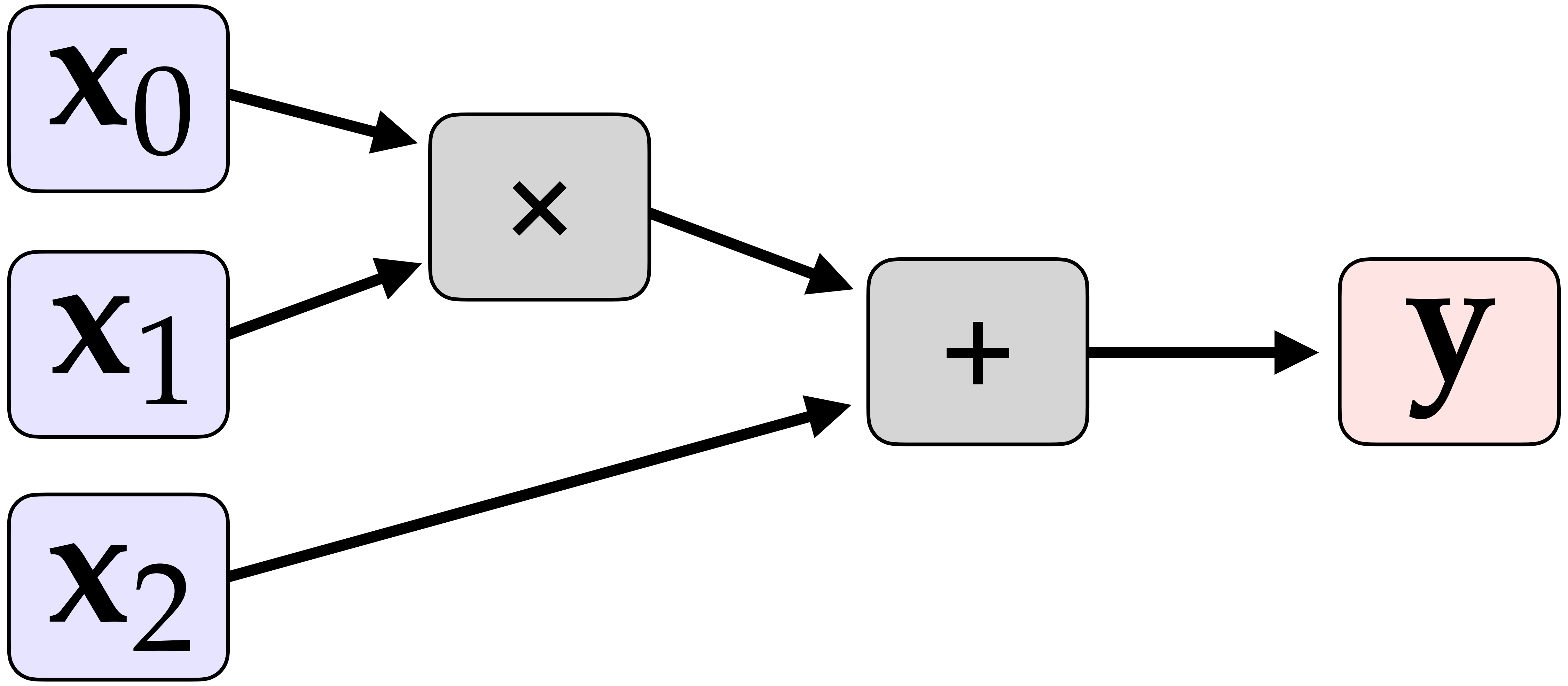


DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$

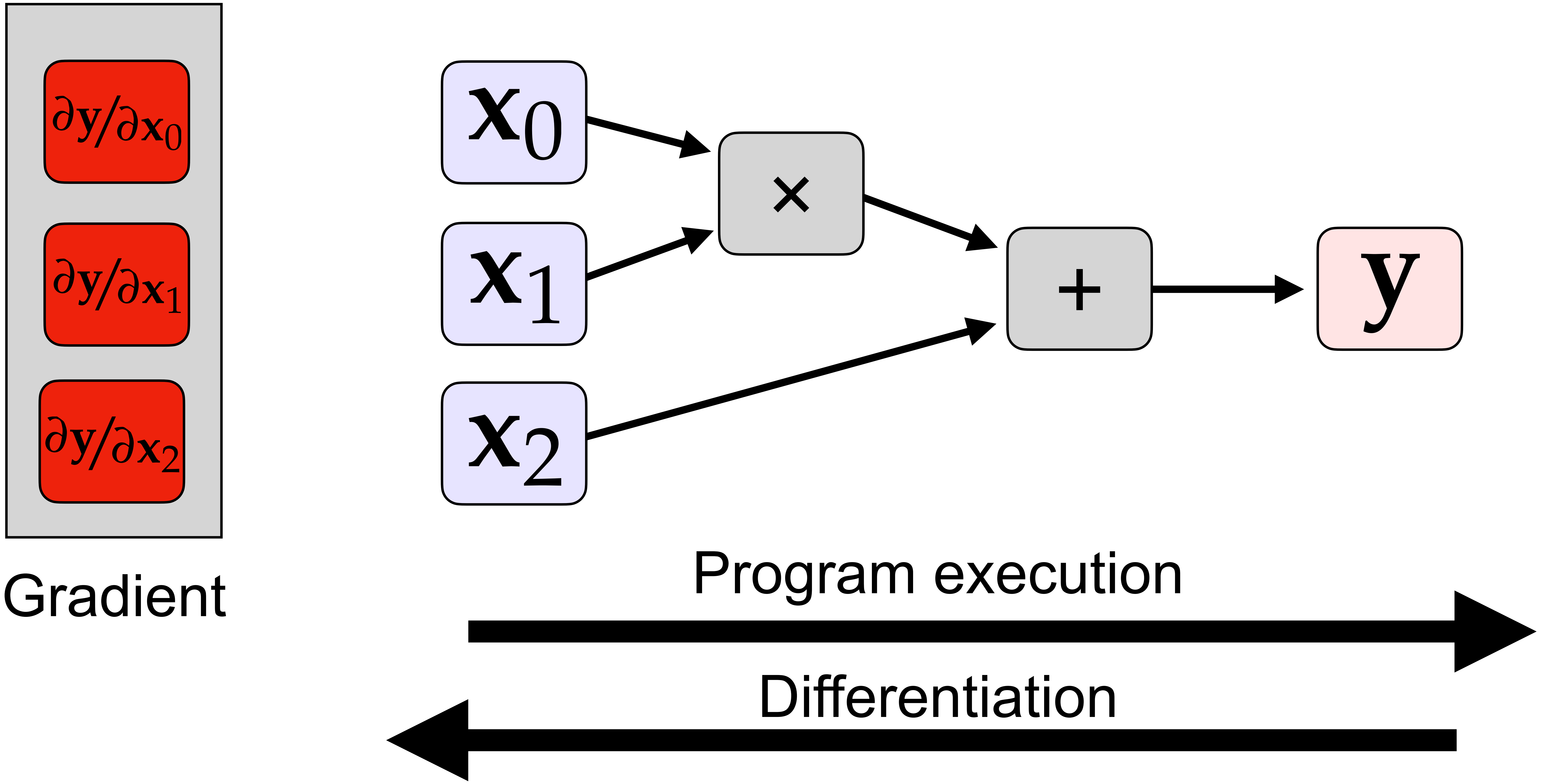




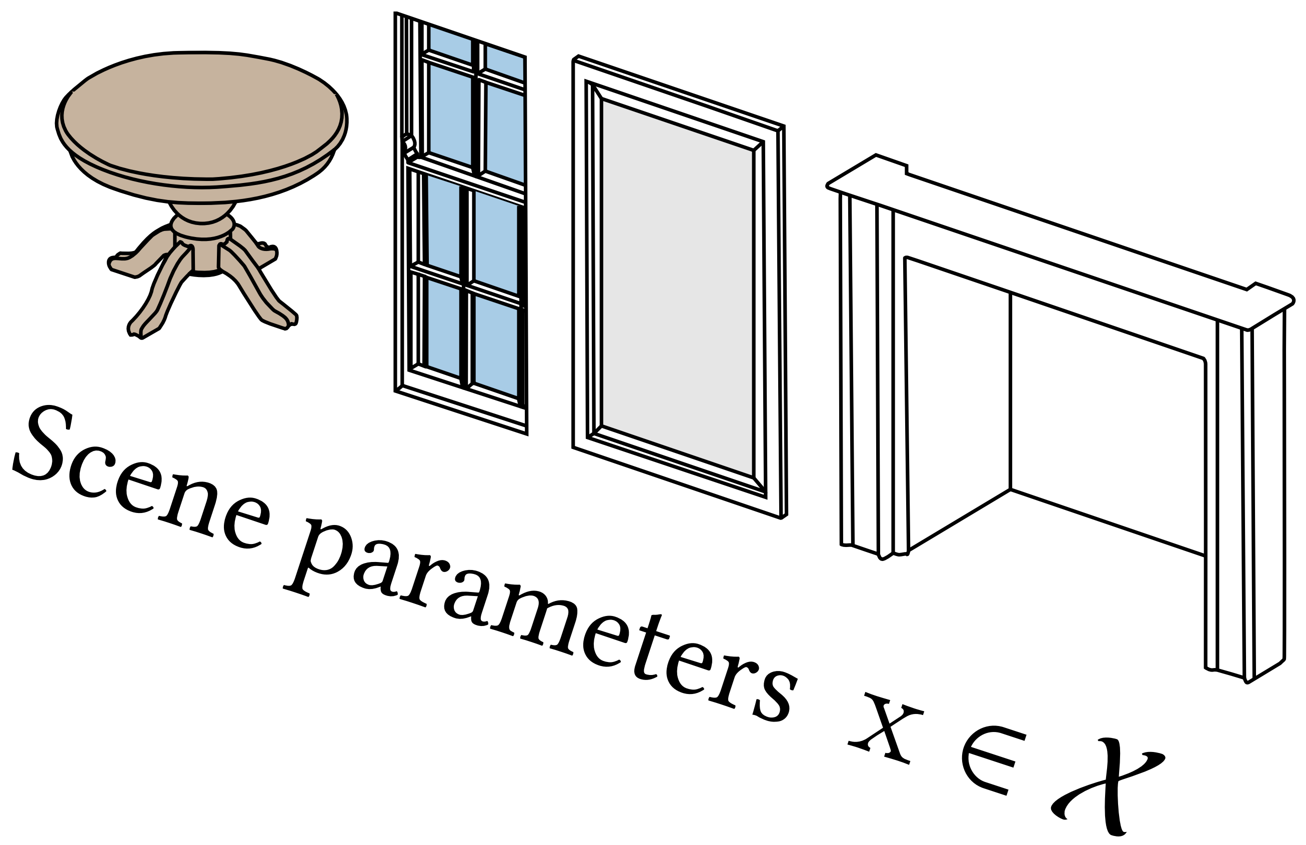
DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

$$\mathbf{y} = \mathbf{x}_0 \cdot \mathbf{x}_1 + \mathbf{x}_2$$



Autodiff-based differentiable rendering

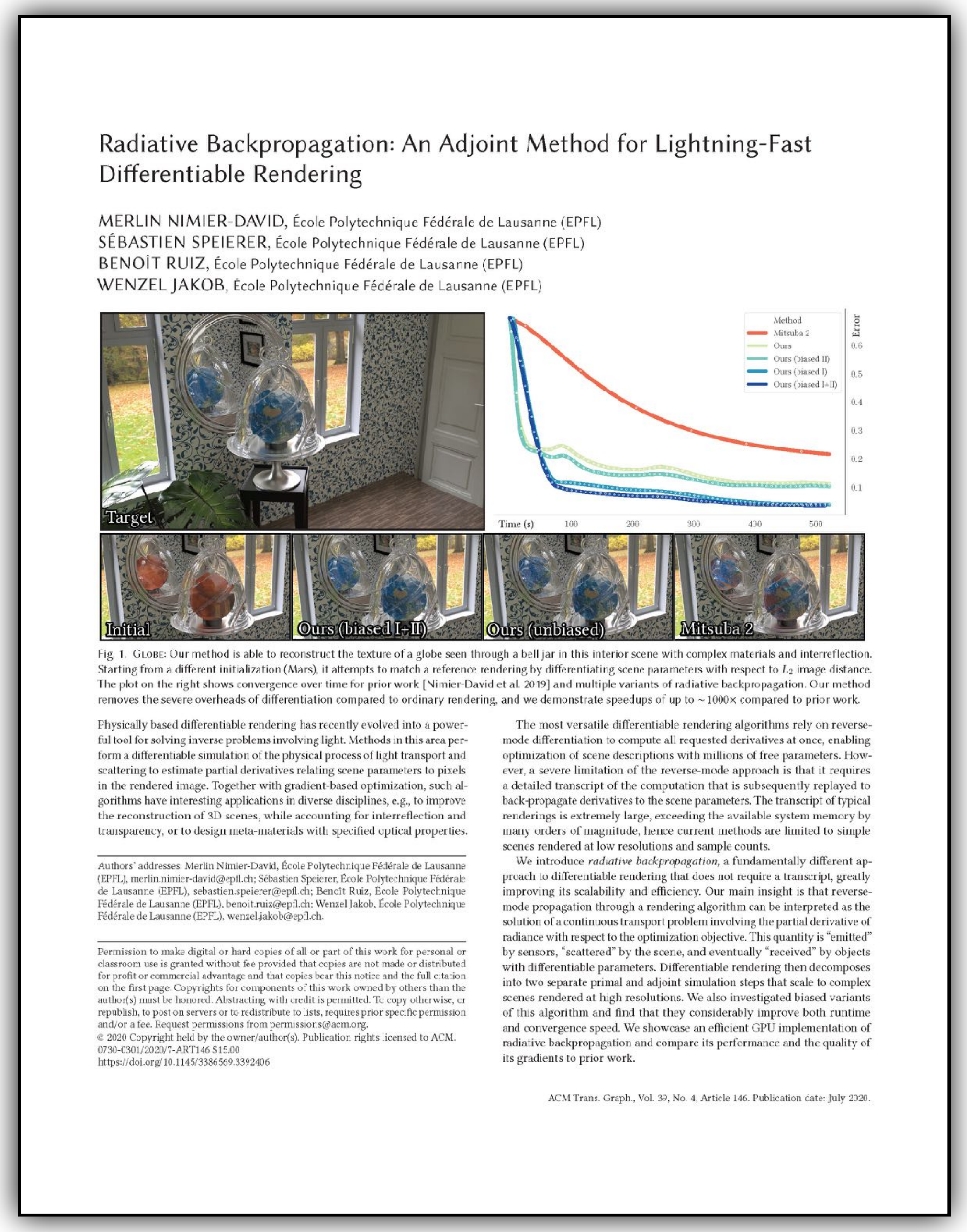


Rendering algorithm

Autodiff-based differentiable rendering

Vector explosion by *upklyak* at Freepik

RADIATIVE BACKPROPAGATION



Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering

MERLIN NIMIER-DAVID, École Polytechnique Fédérale de Lausanne (EPFL)
SÉBASTIEN SPEIERER, École Polytechnique Fédérale de Lausanne (EPFL)
BENOÎT RUIZ, École Polytechnique Fédérale de Lausanne (EPFL)
WENZEL JAKOB, École Polytechnique Fédérale de Lausanne (EPFL)

Fig. 1. GLOBE: Our method is able to reconstruct the texture of a globe seen through a bell jar in this interior scene with complex materials and interreflection. Starting from a different initialization (Mars), it attempts to match a reference rendering by differentiating scene parameters with respect to L_2 image distance. The plot on the right shows convergence over time for prior work [Nimier-David et al. 2019] and multiple variants of radiative backpropagation. Our method removes the severe overheads of differentiation compared to ordinary rendering, and we demonstrate speedups of up to ~1000× compared to prior work.

Physically based differentiable rendering has recently evolved into a powerful tool for solving inverse problems involving light. Methods in this area perform a differentiable simulation of the physical process of light transport and scattering to estimate partial derivatives relating scene parameters to pixels in the rendered image. Together with gradient-based optimization, such algorithms have interesting applications in diverse disciplines, e.g., to improve the reconstruction of 3D scenes, while accounting for interreflection and transparency, or to design meta-materials with specified optical properties.

The most versatile differentiable rendering algorithms rely on reverse-mode differentiation to compute all requested derivatives at once, enabling optimization of scene descriptions with millions of free parameters. However, a severe limitation of the reverse-mode approach is that it requires a detailed transcript of the computation that is subsequently replayed to back-propagate derivatives to the scene parameters. The transcript of typical renderings is extremely large, exceeding the available system memory by many orders of magnitude, hence current methods are limited to simple scenes rendered at low resolutions and sample counts.

We introduce *radiative backpropagation*, a fundamentally different approach to differentiable rendering that does not require a transcript, greatly improving its scalability and efficiency. Our main insight is that reverse-mode propagation through a rendering algorithm can be interpreted as the solution of a continuous transport problem involving the partial derivative of radiance with respect to the optimization objective. This quantity is "emitted" by sensors, "scattered" by the scene, and eventually "received" by objects with differentiable parameters. Differentiable rendering then decomposes into two separate primal and adjoint simulation steps that scale to complex scenes rendered at high resolutions. We also investigated biased variants of this algorithm and find that they considerably improve both runtime and convergence speed. We showcase an efficient GPU implementation of radiative backpropagation and compare its performance and the quality of its gradients to prior work.

Authors' addresses: Merlin Nimier-David, École Polytechnique Fédérale de Lausanne (EPFL), merlin.nimier-david@epfl.ch; Sébastien Speierer, École Polytechnique Fédérale de Lausanne (EPFL), sebastien.speierer@epfl.ch; Benoît Ruiz, École Polytechnique Fédérale de Lausanne (EPFL), benoit.ruiz@epfl.ch; Wenzel Jakob, École Polytechnique Fédérale de Lausanne (EPFL), wenzel.jakob@epfl.ch.

0730-0301/2020/7-ART146 $15.00
https://doi.org/10.1145/3386569.3392406

ACM Trans. Graph., Vol. 39, No. 4, Article 146. Publication date: July 2020.

Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering

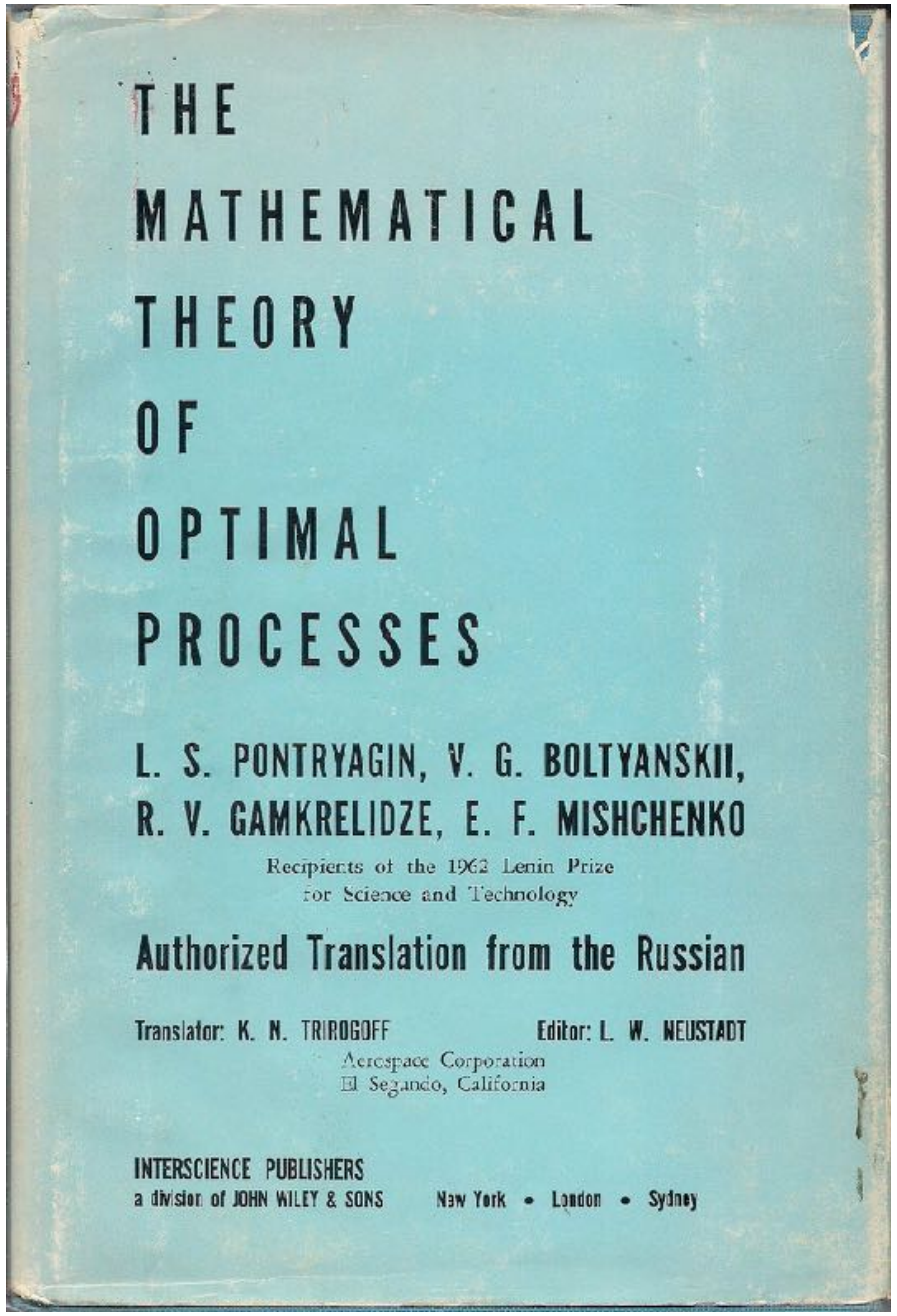
Merlin Nimier-David, Sébastien Speierer, Benoit Ruîz, Wenzel Jakob

SIGGRAPH 2020

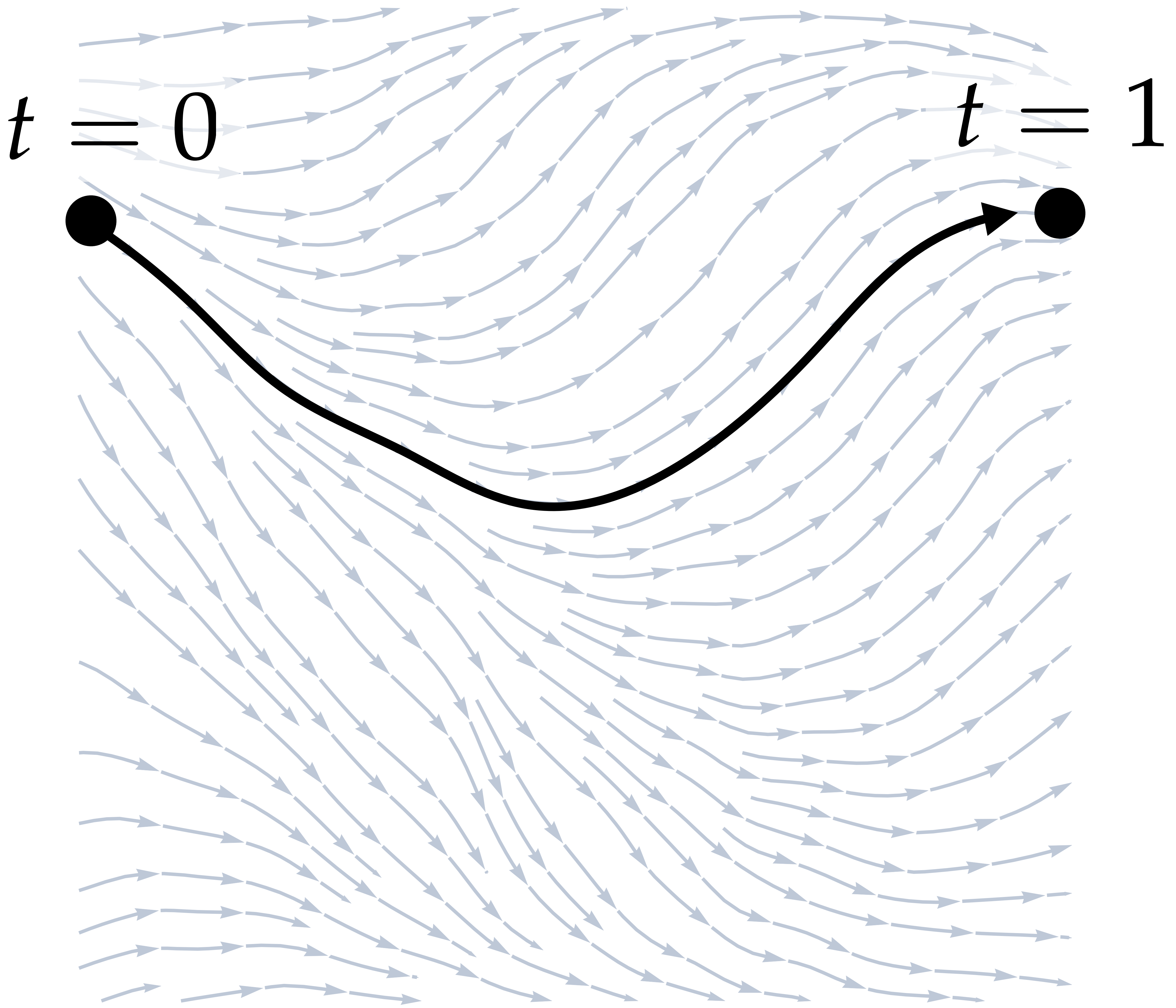
MOTIVATION: ADJOINT SENSITIVITY METHOD

For problems with a time dimension (ODEs, ..)

Pontryagin et al.
1962

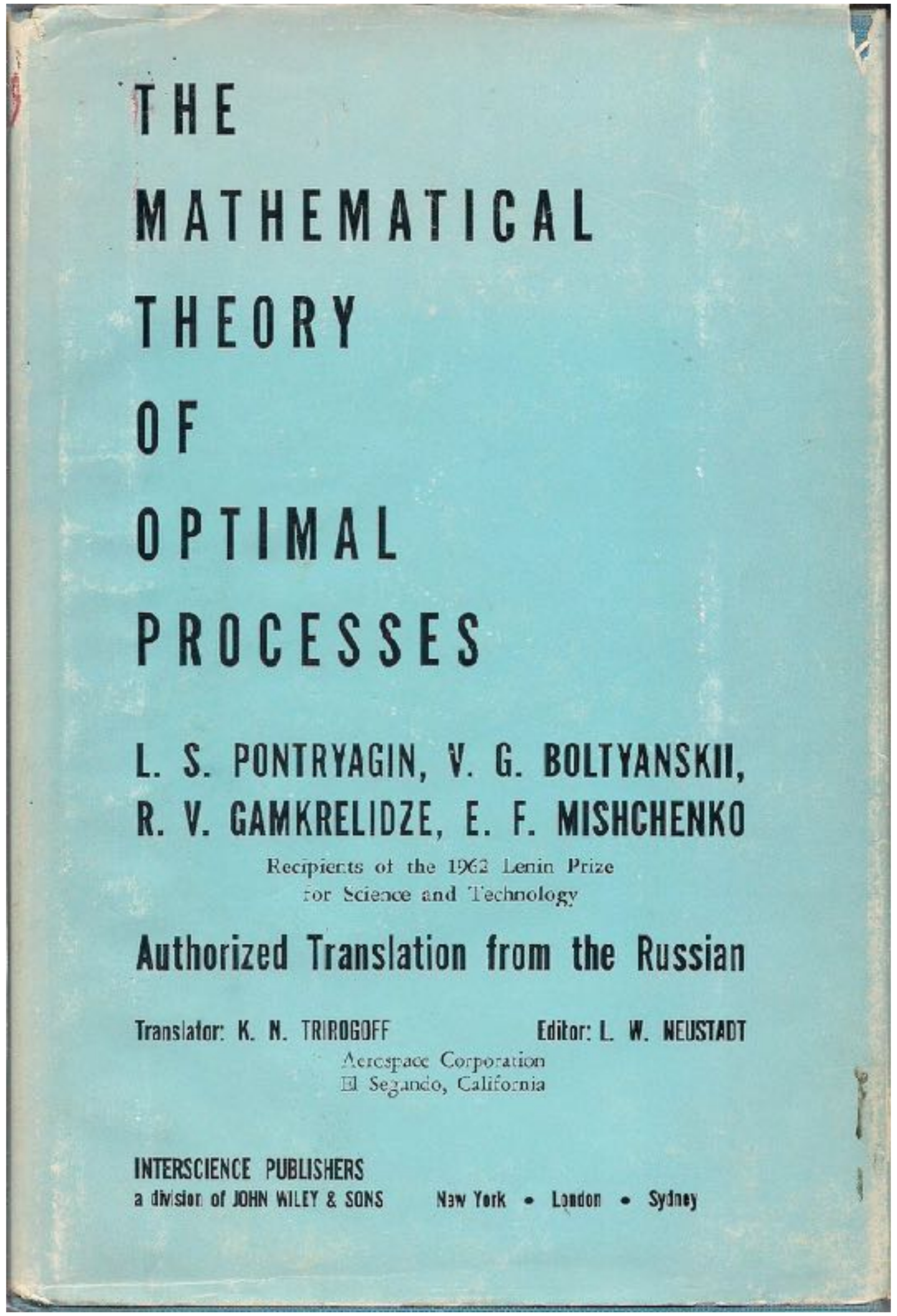


MOTIVATION: ADJOINT SENSITIVITY METHOD

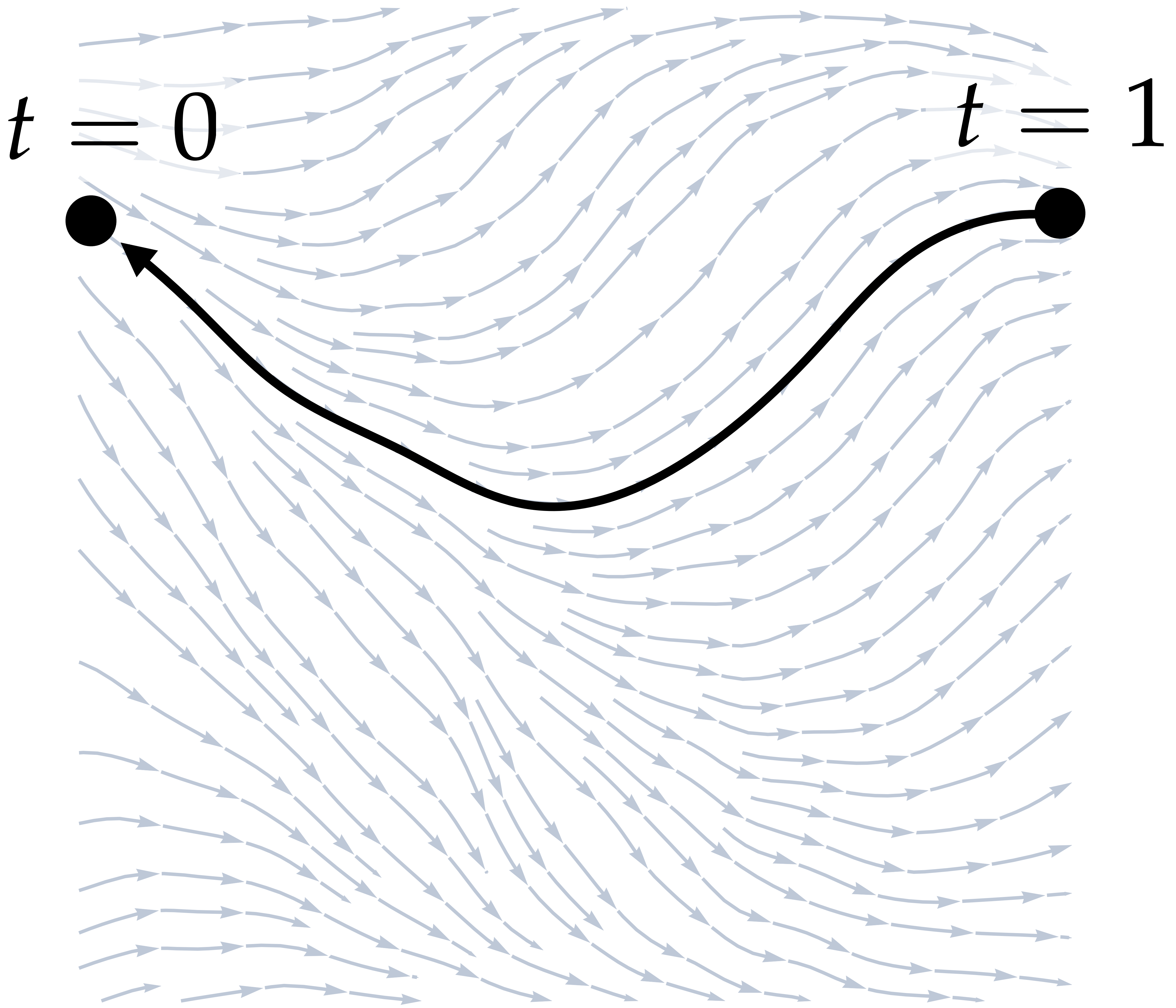


For problems with a time dimension (ODEs, ..)

Pontryagin et al.
1962

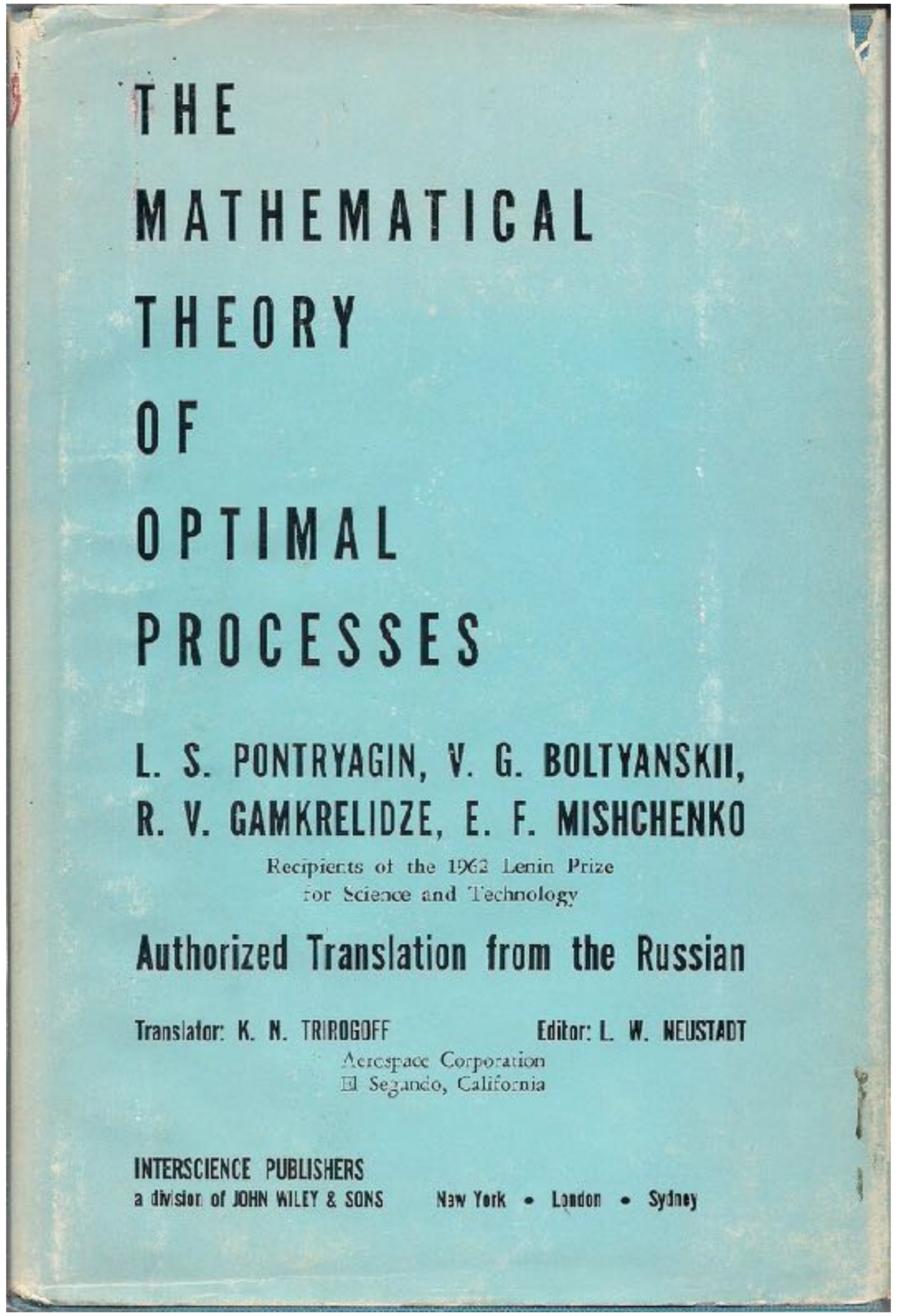


MOTIVATION: ADJOINT SENSITIVITY METHOD

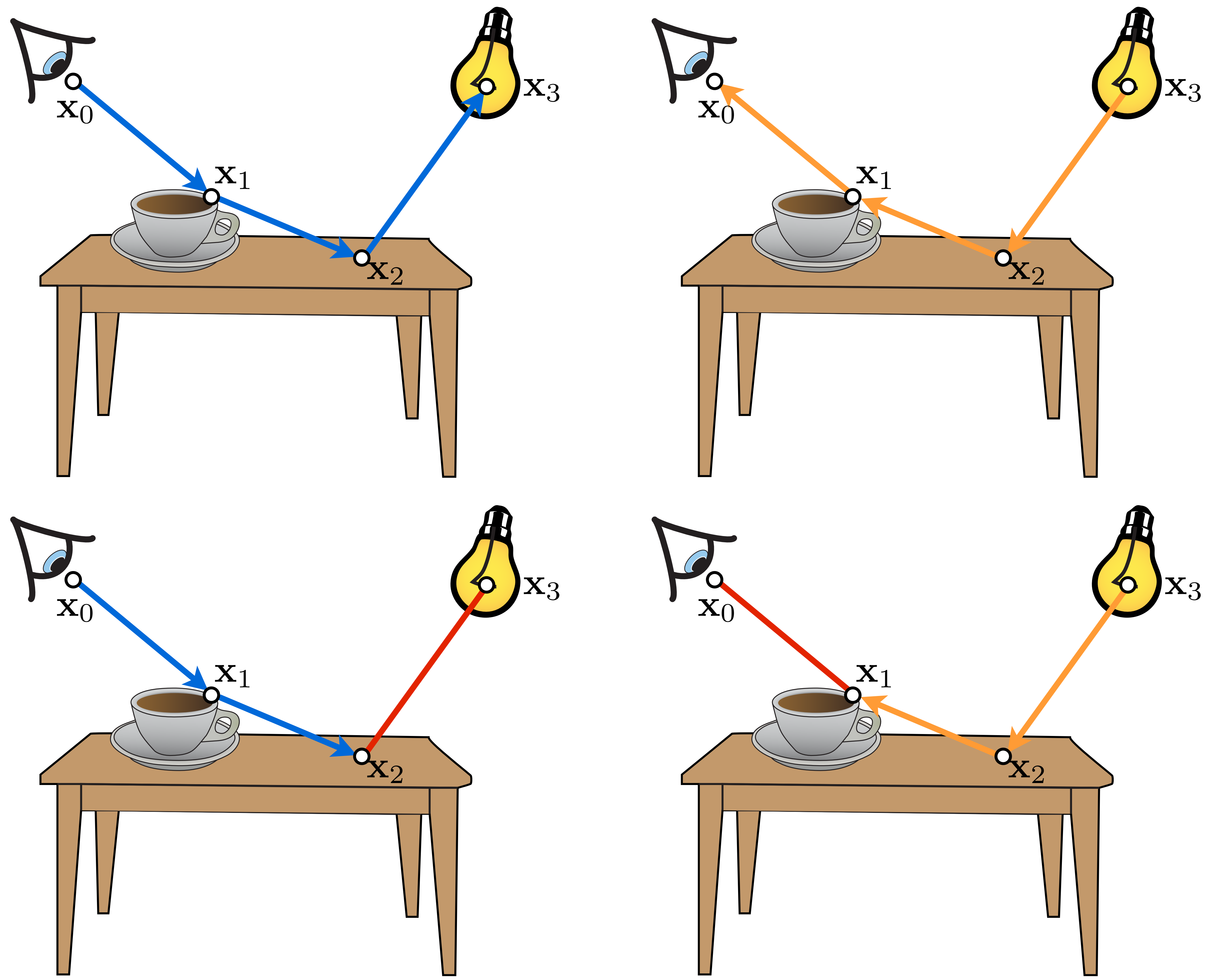


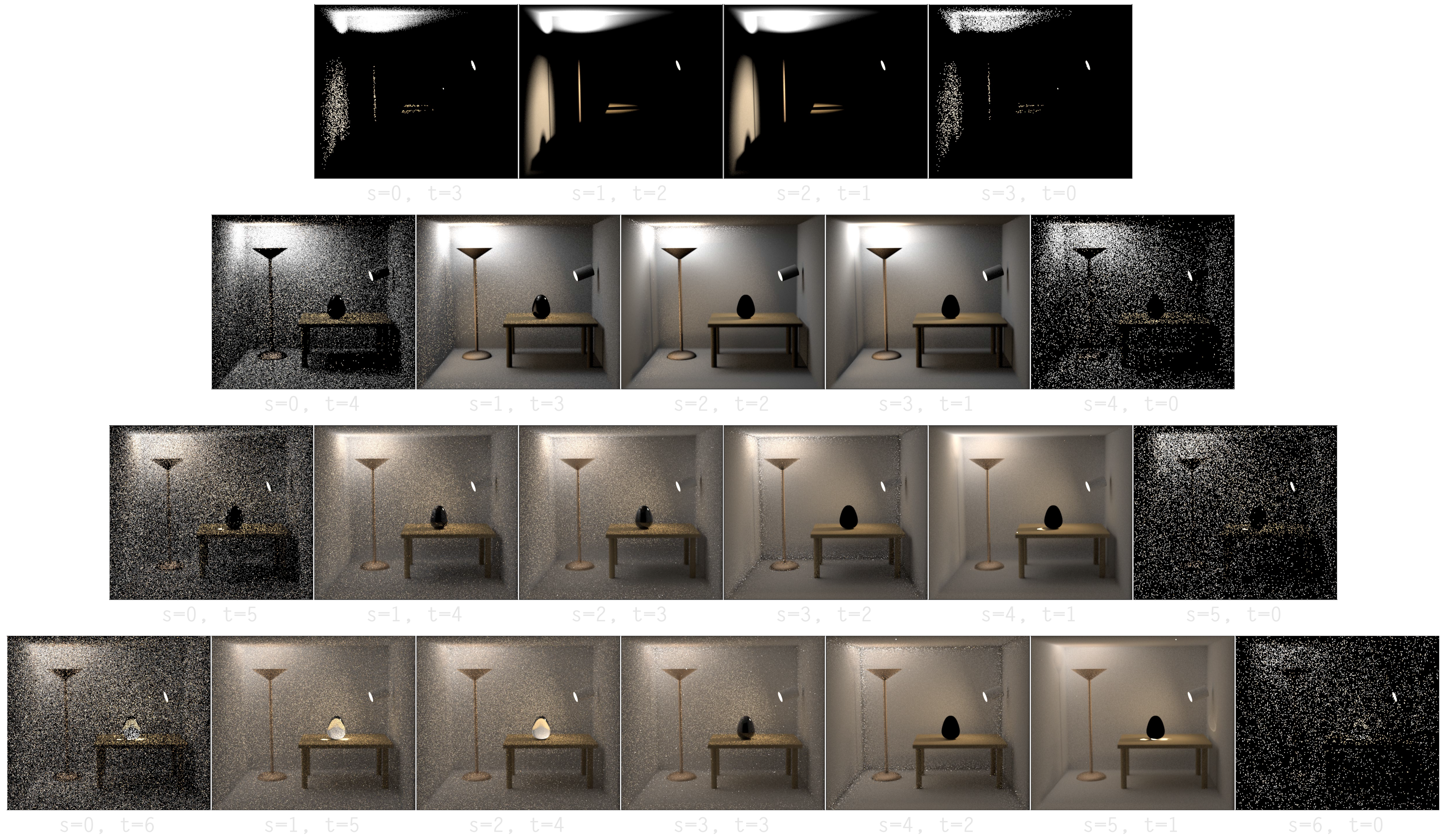
For problems with a time dimension (ODEs, ..)

Pontryagin et al.
1962



"ADJOINT" – THAT SOUNDS FAMILIAR!





Bidirectional Estimators for Light Transport

Veach & Guibas, **1994**

$$\langle O\mathbf{a}, \mathbf{b} \rangle = \langle \mathbf{a}, O\mathbf{b} \rangle$$

(Underlying principle: self-adjoint operators)

Derivatives projected into the scene


Gradients
Deriv. from objects
Deriv. from sensor
Product integral


Gradients
Deriv. from objects
Deriv. from sensor
Product integral


Gradients
Deriv. from objects
Deriv. from sensor
Product integral


$f_s : (\omega_i, \omega_o, \underbrace{x_1, x_2, \ldots}_{\text{parameters}}) \mapsto \mathbb{R}$
parameters


$f_s : (\omega_i, \omega_o, x_1, x_2, \ldots) \mapsto \mathbb{R}$
parameters

ANOTHER PERSPECTIVE

Normal rendering

ANOTHER PERSPECTIVE

Normal rendering

- Transporting from sensor/light may yield lower variance.

ANOTHER PERSPECTIVE

Normal rendering

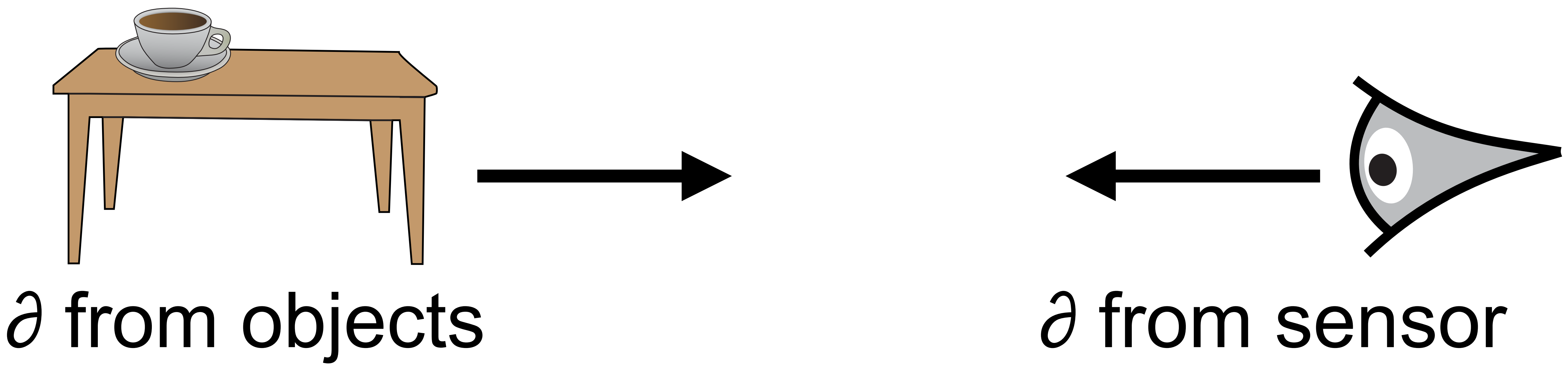
- Transporting from sensor/light may yield lower variance.

ANOTHER PERSPECTIVE

Normal rendering

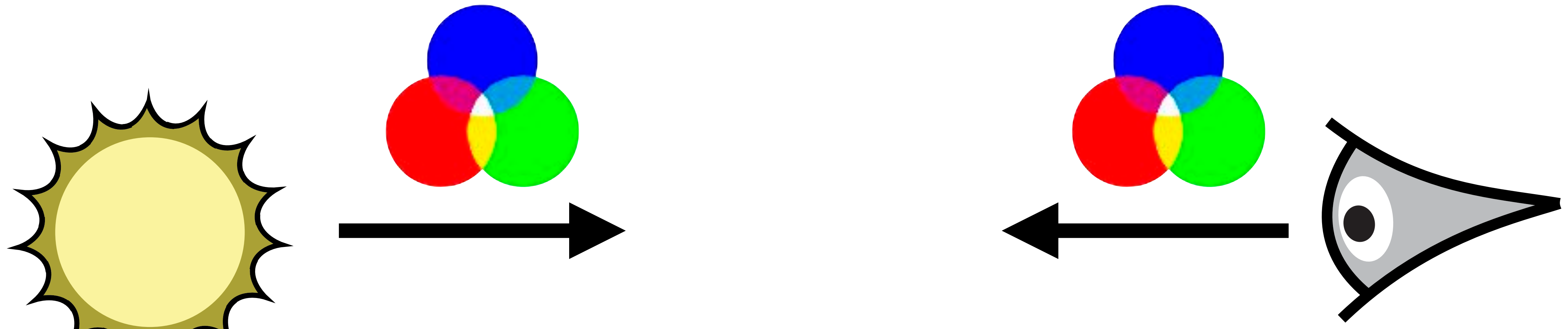
- Transporting from sensor/light may yield lower variance.

Differentiable rendering



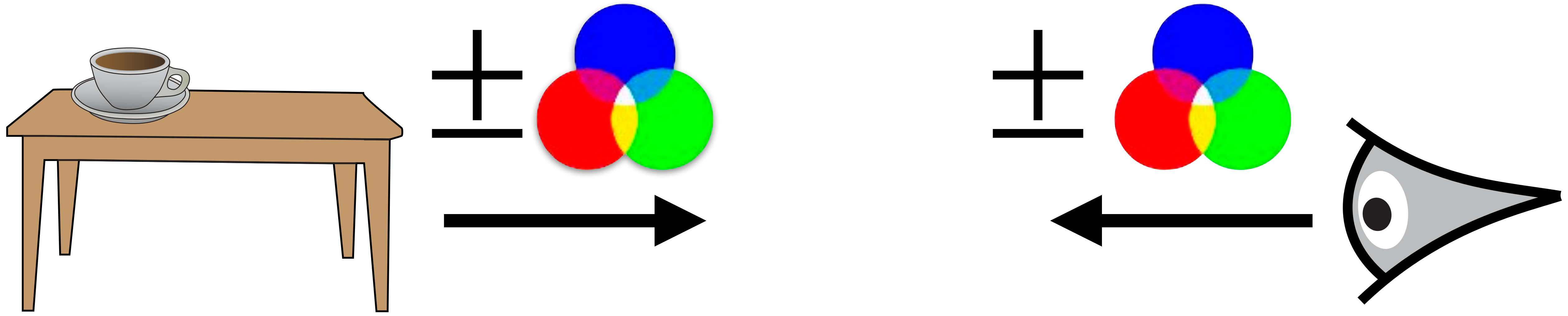
ANOTHER PERSPECTIVE

Normal rendering



- Transporting from sensor/light may yield lower variance.

Differentiable rendering



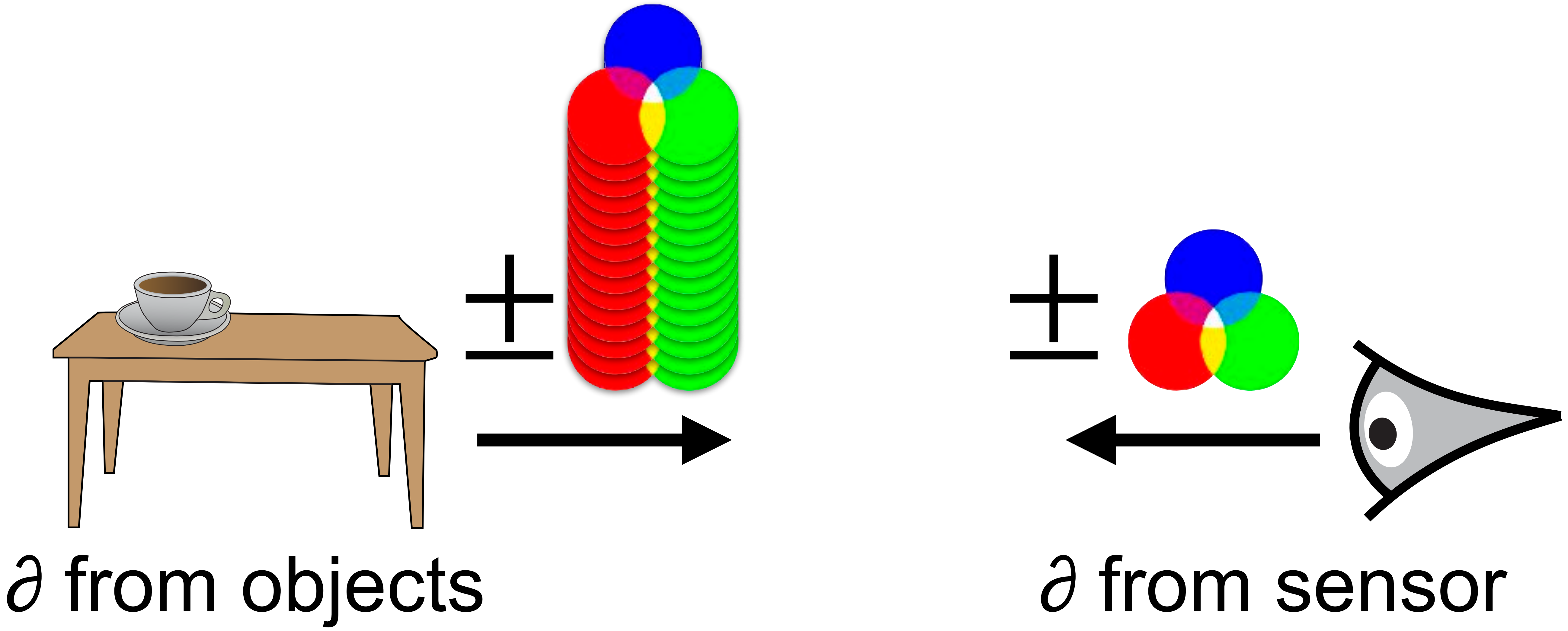
ANOTHER PERSPECTIVE

Normal rendering

- Transporting from sensor/light may yield lower variance.

Differentiable rendering

- Transporting from objects is **completely impractical**.



Surface texture optimization

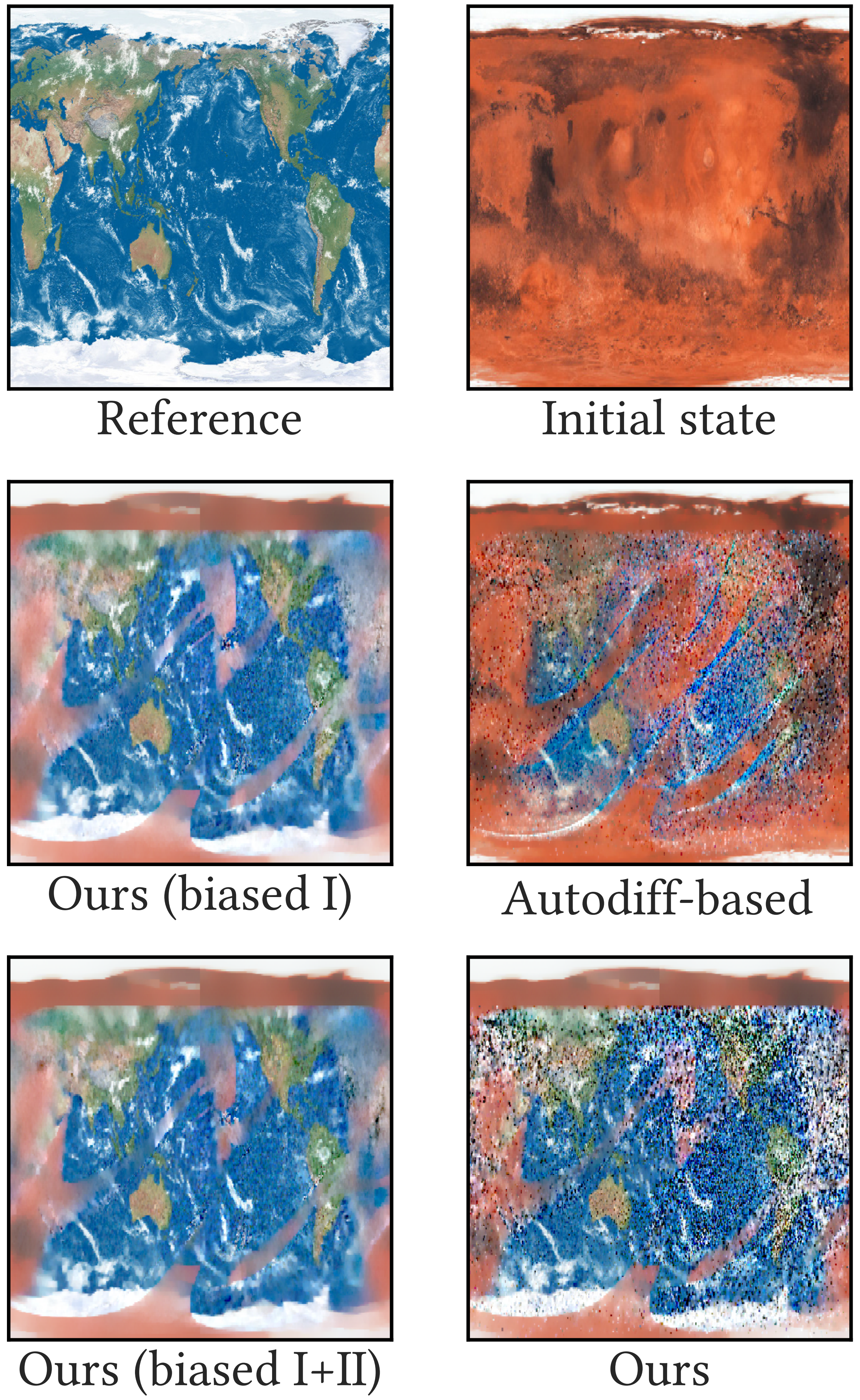
Initial state

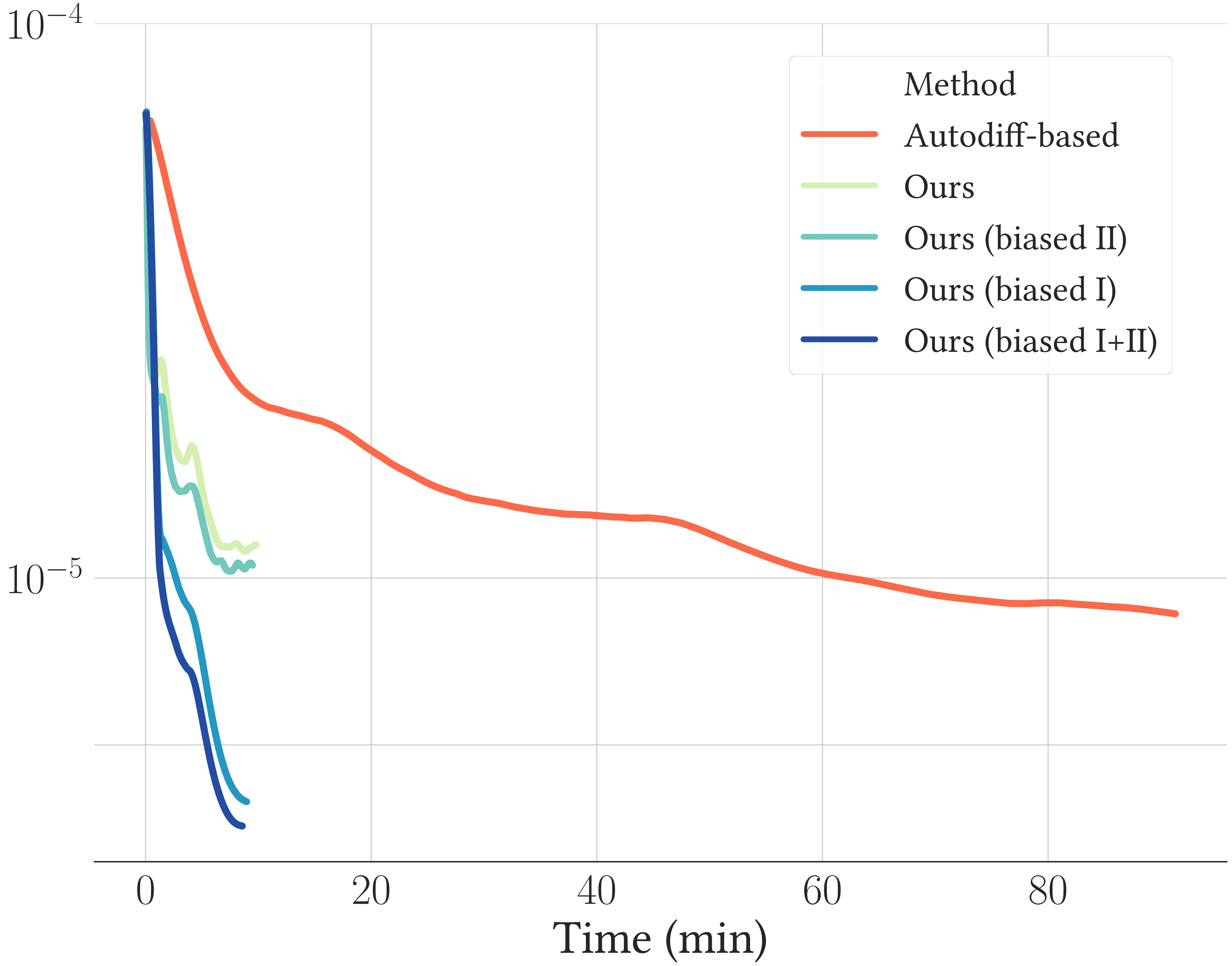
Target state


Optimized texture
Target

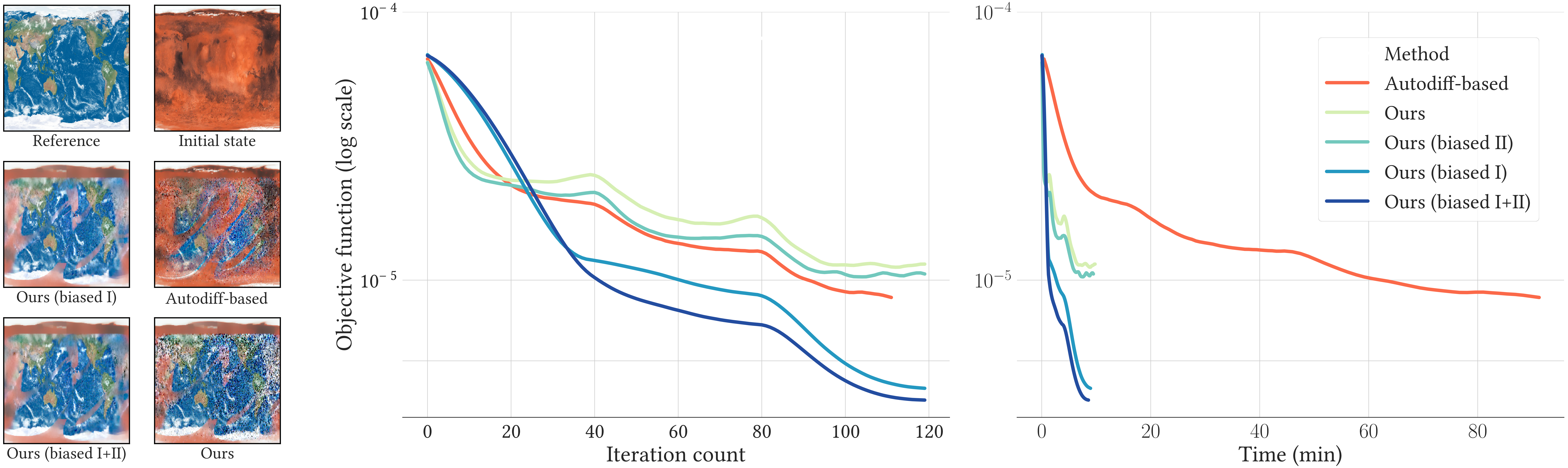

Optimized texture
Target

Surface BSDF optimization





Surface BSDF optimization



Volume density optimization

Mitsuba 2 (AD-based)

Radiative Backprop. (biased I + II)

Target

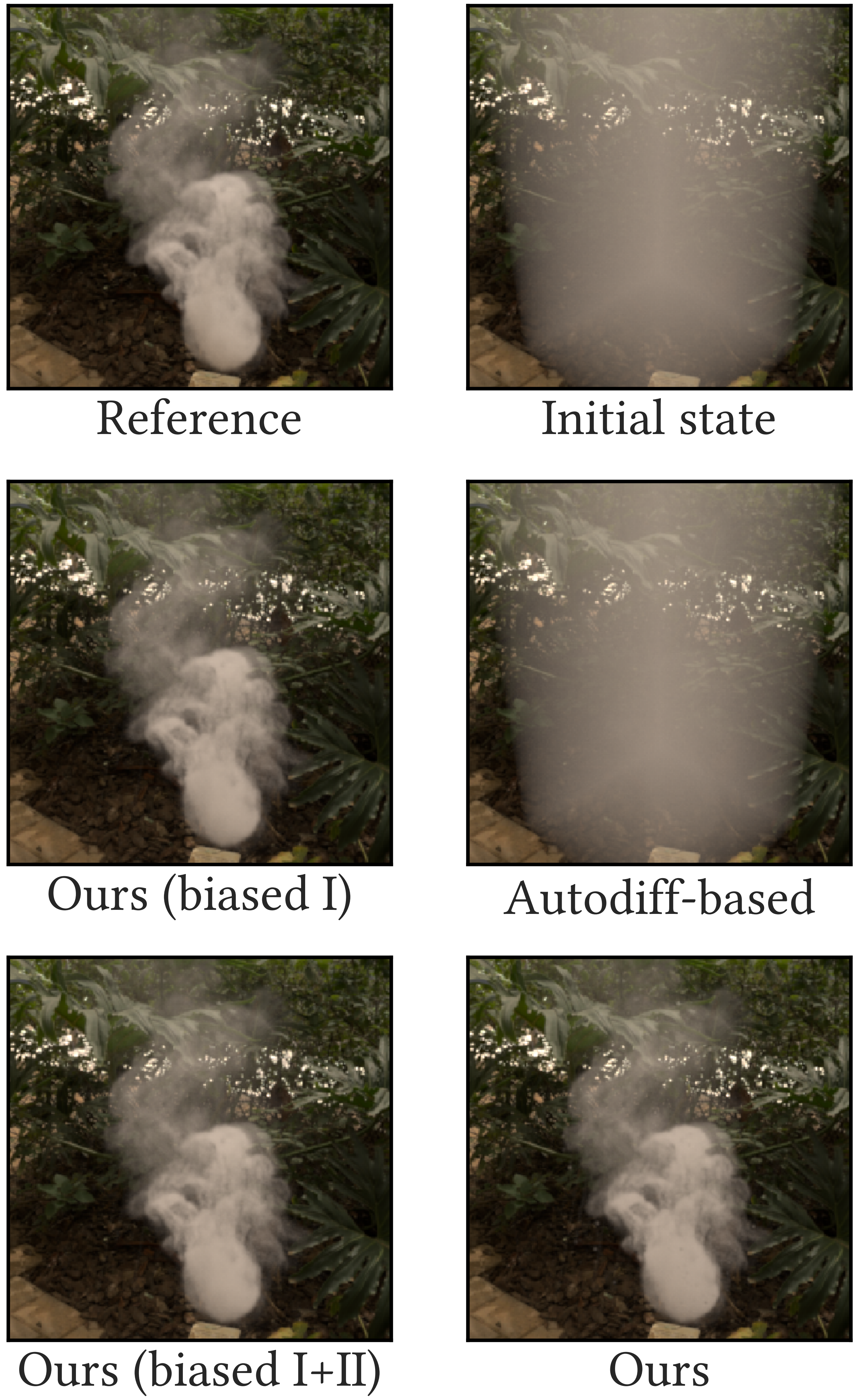
Volume density optimization

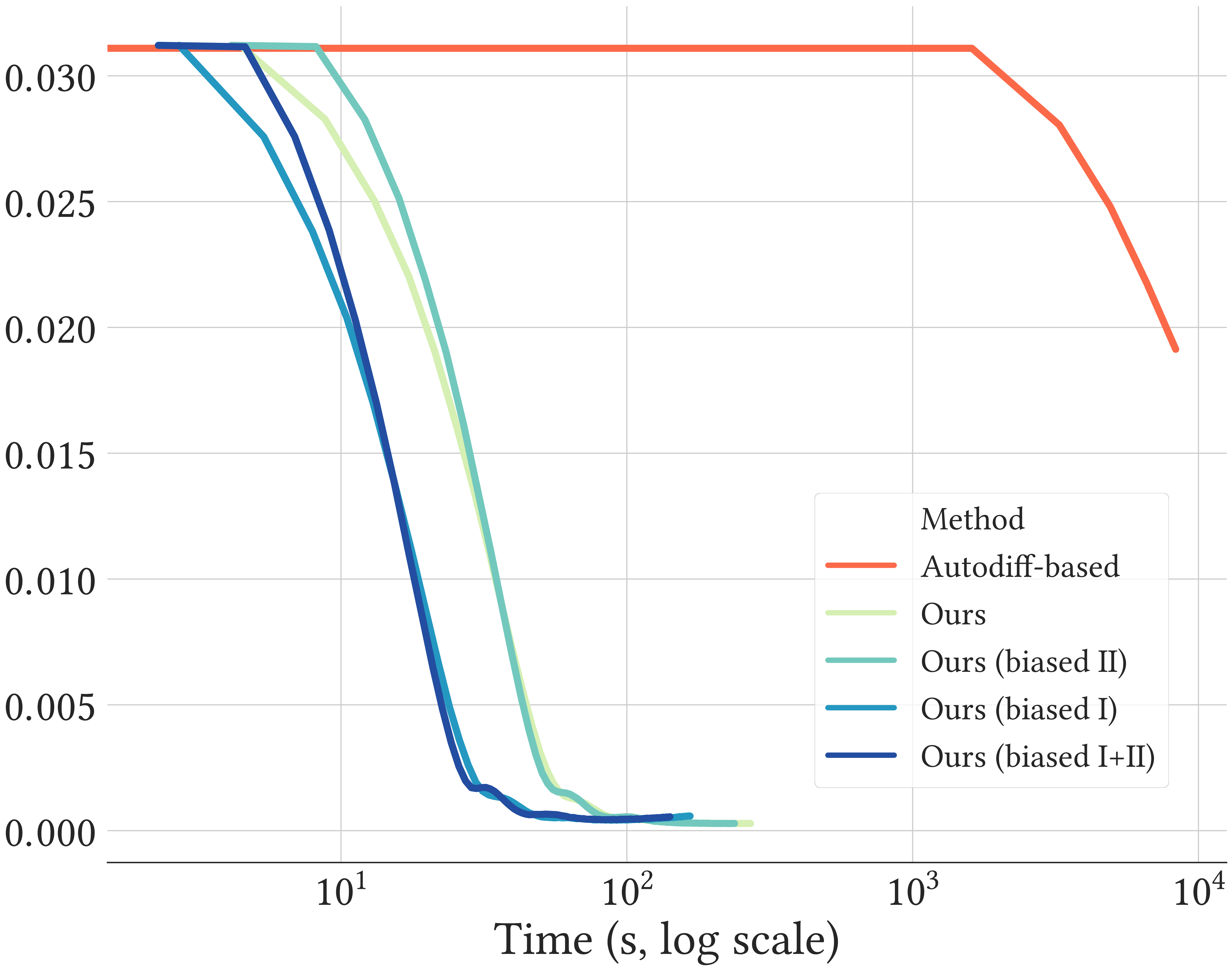
Mitsuba 2 (AD-based)

Radiative Backprop.
(biased I + II)

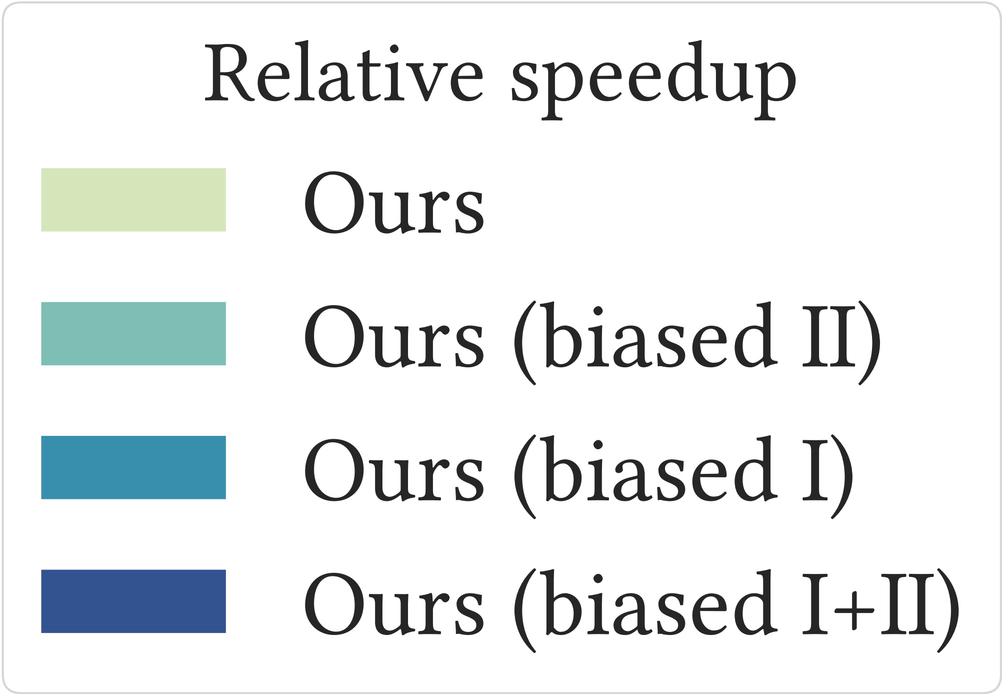
Target

Volume density optimization

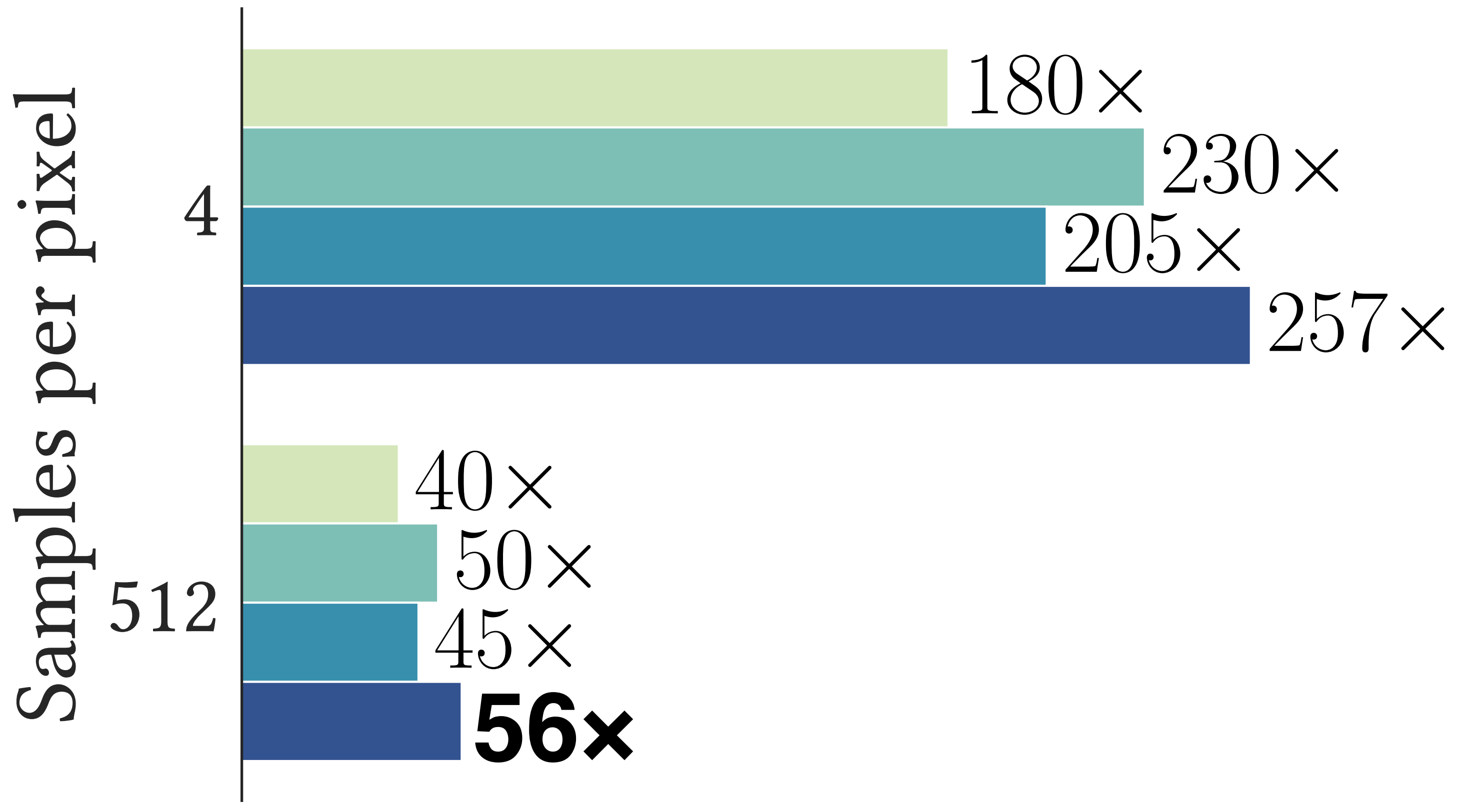




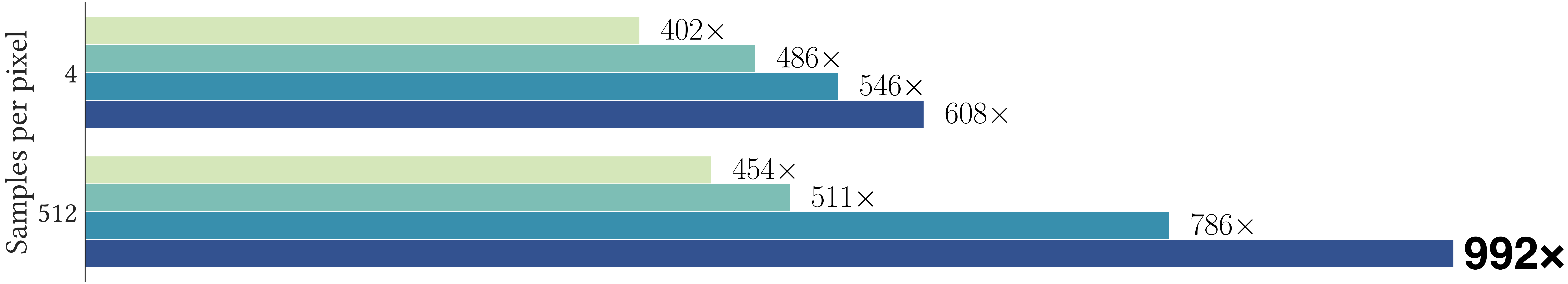
Relative speedups *vs* autodiff-based



Surface texture optimization



Volume density optimization



TL;DR

- Radiative Backpropagation is **"just" another kind** of light transport simulation with weird sensors and emitters.
  - Orders of magnitude faster (up to ~1000× in our experiments)
  - Lifts memory limitations entirely
  - Only need to differentiate BSDFs etc. ("easy")
  - Can build on decades of research targeting such problems!

ACKNOWLEDGEMENTS

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

UCI

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

EPFL

TZU-MAO LI
POSTDOCTORAL RESEARCHER
MIT CSAIL, Cambridge
(joining UCSD as an assistant professor in 2021)

ACKNOWLEDGEMENTS

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

TZU-MAO LI
POSTDOCTORAL RESEARCHER
MIT CSAIL, Cambridge
(joining UCSD as an assistant professor in 2021)

ACKNOWLEDGEMENTS

SHUANG ZHAO

ASSISTANT PROFESSOR

University of California, Irvine

WENZEL JAKOB

ASSISTANT PROFESSOR

EPFL, Lausanne, Switzerland

TZU-MAO LI

POSTDOCTORAL RESEARCHER

MIT CSAIL, Cambridge

(joining UCSD as an assistant professor in 2021)

ACKNOWLEDGEMENTS

SHUANG ZHAO

ASSISTANT PROFESSOR

University of California, Irvine

WENZEL JAKOB

ASSISTANT PROFESSOR

EPFL, Lausanne, Switzerland

TZU-MAO LI

POSTDOCTORAL RESEARCHER

MIT CSAIL, Cambridge

(joining UCSD as an assistant professor in 2021)