

CVPR 2021 Tutorial

Physics-Based Differentiable Rendering

Speakers



Shuang Zhao

Assistant Professor
University of California, Irvine



Ioannis Gkioulekas

Assistant Professor
Carnegie Mellon University



Sai Bangaru

Ph.D. student
Massachusetts Institute of Technology

Talk Outline

- Introduction
- Differentiable rendering theory and algorithms
- Differentiable rendering systems and applications
- Q&A

Introduction

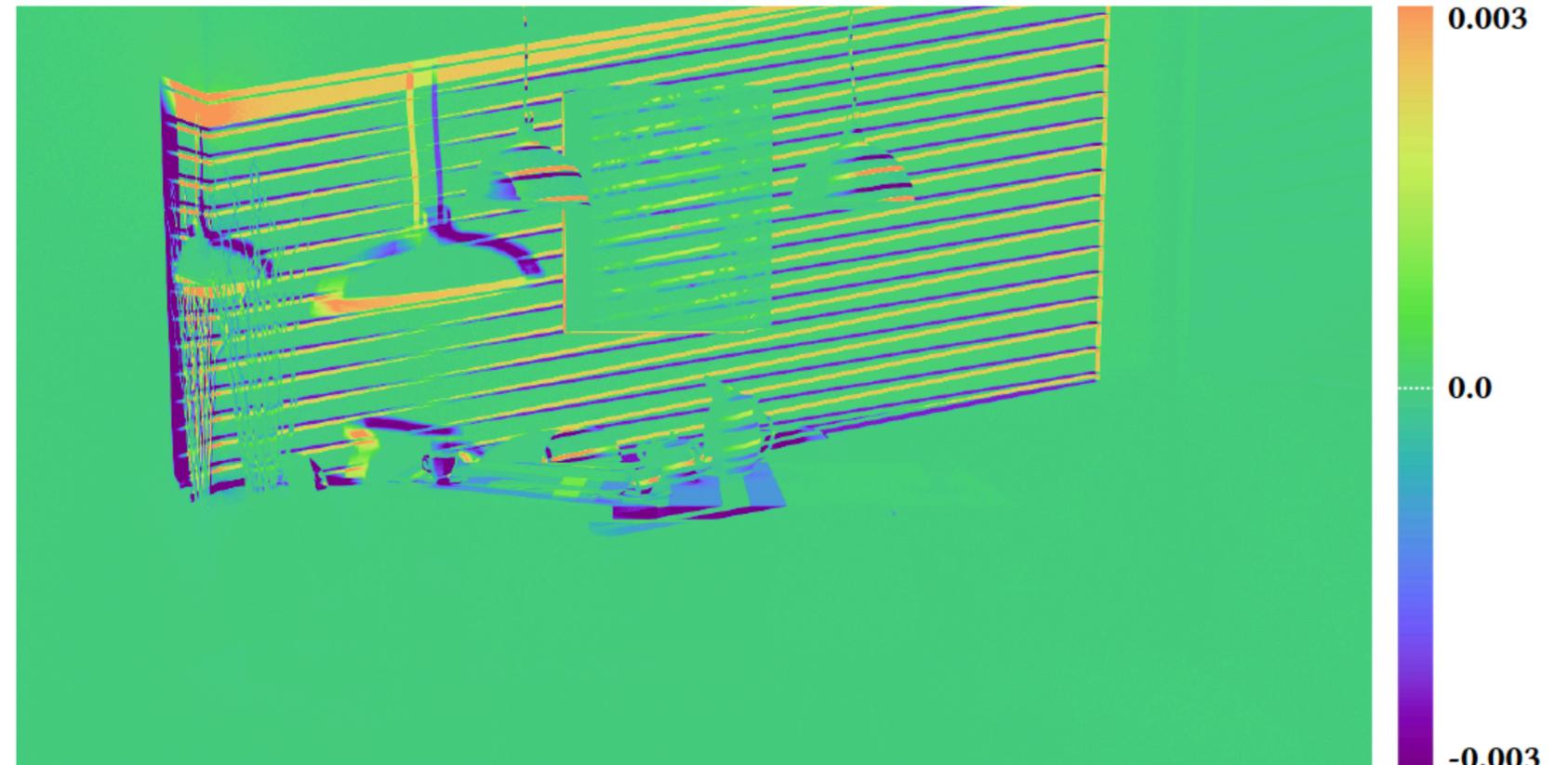
What is Differentiable Rendering?

- Computing **derivative** images (with respect to various parameters)



Original

Forward-rendering result



Derivative with respect to sun location

Differentiable-rendering result

Why Use Differentiable Rendering?

- Solving **inverse-rendering** problems
 - i.e., inferring **scene parameters** based on **images** of the scene
- Integrating **forward rendering** into **probabilistic inference** and **machine learning** pipelines
 - e.g., backpropagating losses during training
- Numerous applications in computer vision, computer graphics, computational imaging, VR/AR, ...

Forward and Inverse Rendering

Scene parameters



Geometry, materials, lighting, ...

Forward rendering

$I = \mathcal{R}(\theta)$

Inverse rendering

$\theta = \mathcal{R}^{-1}(I)?$

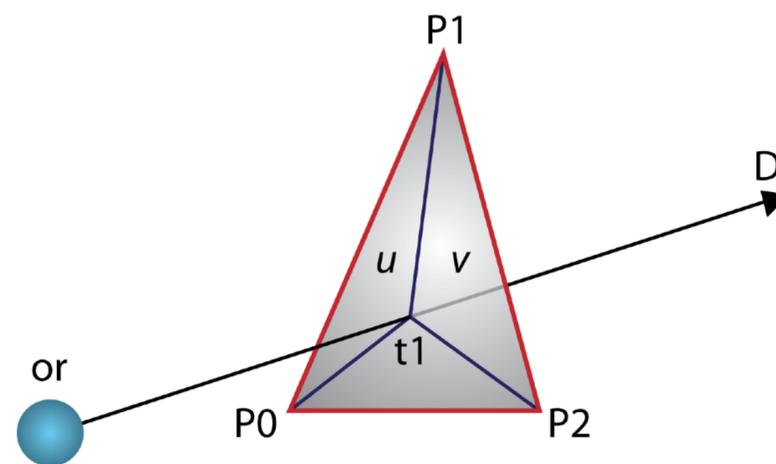
Rendered image



Scene: "bed classic" from Jiraniano

Ray Tracing

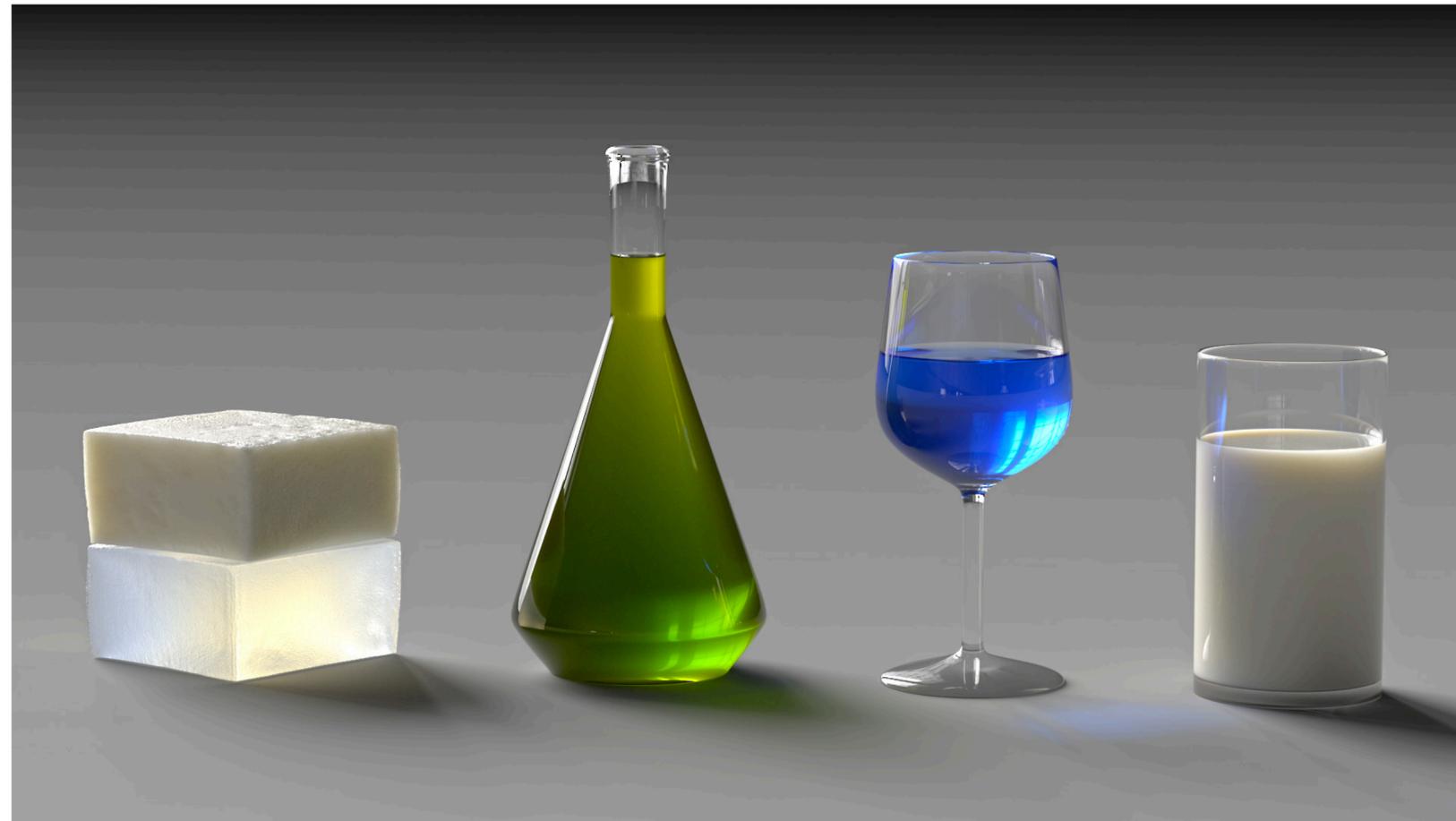
- A heavily abused term in graphics and vision
- We use **ray tracing** to mean **ray-surface intersection computations**
 - Applicable to both **explicit** (e.g., mesh) and **implicit** (e.g., SDF) surfaces



- *Basic building block* for most (if not all) physics-based rendering algorithms
 - e.g., path tracing, bidirectional path tracing, ...

Physics-Based Forward Rendering

- Relies heavily on Monte Carlo integration
- Can capture **complex** light-transport effects
 - Soft shadows, interreflection, subsurface scattering, ...



[Gkioulekas et al. 2013]

Physics-Based Inverse Rendering

Scene parameters



Geometry, materials, lighting, ...

Inverse rendering

$$\theta = \mathcal{R}^{-1}(I)?$$

- Inverting **physics-based** forward rendering
- Crucial to many applications

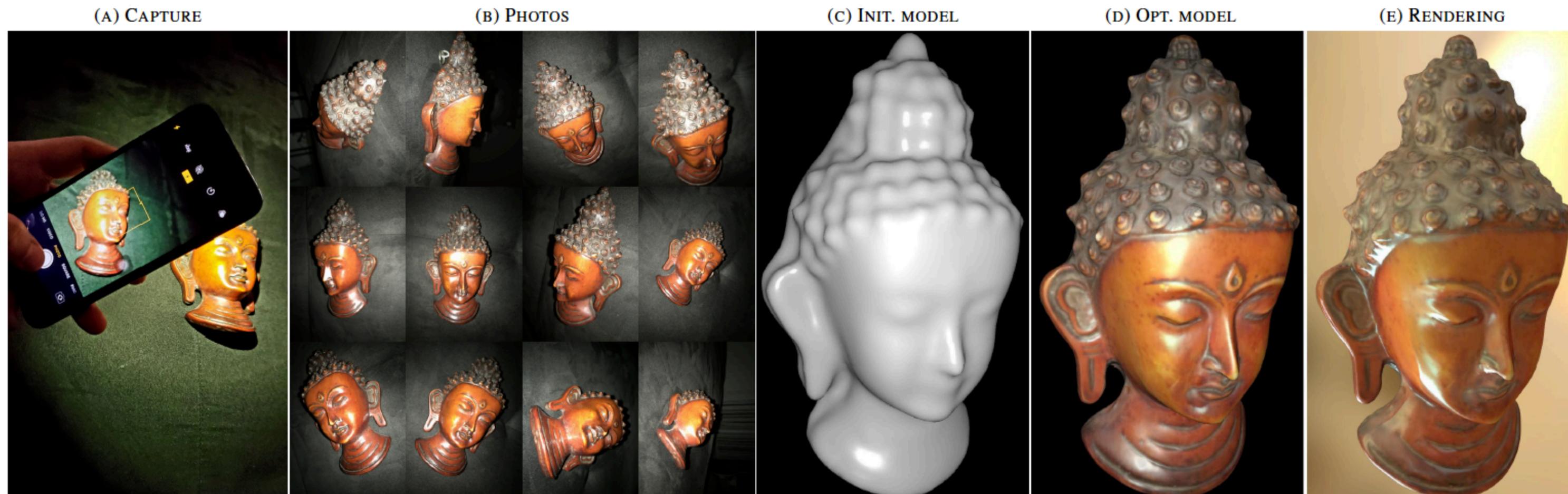
Rendered image



Scene: "bed classic" from Jiraniano

Shape and Material Reconstruction

Joint optimization of *object shape* and *spatially varying reflectance* (our recent work)



Computational Fabrication

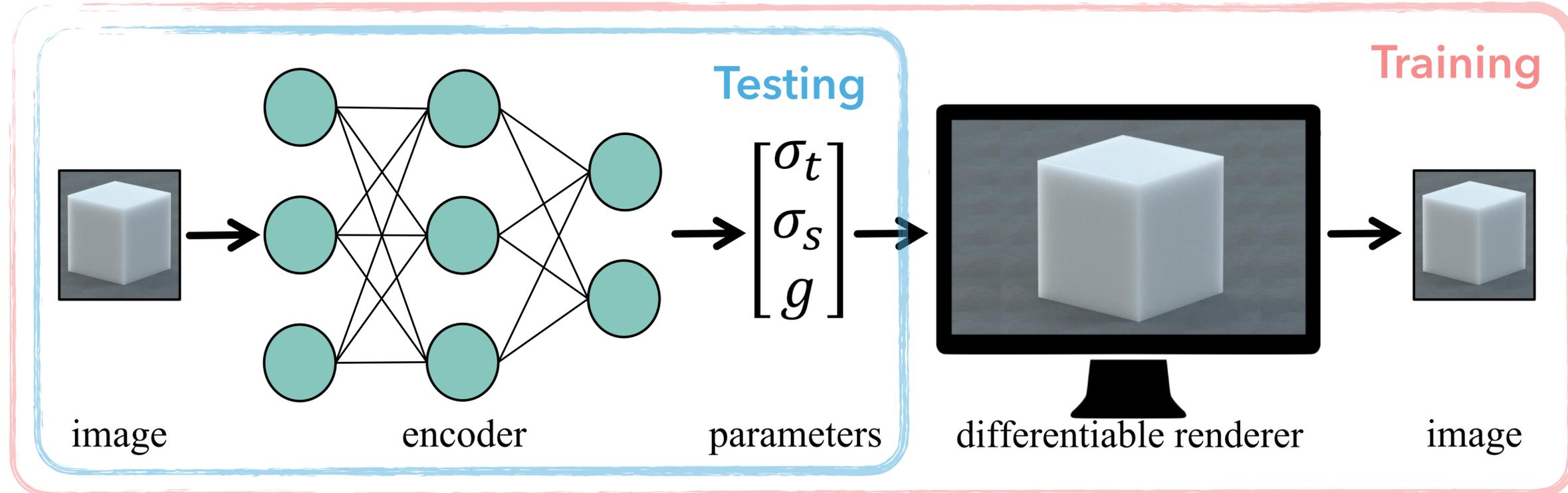
Determining the material configuration for individual voxels in full-color inkjet 3D printing



[Nindel et al. 2021]

Physics-Based Learning

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



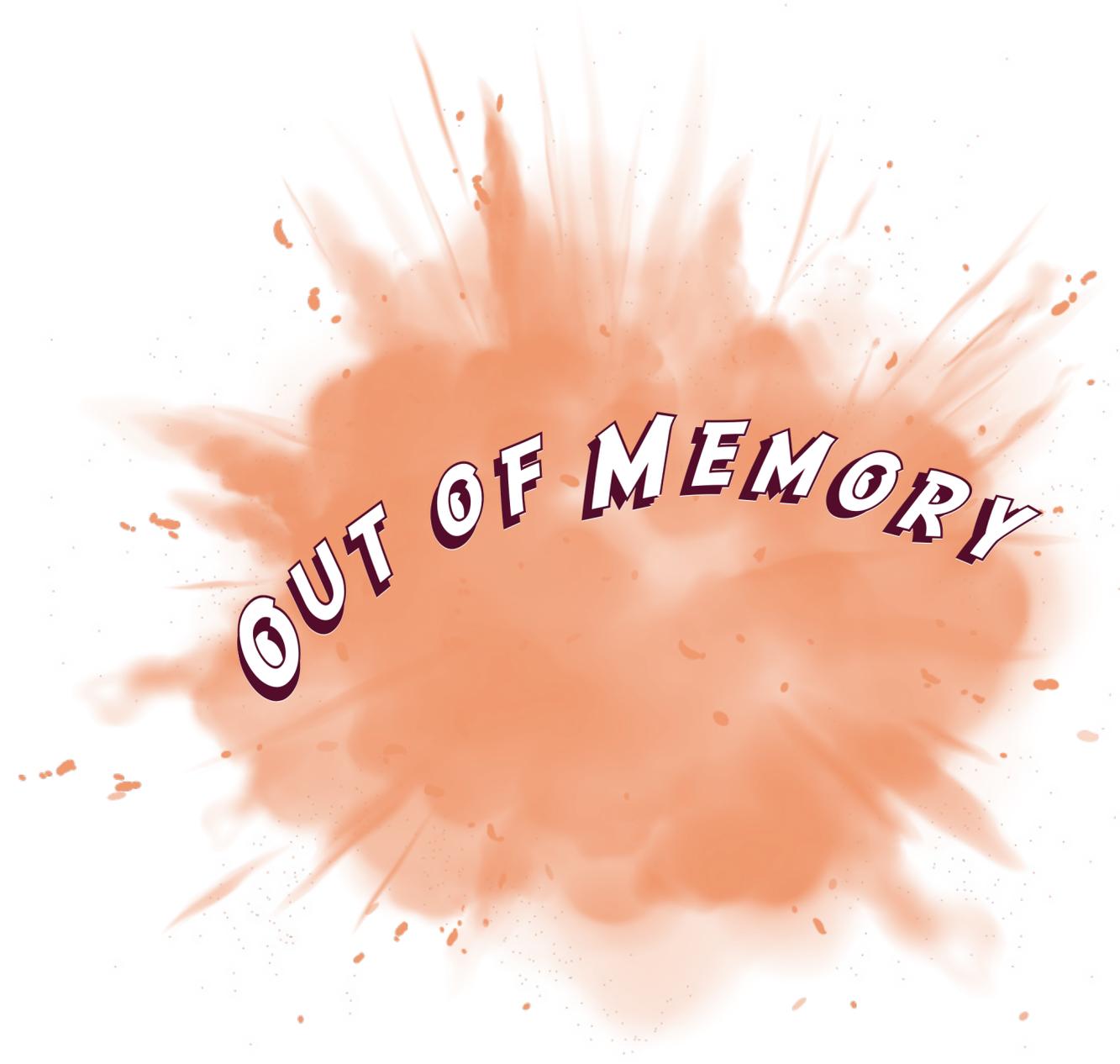
- Utilizing *image loss* provided by a volume path tracer to regularize training
- Use the trained encoder to solve inverse problems during testing

Why is Physics-Based Differentiable Rendering Hard?

- Need to differentiate solutions of integral equations (or path integrals)
 - e.g., the rendering equation: $L(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathbb{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_o) d\boldsymbol{\omega}_i + L_e(\mathbf{x}, \boldsymbol{\omega}_o)$
 - The relation between such solutions and scene parameters can be highly complex
- Requires handling very large gradient matrices (e.g., with 10^{12} or more entries)
- Can be tricky to implement correctly

Handling Many Parameters

- Forward-rendering function: $I = \mathcal{R}(\theta)$
 - $\theta \in \mathbb{R}^n$ (n : number of parameters)
 - $I \in \mathbb{R}^m$ (m : number of pixels)
- Gradient matrix: $\frac{d\mathcal{R}}{d\theta}(\mathbf{x}) \in \mathbb{R}^{m \times n}$
- Challenges:
 - m and n can both be large ($\sim 10^6$)
 - $(d\mathcal{R}/d\theta)$ can involve 10^{12} entries
 - Reverse-mode automatic differentiation can easily run out of memory



Precautions Must Be Taken

- Precautions must be taken to ensure **correctness**
 - E.g., applying automatic differentiation to a path tracer does not always work
- Should the PDF (used by a Monte Carlo estimator) be differentiated?
 - Can go either way...
(More on this later.)
- Discontinuities
 - Differentiating only the integrand is insufficient
(More on this later.)

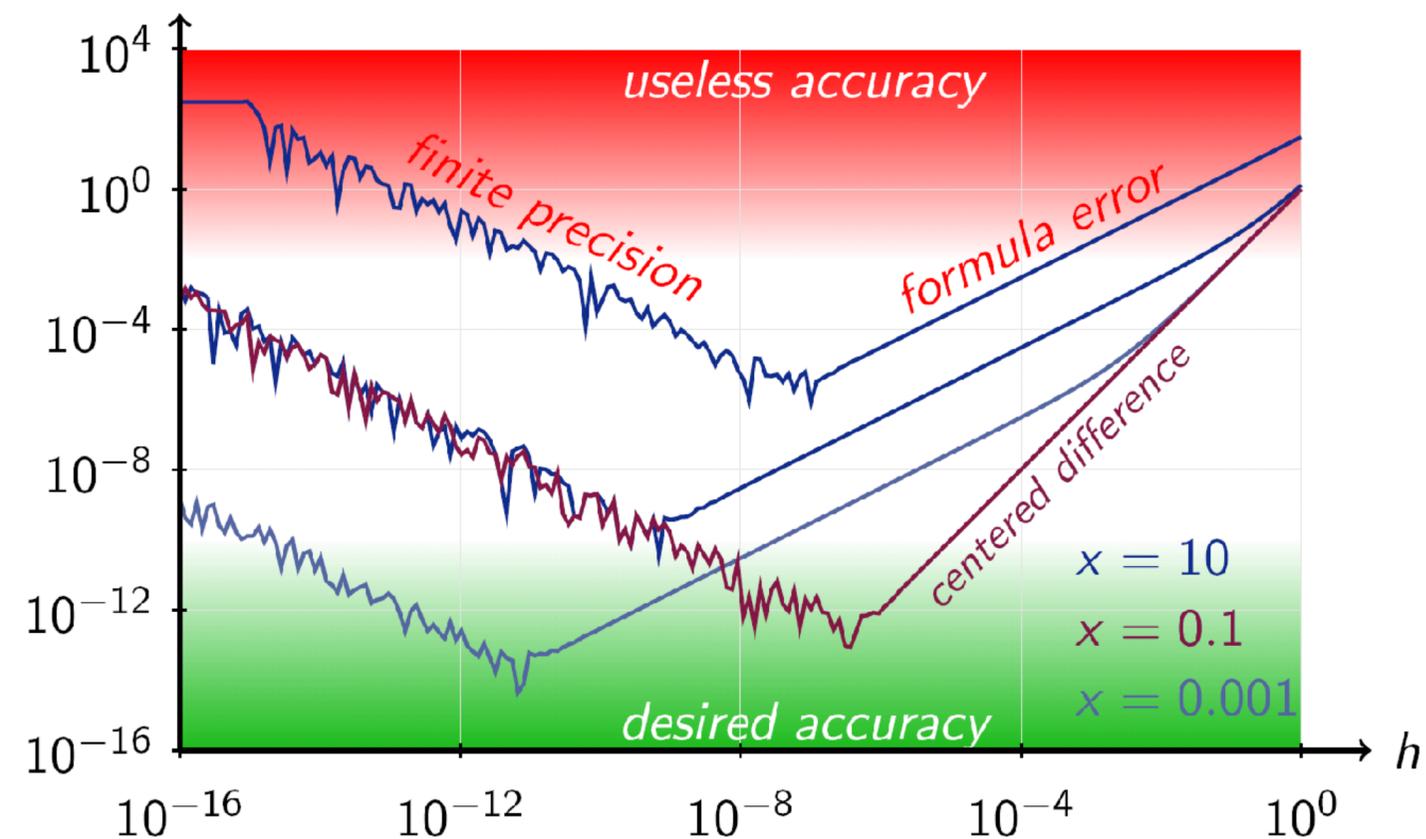
Why Not Simply Use Finite Differences?

Finite difference:

$$\frac{\partial \mathcal{R}}{\partial \theta_i}(\boldsymbol{\theta}) \approx \frac{\mathcal{R}(\boldsymbol{\theta} + \varepsilon \mathbf{e}_i) - \mathcal{R}(\boldsymbol{\theta} - \varepsilon \mathbf{e}_i)}{2\varepsilon}$$

Potential problems:

- High bias (large ε), rounding error (small ε)
- Need to correlate Monte Carlo samples
- Scales poorly with the number of parameters



[Wikipedia]

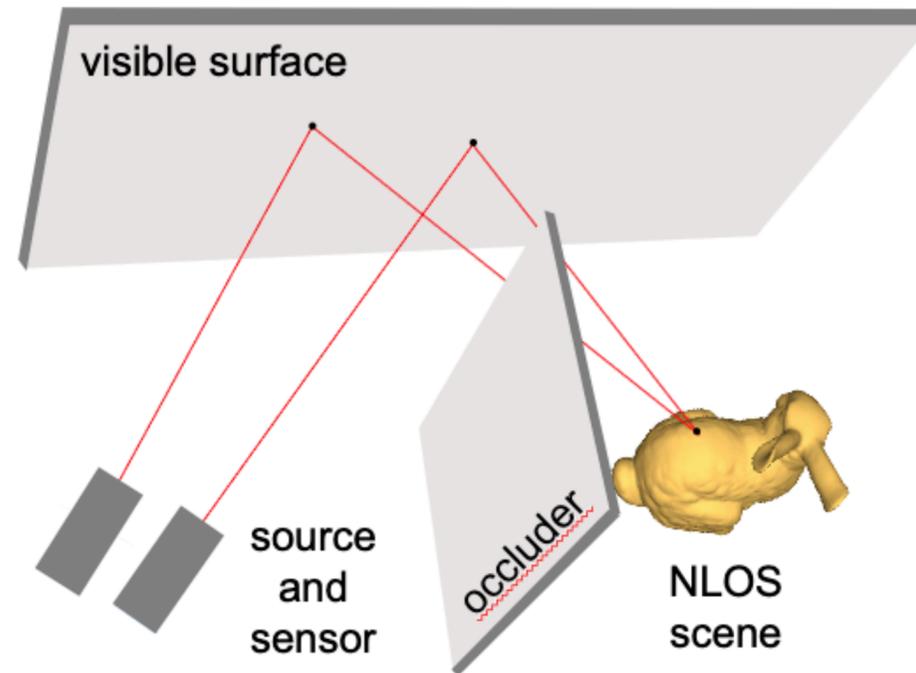
Global Illumination

- Can be simulated with modern differentiable renderers
- Required when solving many inverse-rendering problems



[theawesomer.com]

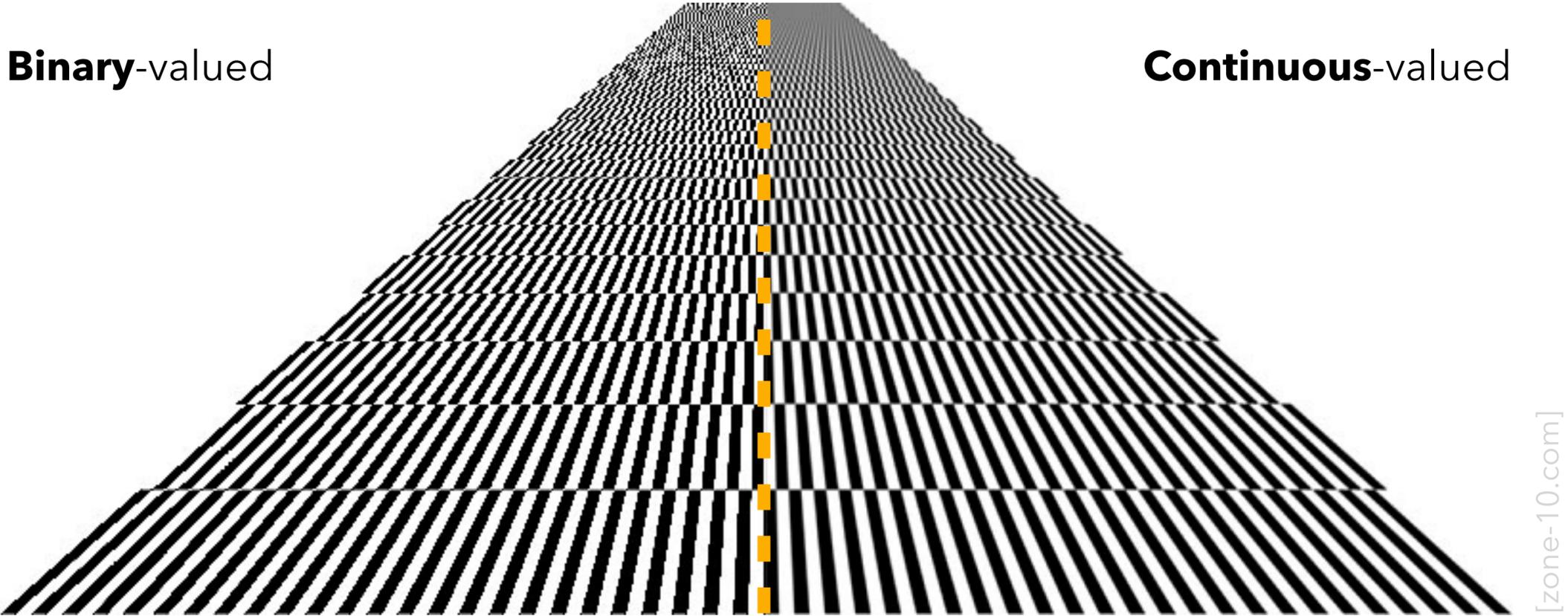
Computational fabrication



[Tsai et al. 2019]

Non-line-of-sight imaging

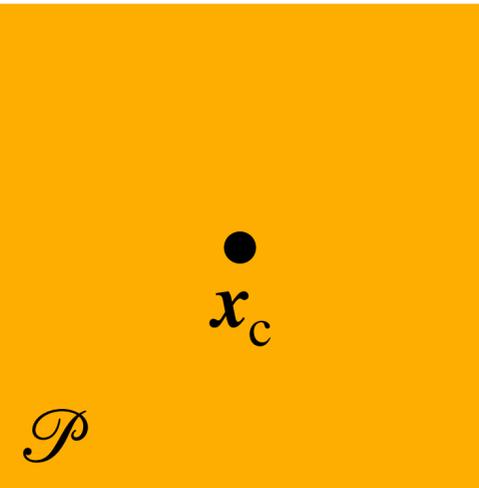
Pixel-Level Antialiasing Matters



[zone-10.com]

No antialiasing

Pixel value = $I(\mathbf{x}_c)$



One pixel

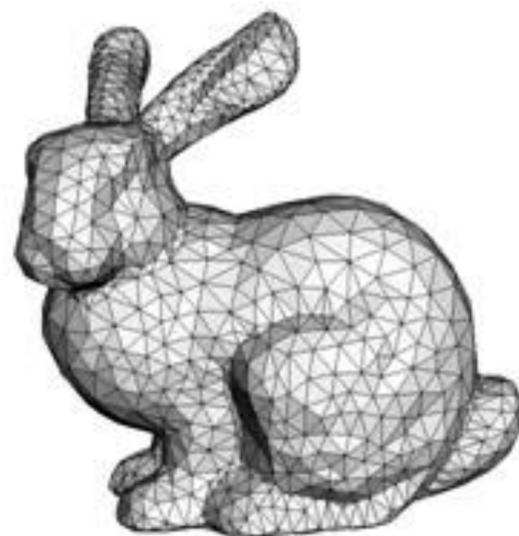
Perfect antialiasing

$$\text{Pixel value} = \frac{1}{|\mathcal{P}|} \int_{\mathcal{P}} I(\mathbf{x}) \, d\mathbf{x}$$

More information, more differentiable!

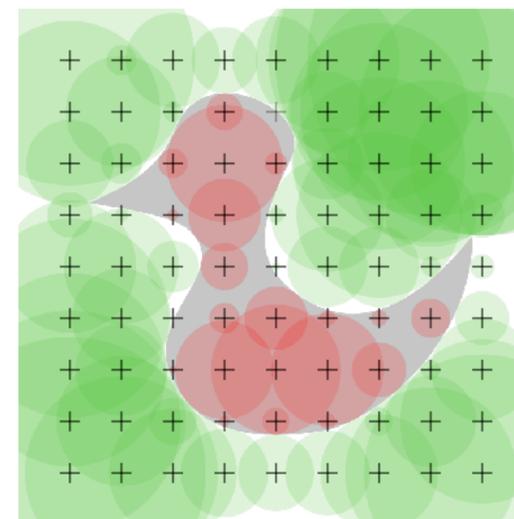
Can make inverse-rendering optimizations more robust

Geometric Representations



Explicit

(e.g., polygonal meshes)



Implicit

(e.g., signed distance functions)

- *Ray-tracing-based* **forward** rendering is agnostic to geometric representations
- The situation is more complex for **differentiable** rendering
 - Due to the need to handle discontinuities (will discuss in details later)

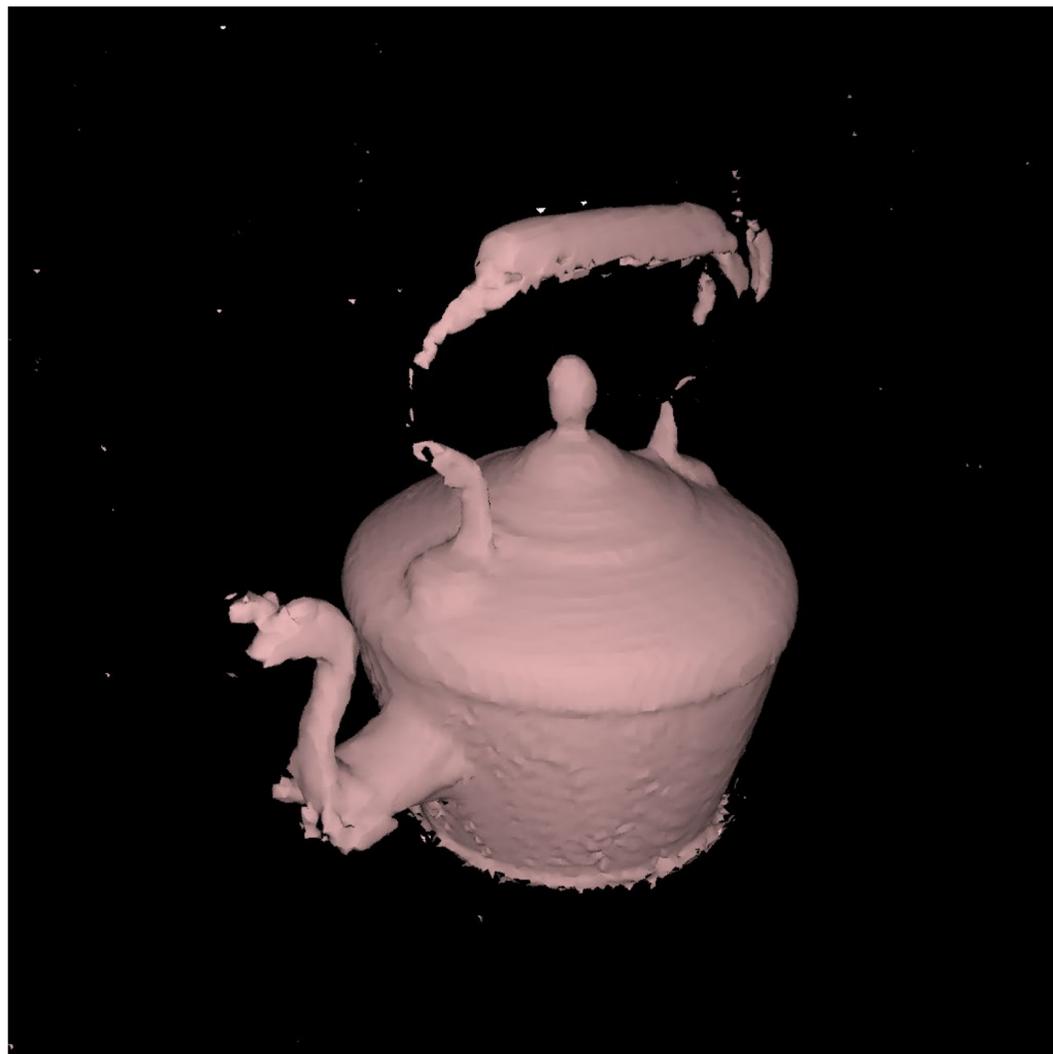
Why you should use ray-tracing-based differentiable rendering

Ray Tracing vs. Rasterization

- We believe that **ray tracing** is the way to go for future **differentiable** renderers
- Ray-tracing-based methods are **not** much slower than rasterization
 - Hardware-accelerated ray tracing has been improving rapidly (e.g., Nvidia RTX)
 - Visibility checks and intersections are typically not the performance bottleneck

Ray Tracing vs. Rasterization

23823 vertices, 44702 faces



Initial



Target

1024x1024 at 2 spp (Titan RTX)
differentiable render time:

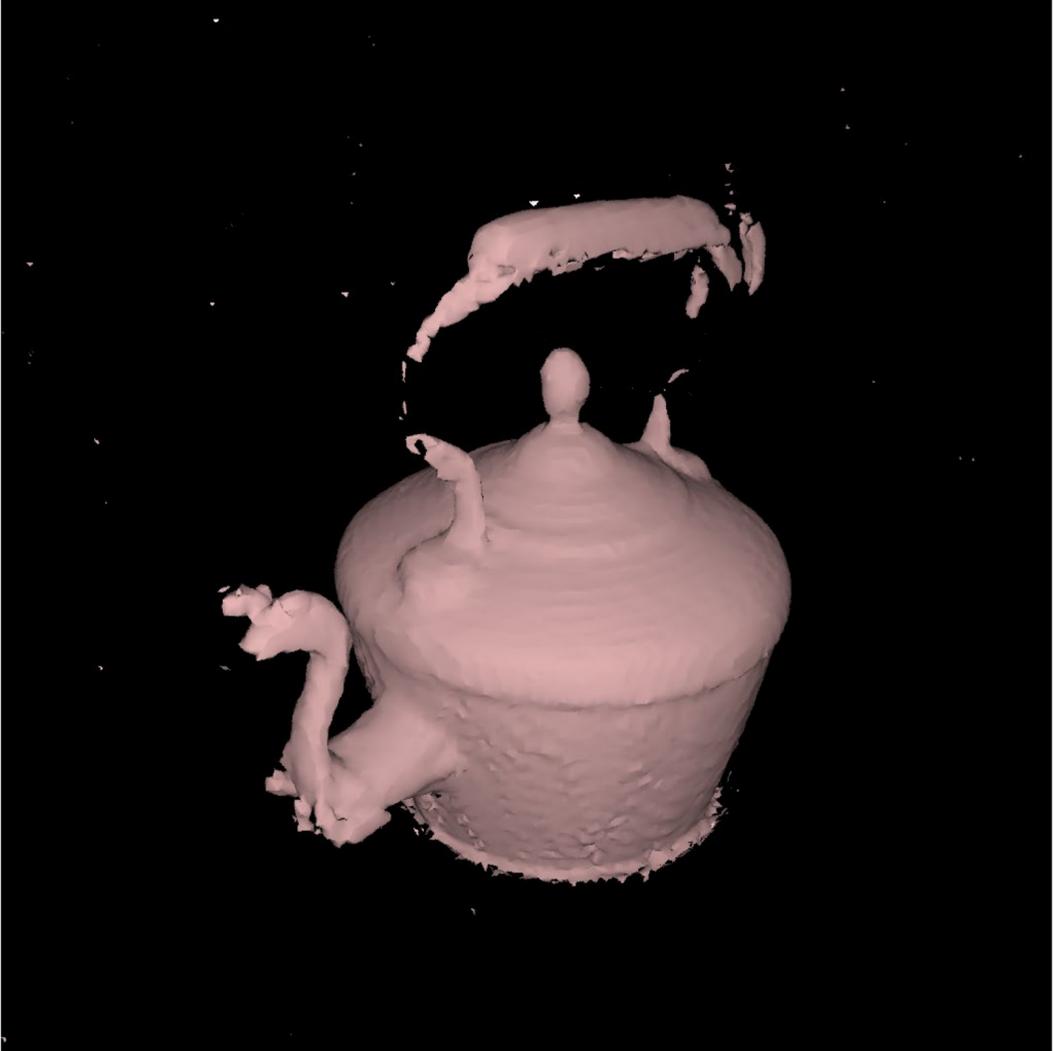
- **psdr-cuda** (ray-tracing-based)*: 2.8 msec
- **PyTorch3D** (soft rasterizer): 52.5 msec

Other computations (loss backpropagation, mesh evolution and remeshing):
~ 1000 msec

*Luan et al., EGSR 2021 (*to appear*)

Ray Tracing vs. Rasterization

23823 vertices, 44702 faces



Initial



Optimized (psdr-cuda)

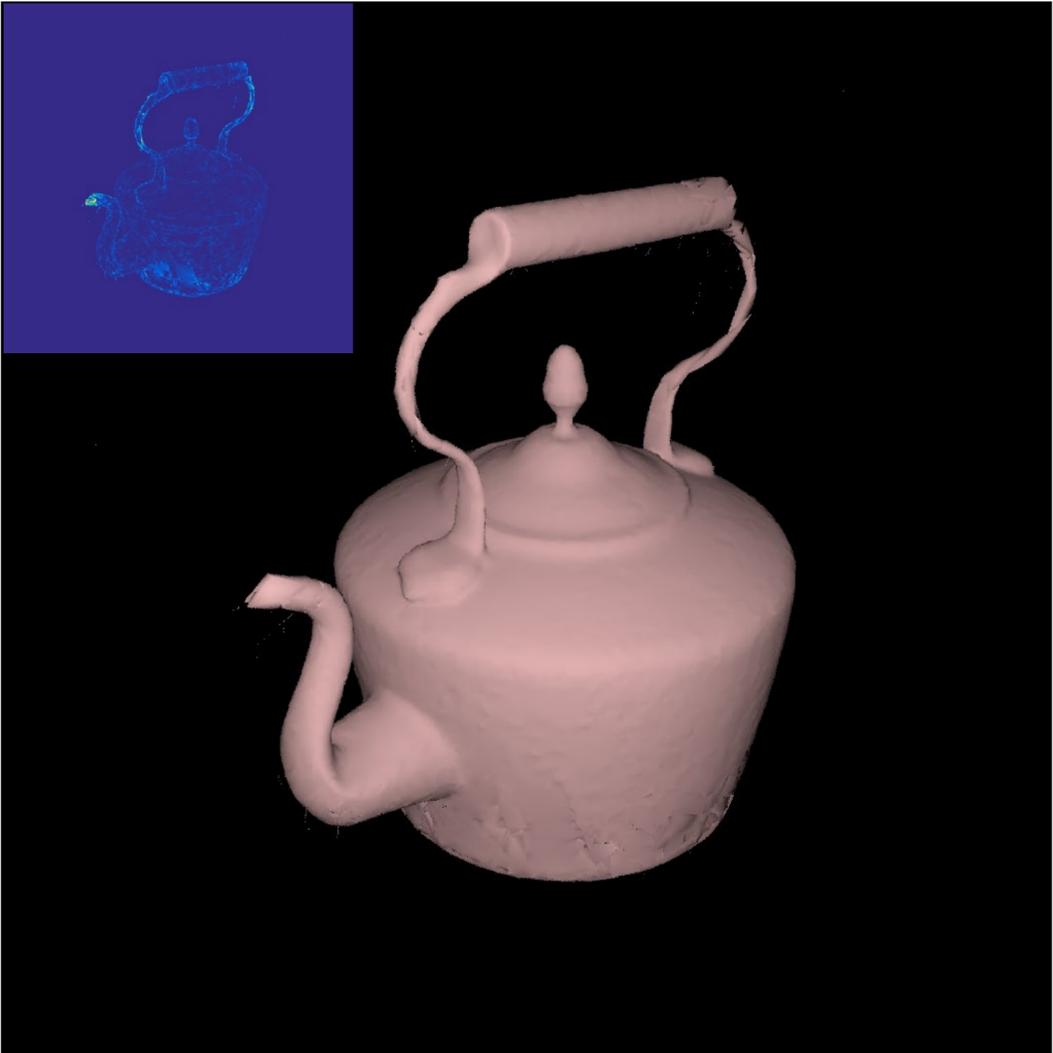
Absolute error

Ray Tracing vs. Rasterization

- We believe that **ray racing** is the way to go for future **differentiable** renderers
- Ray-tracing-based methods are **not** much slower than rasterization
 - Hardware-accelerated ray tracing has been improving rapidly (e.g., Nvidia RTX)
 - Visibility checks and intersections are typically not the performance bottleneck
- Ray-tracing-based methods can compute **correct** (i.e., unbiased) gradients
 - Correct gradients matter in optimization!

Ray Tracing vs. Rasterization

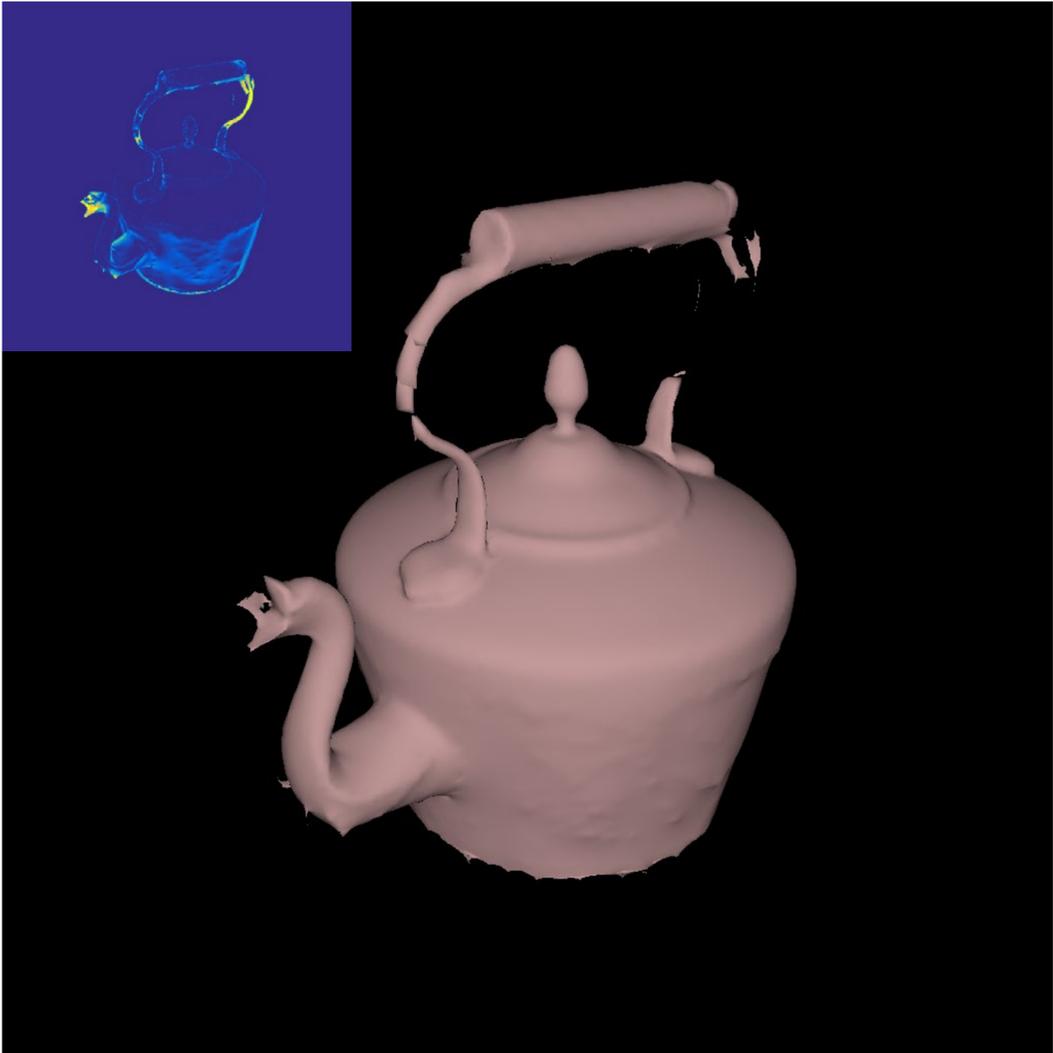
Optimization results after 5000 iterations (w/ identical settings)



Optimized (psdr-cuda)



Target



Optimized (PyTorch3D)

Ray Tracing vs. Rasterization

- We believe that **ray racing** is the way to go for future **differentiable** renderers

- Ray-tracing-based methods are **not** much slower than rasterization

- Hardware-accelerated ray tracing has been improving rapidly (e.g., Nvidia RTX)
- Visibility checks and intersections are typically not the performance bottleneck

Second part of
this tutorial

- Ray-tracing-based methods can compute **correct** (i.e., unbiased) gradients

- Correct gradients matter in optimization!

- Ray-tracing-based methods can handle **complex** light-transport effects

- Soft shadows, environmental illumination
- Inter-reflections, radiative transfer (e.g., subsurface scattering), caustics

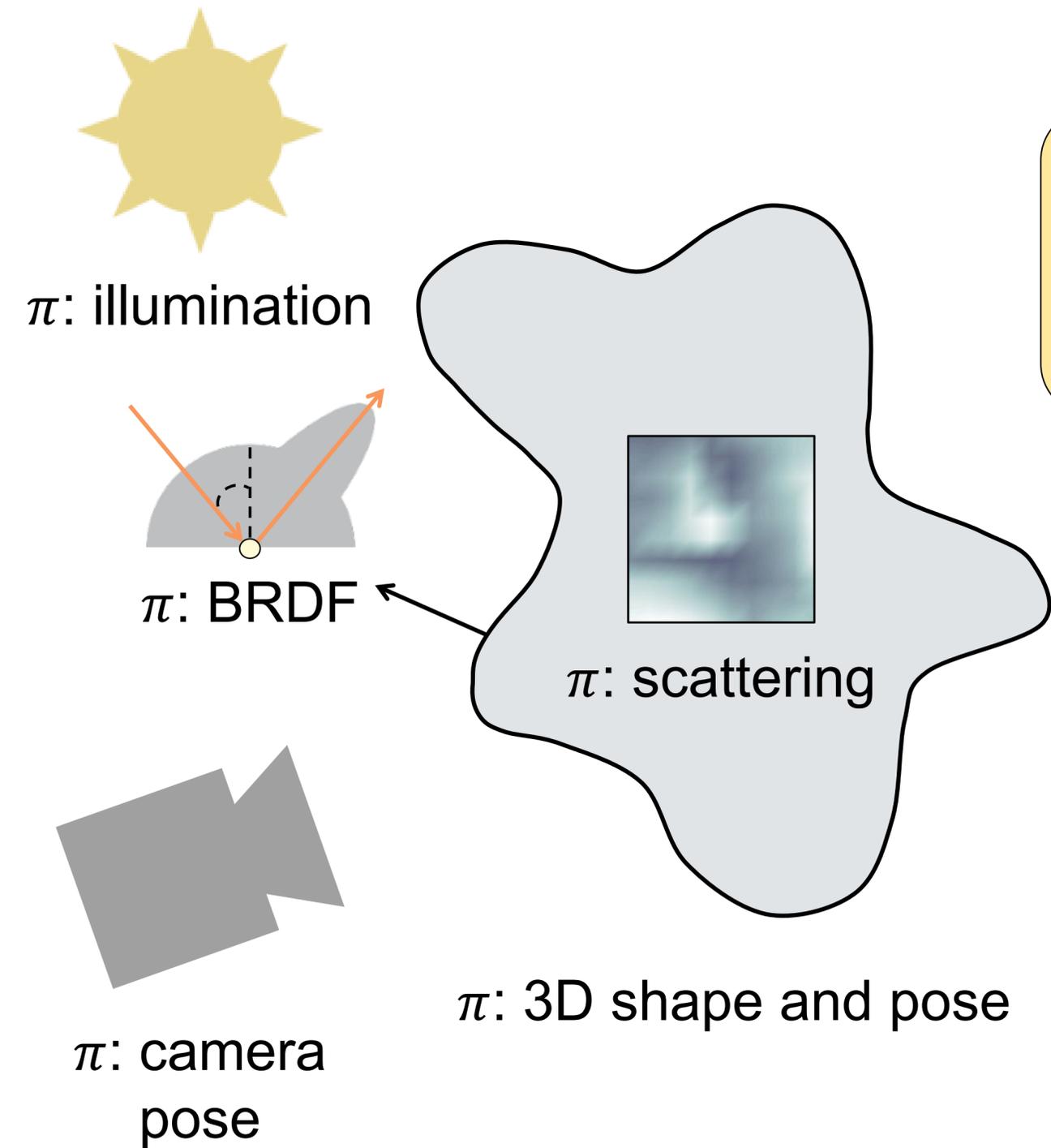
Third part of
this tutorial

- Ray-tracing-based methods can provide gradients in **general** scenes

- Different shape representations, including point clouds, explicit (e.g., meshes), implicit (e.g., neural SDFs)
- Different types of cameras (e.g., intensity, lightfield, polarization, time-of-flight, hyperspectral, ...)

What differentiable rendering does not give us

Inverse rendering (a.k.a. analysis by synthesis)



Analysis-by-synthesis optimization:

$$\min_{\text{scene unknowns } \pi} \text{loss} \left[\text{image of pumpkin}, \text{render} \left(\begin{array}{c} \text{scene} \\ \text{unknowns } \pi \end{array} \right) \right]$$

Stochastic gradient descent (e.g., Adam):

$$\text{initialize } \pi \leftarrow \pi_0$$

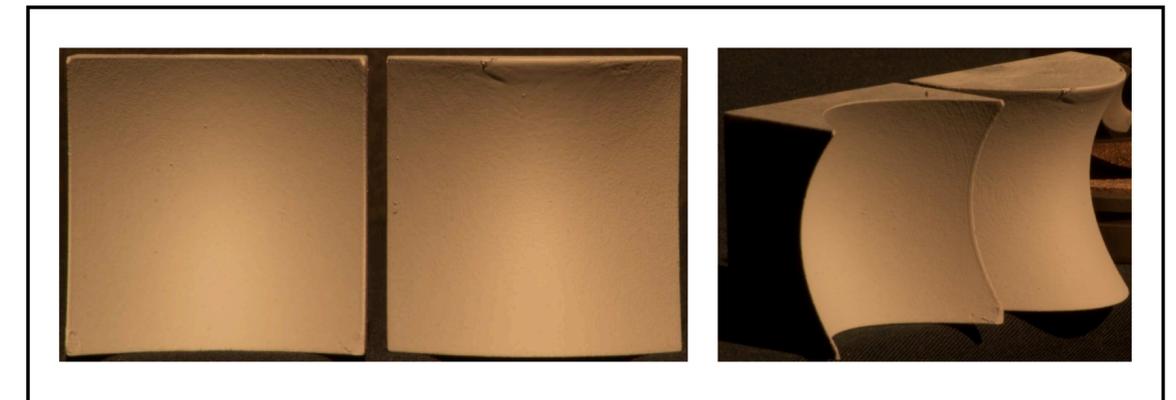
while (not converged)

$$\text{update } \pi \leftarrow \pi + \eta \cdot \frac{d\text{loss}(\pi)}{d\pi}$$

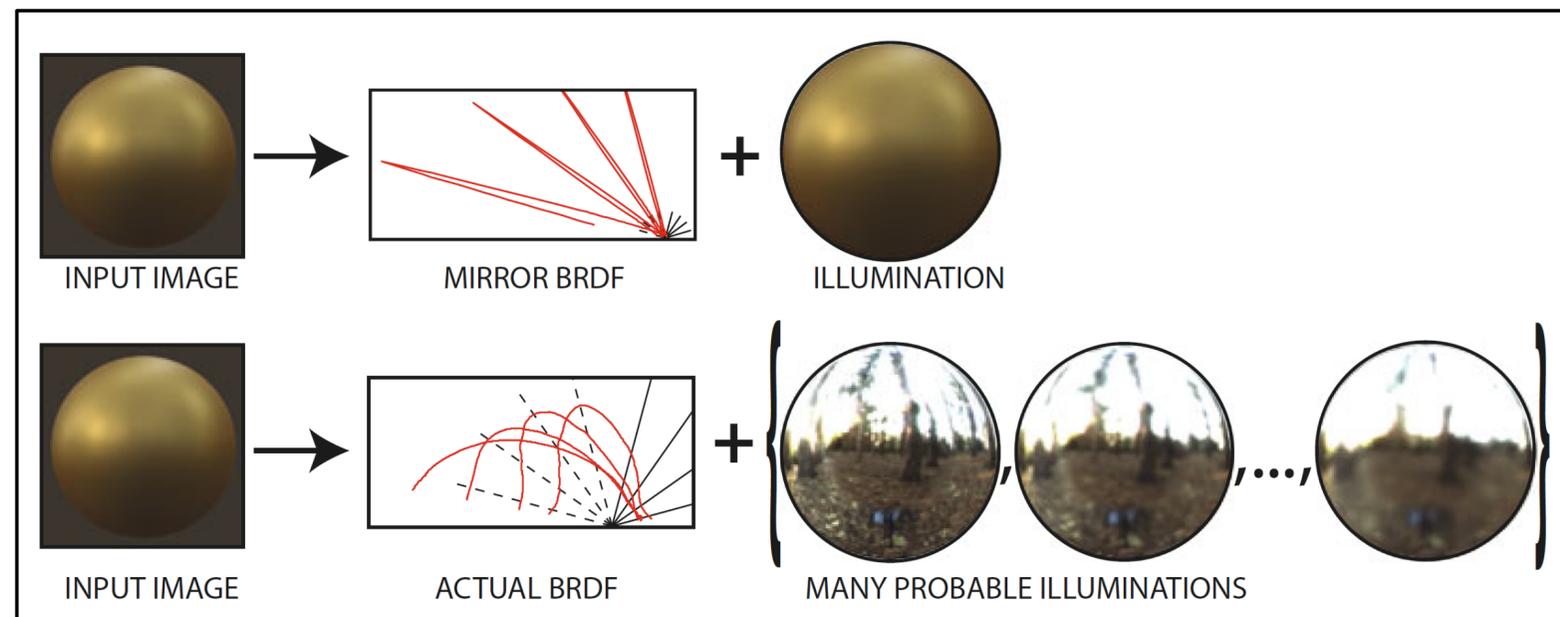
Differentiable rendering

Why we need good initializations

- Analysis-by-synthesis objectives are highly non-convex, non-linear
 - Multiple *local* minima
- Ambiguities exist between different parameters
 - Multiple *global* minima



Ambiguities between shape and lighting
[Xiong et al. 2015]



Ambiguities between BRDF and lighting
[Romeiro and Zickler 2010]

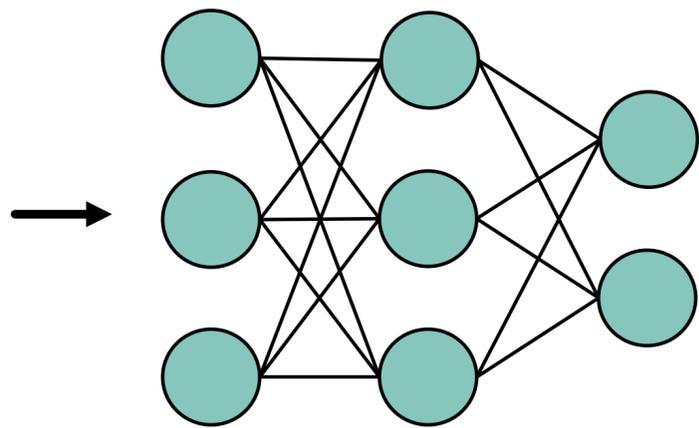


Ambiguities between scattering parameters
[Zhao et al. 2014]

Inverse rendering (a.k.a. analysis by synthesis)

Learned initializations help:

- avoid local minima
- accelerate convergence



Neural network

Analysis-by-synthesis optimization:

$$\min_{\text{scene unknowns } \pi} \text{loss} \left[\text{render} \left(\begin{array}{c} \text{scene} \\ \text{unknowns } \pi \end{array} \right) \right]$$

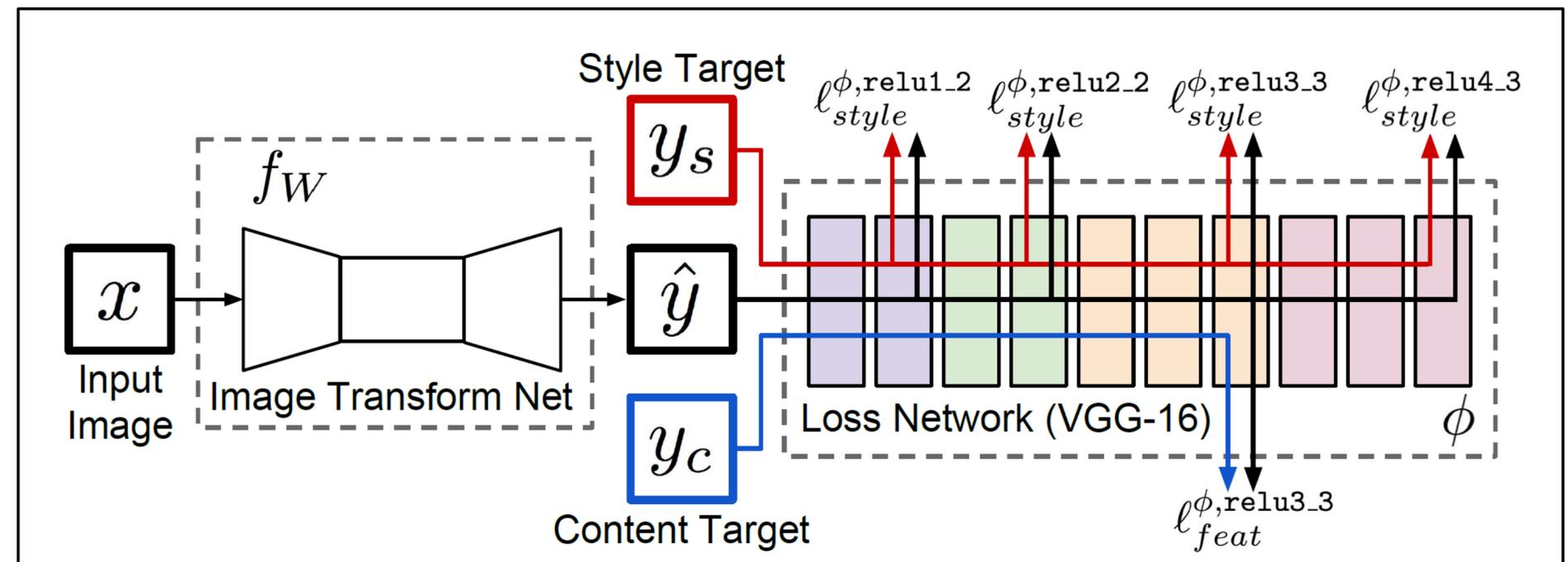
Stochastic gradient descent (e.g., Adam):

$$\begin{aligned} &\text{initialize } \pi \leftarrow \pi_0 \\ &\text{while (not converged)} \\ &\quad \text{update } \pi \leftarrow \pi + \eta \cdot \frac{d\text{loss}(\pi)}{d\pi} \end{aligned}$$

Differentiable rendering

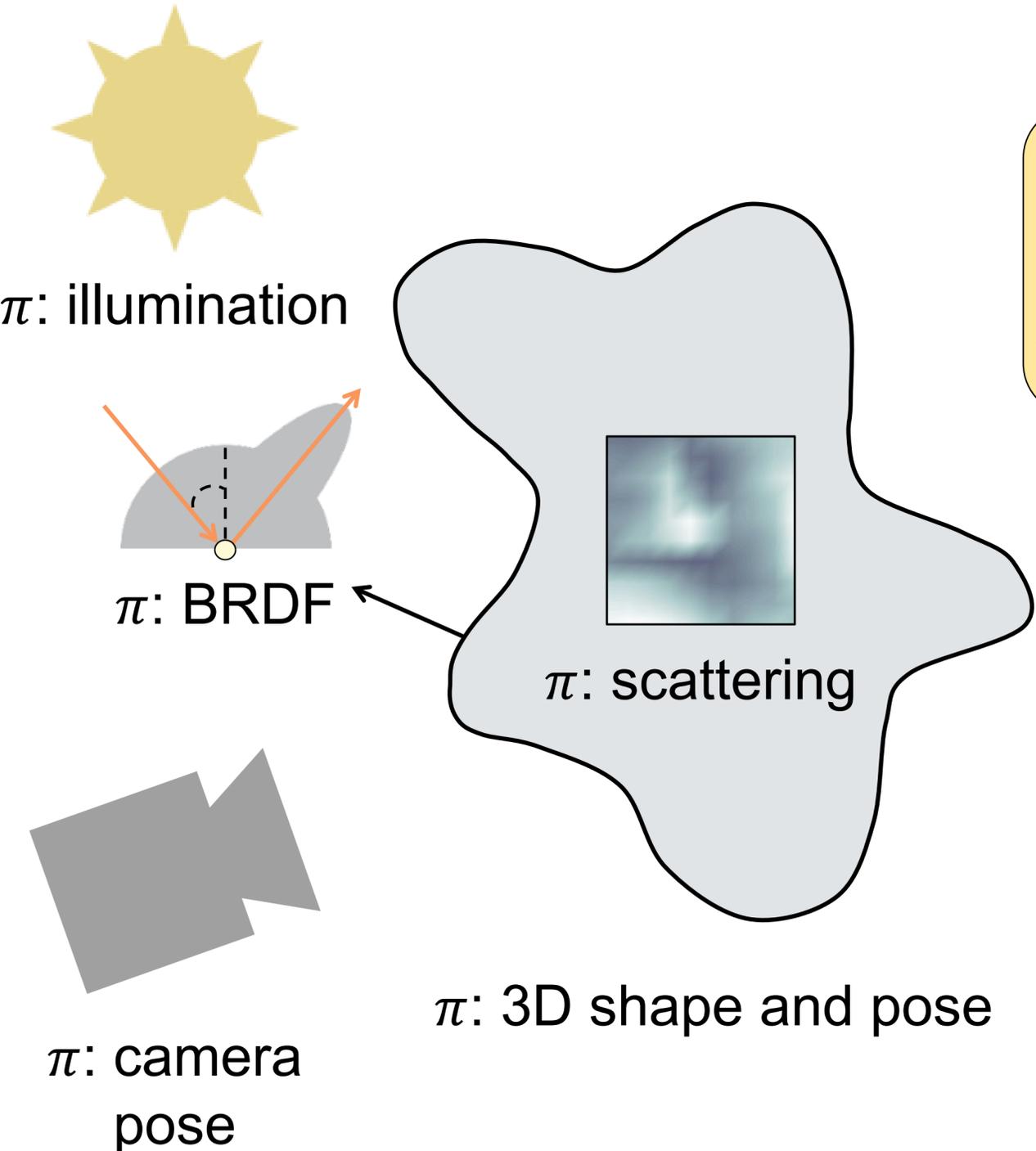
Why we need discriminative loss functions

- Well-designed loss functions can help reduce ambiguities
- Perceptual losses can help emphasize design aspects that matter
- Differentiable rendering can be combined with any loss function that can be backpropagated through



VGG-based *perceptual loss* [Johnson et al. 2016]

Inverse rendering (a.k.a. analysis by synthesis)



Analysis-by-synthesis optimization:

$$\min_{\text{scene unknowns } \pi} \text{loss} \left[\text{img}, \text{render} \left(\begin{array}{c} \text{scene} \\ \text{unknowns } \pi \end{array} \right) \right]$$

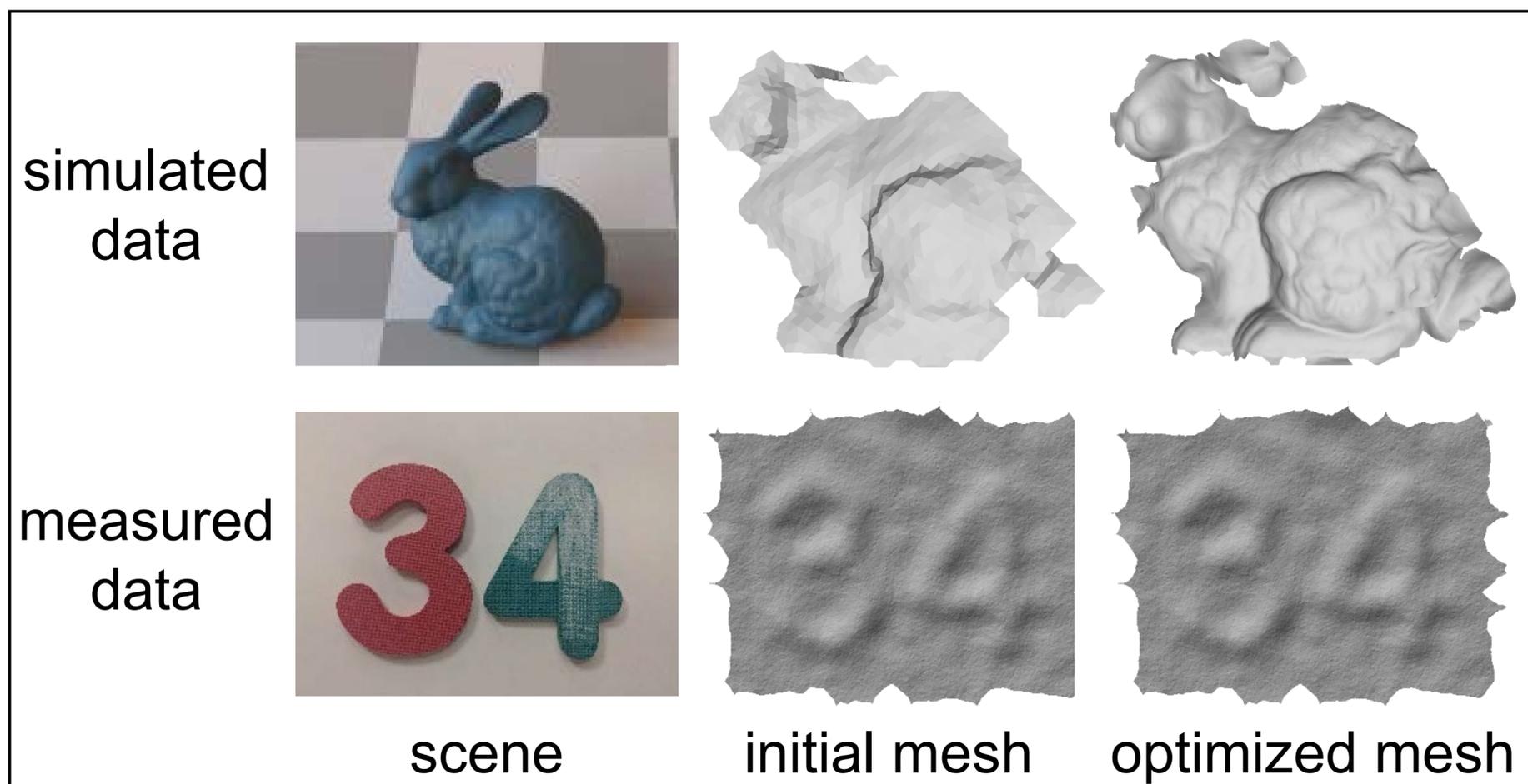
Stochastic gradient descent (e.g., Adam):

```
initialize  $\pi \leftarrow \pi_0$ 
while (not converged)
  update  $\pi \leftarrow \pi + \eta \cdot \frac{d\text{loss}(\pi)}{d\pi}$ 
```

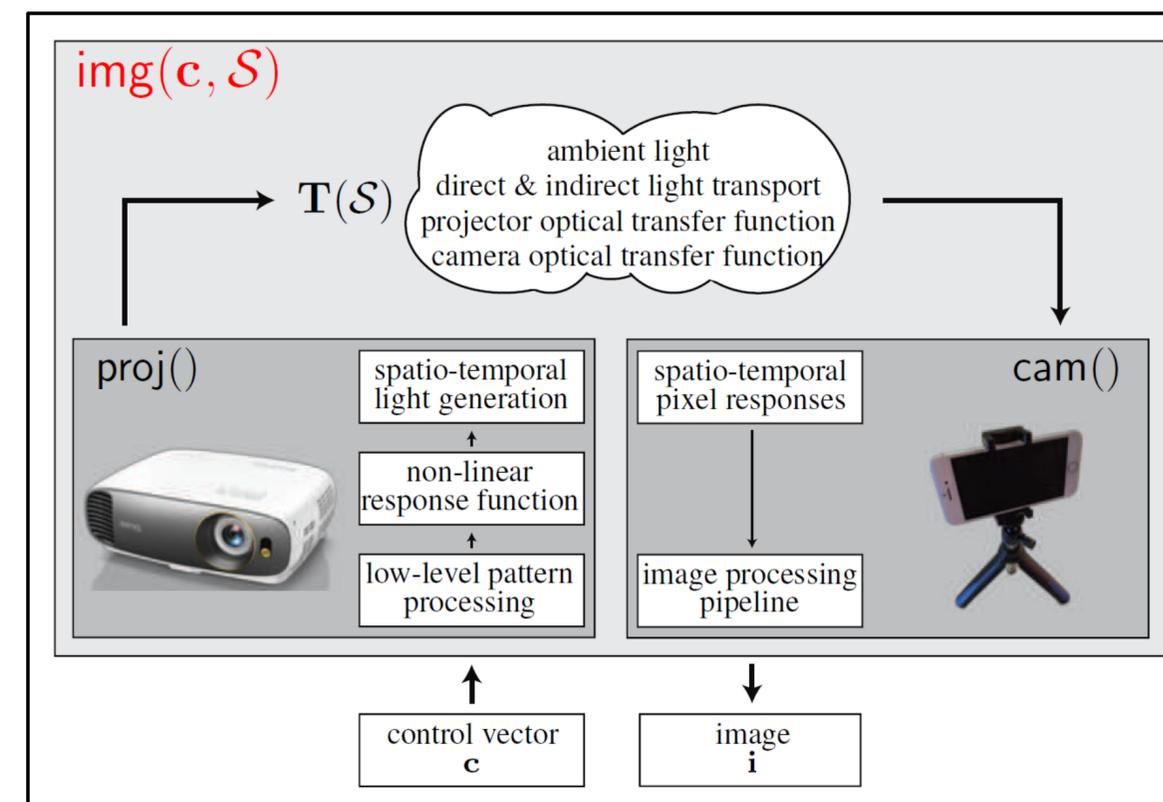
Differentiable rendering

High signal-to-noise ratio is critical

- The extent to which we can improve upon an initialization strongly depends on the signal-to-noise ratio of our measurements
- We need reliable camera models (noise, aberrations, other non-idealities)



Non-line-of-sight imaging [Tsai et al. 2019]



Optical gradient descent [Chen et al. 2020]

Differential Direct Illumination

Reminder from calculus

Differentiation under the integral sign

Also known as the Leibniz integral rule

$$\frac{d}{d\pi} \int_{a(\pi)}^{b(\pi)} f(x, \pi) dx \stackrel{?}{=} \int_{a(\pi)}^{b(\pi)} \frac{d}{d\pi} f(x, \pi) dx$$

Move derivative
inside integral

Account for changes in
integration limits

$$+ f(b(\pi), \pi) \frac{db(\pi)}{d\pi} - f(a(\pi), \pi) \frac{da(\pi)}{d\pi}$$

Account for discontinuities of
integrand that depend on π

$$+ \sum_i (f(c_i(\pi)^-, \pi) - f(c_i(\pi)^+, \pi)) \frac{dc_i(\pi)}{d\pi}$$

A simple example

$$f(x, \pi) = \begin{cases} 0 & \text{if } x < 2\pi \\ 1 & \text{if } x \geq 2\pi \end{cases}$$

$$\frac{d}{d\pi} \int_0^{4\pi} f(x, \pi) dx = \int_0^{2\pi} \frac{d}{d\pi} 0 dx + \int_{\pi}^{4\pi} \frac{d}{d\pi} 1 dx \quad \text{Move derivative inside integral}$$

Account for changes in integration limits

$$+ 1 \frac{d(4\pi)}{d\pi} - 0 \frac{d0}{d\pi}$$

Account for discontinuities of integrand that depend on π

$$+ (0 - 1) \frac{d(2\pi)}{d\pi}$$

Leibniz integral rule

Differentiation under the integral sign

Also known as the Leibniz integral rule

$$\frac{d}{d\pi} \int_{a(\pi)}^{b(\pi)} f(x, \pi) dx$$

Interior integral

$$\int_{a(\pi)}^{b(\pi)} \frac{d}{d\pi} f(x, \pi) dx$$

Move derivative
inside integral

Account for changes in
integration limits

Boundary terms

$$+ f(b(\pi), \pi) \frac{db(\pi)}{d\pi} - f(a(\pi), \pi) \frac{da(\pi)}{d\pi}$$

Account for discontinuities of
integrand that depend on π

$$+ \sum_i (f(c_i(\pi)^-, \pi) - f(c_i(\pi)^+, \pi)) \frac{dc_i(\pi)}{d\pi}$$

Simplified Leibniz integral rule

Differentiation under the integral sign

Also known as the Leibniz integral rule

$$\frac{d}{d\pi} \int_a^b f(x, \pi) dx = \int_a^b \frac{d}{d\pi} f(x, \pi) dx$$

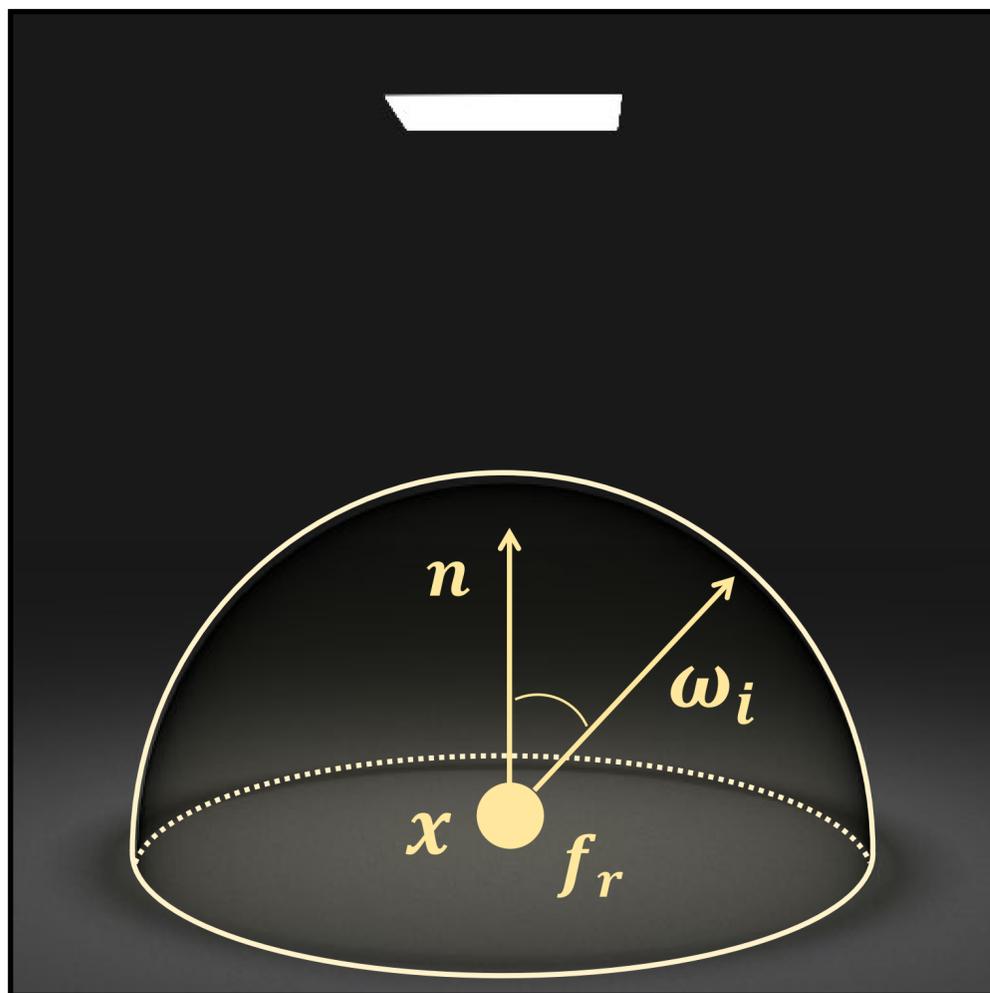
Interior integral

Move derivative
inside integral

Differentiation wrt π simplifies to just moving derivative inside integral when:

- Integration limits are independent of π .
- Integrand discontinuities are independent of π .

Direct illumination integral



Radiance from x :

$$I = \int_{\mathbb{H}^2} \overset{\text{Reflectance (BRDF)}}{f_r(\omega_i, \omega_o)} \overset{\text{Incident radiance}}{L_i(\omega_i)} \overset{\text{Shading wrt normal } \mathbf{n}}{(\mathbf{n} \cdot \omega_i)} d\sigma(\omega_i)$$

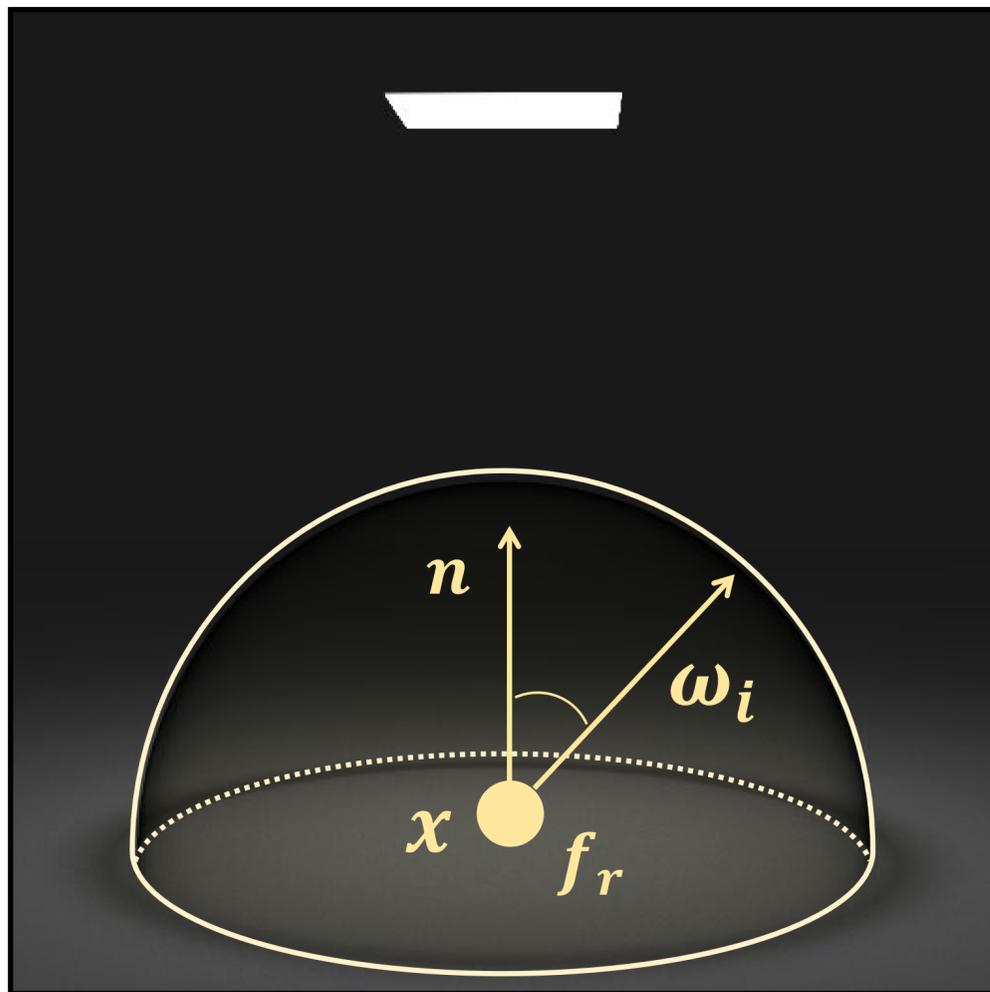
Unit hemisphere

Monte Carlo rendering:

- Sample random directions ω_i^s from PDF $p(\omega_i)$
- Form estimator

$$I \approx \sum_s \frac{f_r(\omega_i^s, \omega_o) L_i(\omega_i^s) (\mathbf{n} \cdot \omega_i^s)}{p(\omega_i^s)}$$

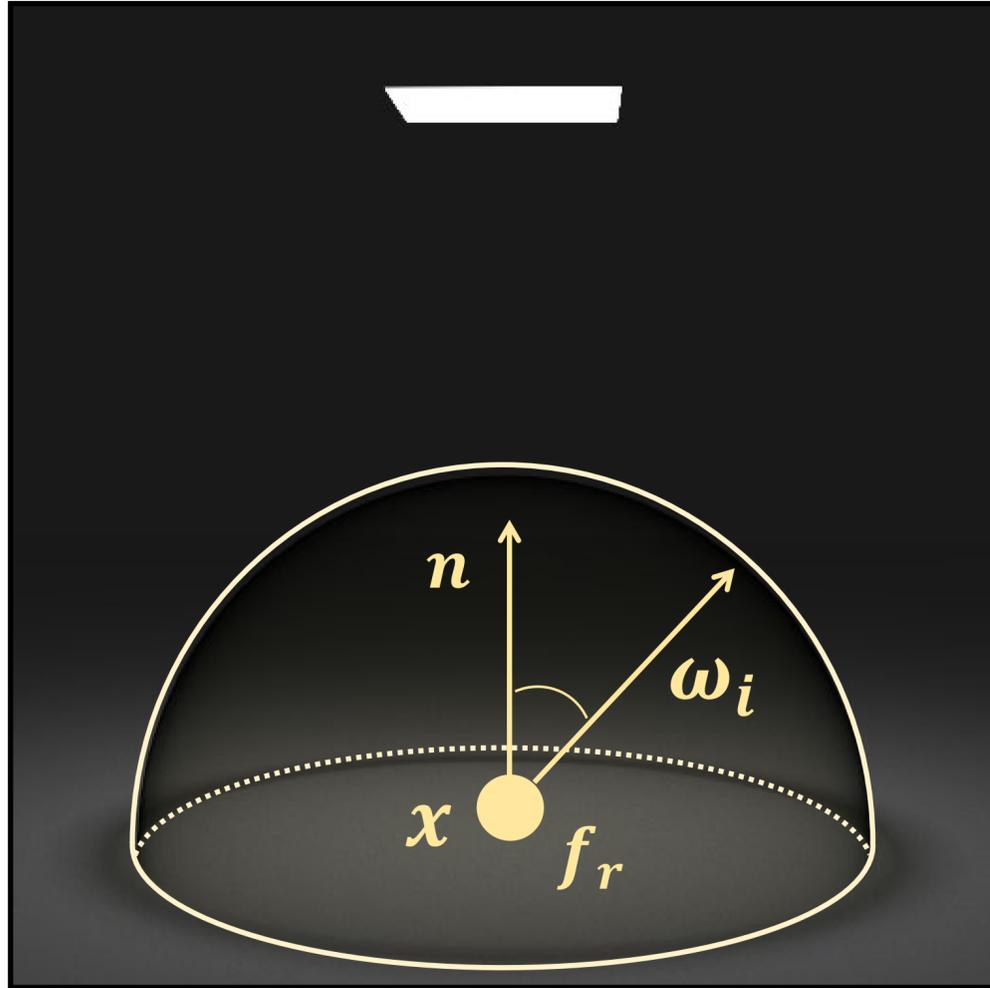
Differential direct illumination



Differential radiance from x :

$$\frac{dI}{d\pi} = \frac{d}{d\pi} \int_{\mathbb{H}^2} f_r(\omega_i, \omega_o) L_i(\omega_i) (n \cdot \omega_i) d\sigma(\omega_i)$$

Differential direct illumination: local parameters



π : local parameters

- BRDF parameters
- shading normal
- illumination brightness

Differential radiance from x :

$$\frac{dI}{d\pi} = \int_{\mathbb{H}^2} \frac{d}{d\pi} \{f_r(\omega_i, \omega_o) L_i(\omega_i) (n \cdot \omega_i)\} d\sigma(\omega_i)$$

Just move derivative inside integral

Monte Carlo differentiable rendering:

- Sample random directions ω_i^s from PDF $p(\omega_i)$

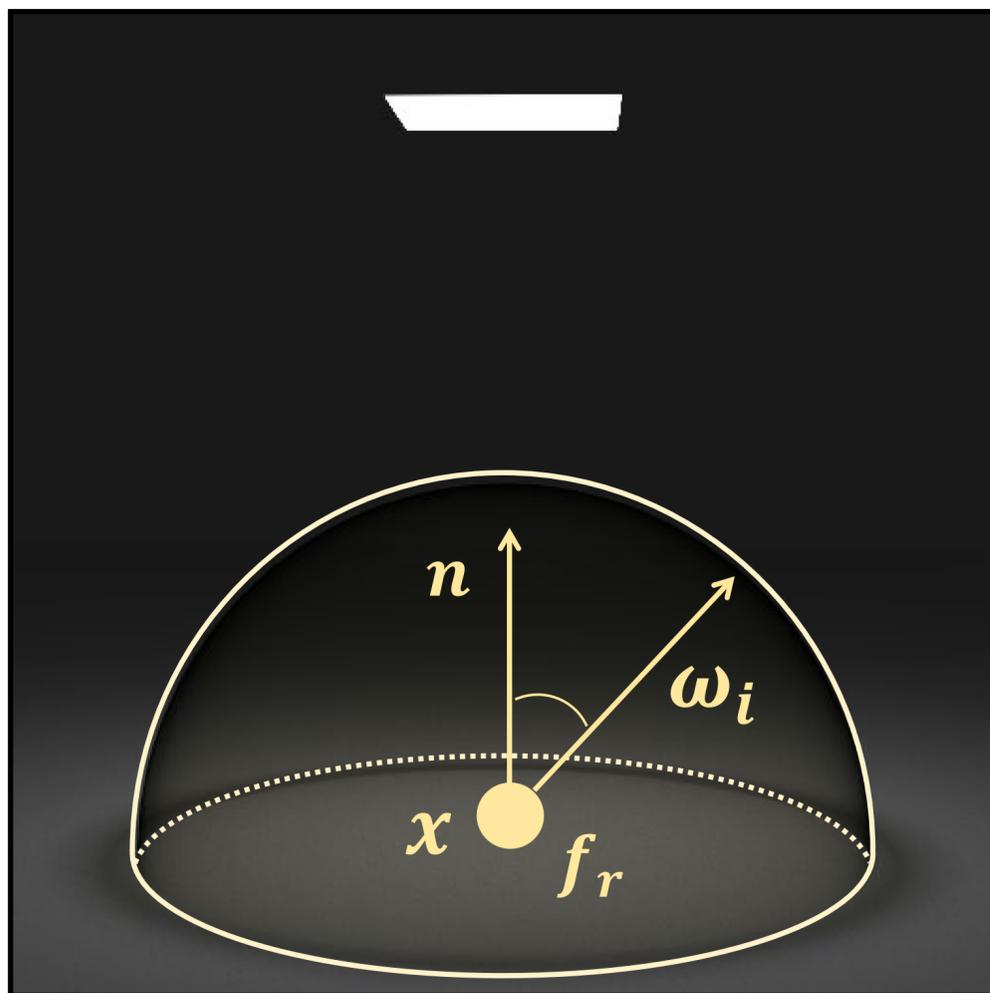
Just differentiate numerator

- Form estimator

[Khungurn et al. 2015, Gkioulekas et al. 2015]

$$\frac{dI}{d\pi} \approx \sum_s \frac{\frac{d}{d\pi} \{f_r(\omega_i^s, \omega_o) L_i(\omega_i^s) (n \cdot \omega_i^s)\}}{p(\omega_i^s)}$$

Alternative estimator



π : local parameters

- BRDF parameters

Differential radiance from x :

$$\frac{dI}{d\pi} = \int_{\mathbb{H}^2} \frac{d}{d\pi} \{f_r(\omega_i, \omega_o, \pi) L_i(\omega_i) (n \cdot \omega_i)\} d\sigma(\omega_i)$$

Just move derivative inside integral

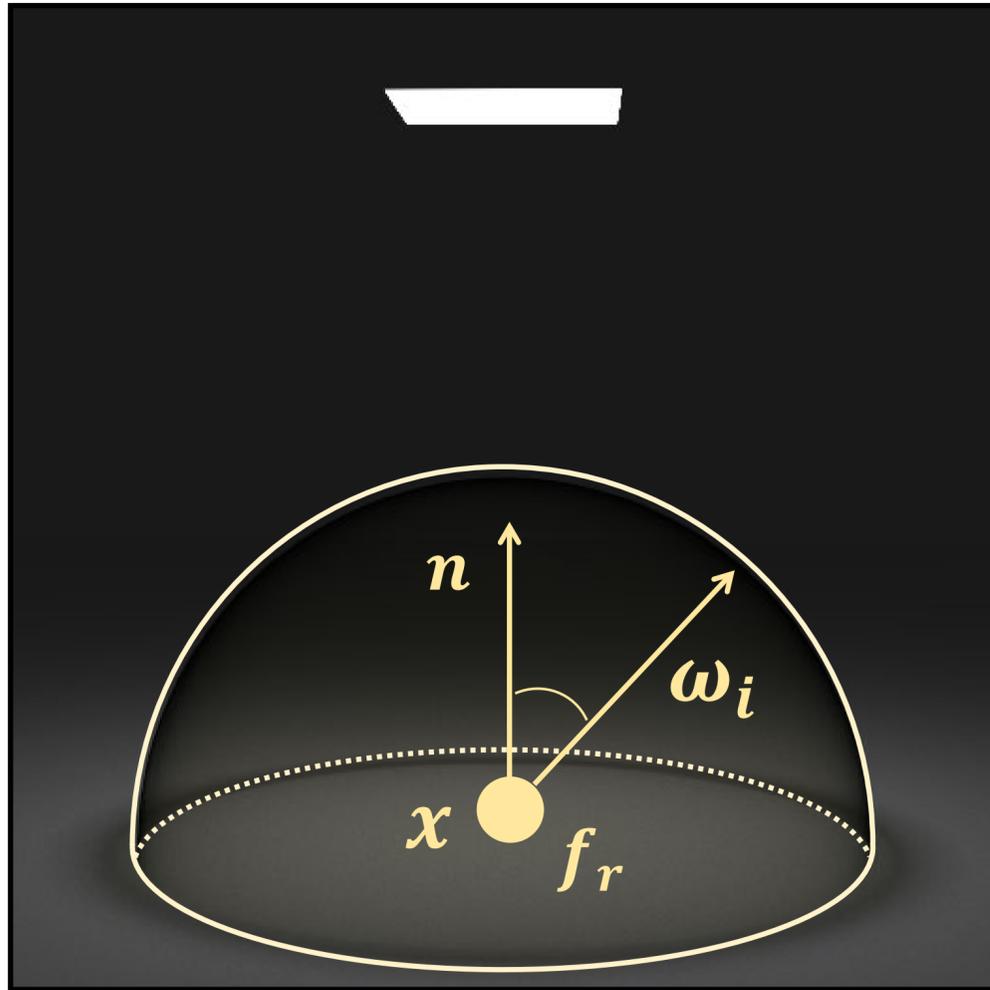
Monte Carlo estimation:

- Sample random directions ω_i^s from PDF $p(\omega_i, \pi)$
- Form estimator

Differentiate entire contribution
[Zeltner et al. 2021]

$$\frac{dI}{d\pi} \approx \sum_s \frac{d}{d\pi} \left\{ \frac{f_r(\omega_i^s, \omega_o, \pi) L_i(\omega_i^s) (n \cdot \omega_i^s)}{p(\omega_i^s, \pi)} \right\}$$

Differential direct illumination: global parameters



Differential radiance from x :

$$\frac{dI}{d\pi} = \frac{d}{d\pi} \int_{\mathbb{H}^2} f_r(\omega_i, \omega_o) L_i(\omega_i) (n \cdot \omega_i) d\sigma(\omega_i)$$

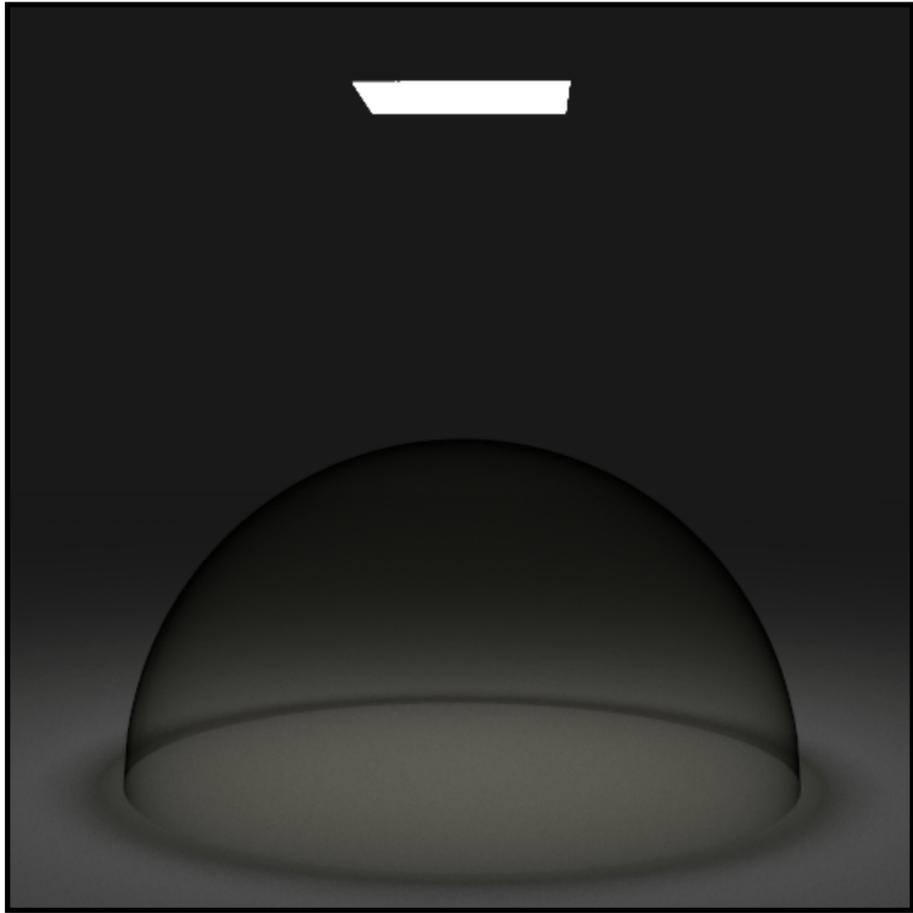
~~$$= \int_{\mathbb{H}^2} \frac{d}{d\pi} \{f_r(\omega_i, \omega_o) L_i(\omega_i) (n \cdot \omega_i)\} d\sigma(\omega_i)$$~~

Need to use full Reynolds transport theorem

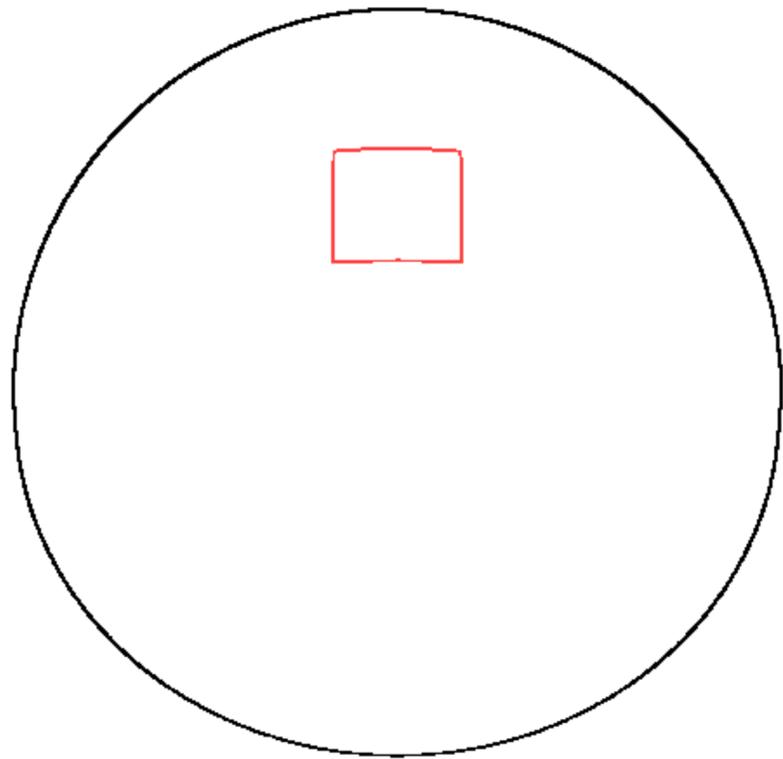
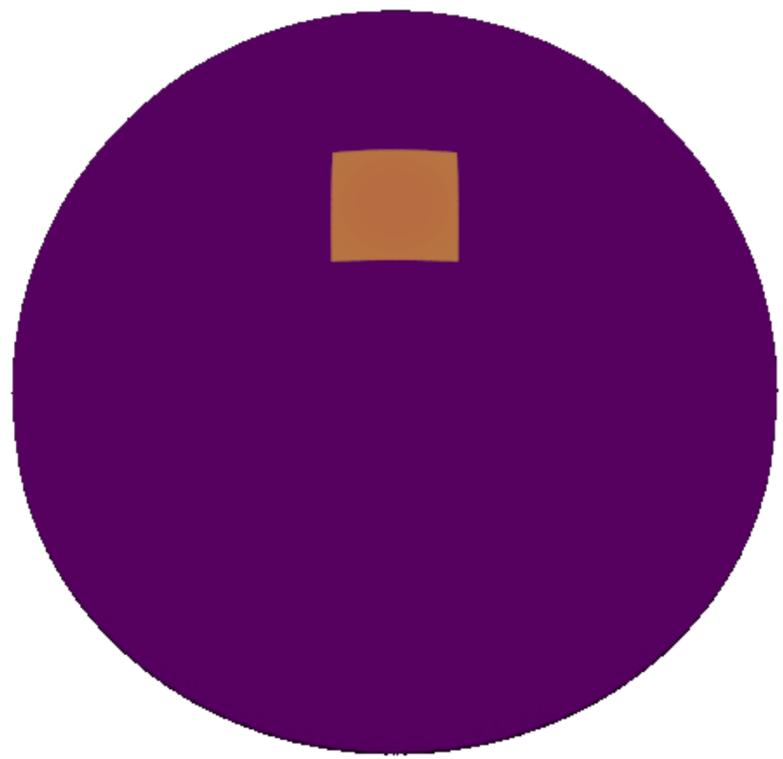
π : global parameters

- shape and pose of different scene elements (camera, sources, objects)

Discontinuities in the integrand



Low  High



π : size of the emitter

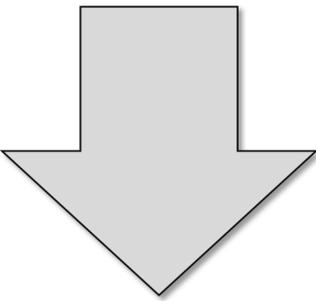
$$I = \int_{\mathbb{H}^2} \underbrace{f_r(\omega_i, \omega_o) L_i(\omega_i) (n \cdot \omega_i)}_{f(\omega_i)} d\sigma(\omega_i)$$

Integrand
 $f(\omega_i)$

Discontinuous points
(π -dependent)

Applying the Reynolds transport theorem

$$I = \int_{\mathbb{H}^2} f(\omega_i, \omega_o) d\sigma(\omega_i)$$

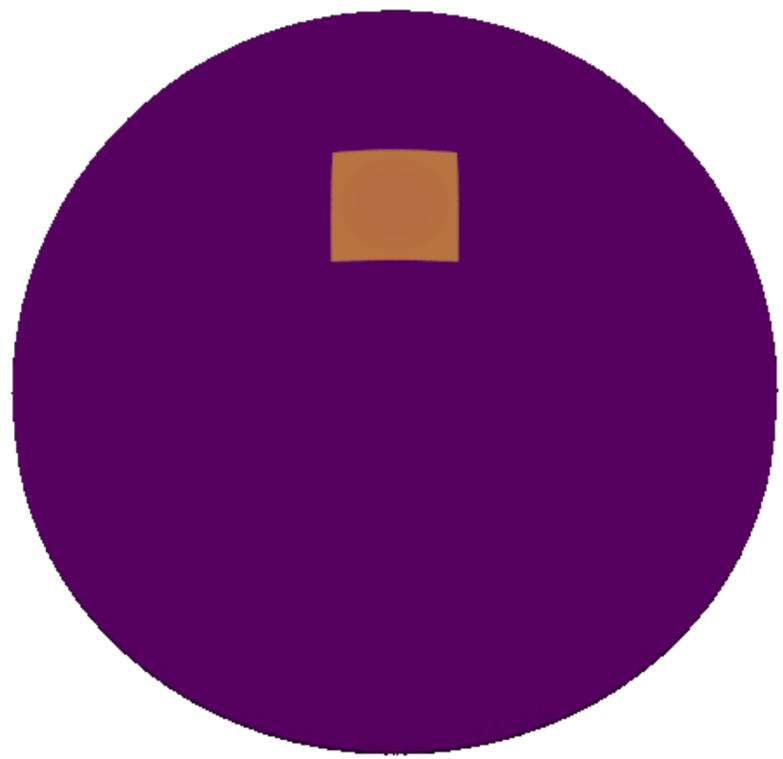


$$\frac{dI}{d\pi} = \int_{\mathbb{H}^2} \frac{df}{d\pi} d\sigma + \int_{\partial\mathbb{H}^2} g dl$$

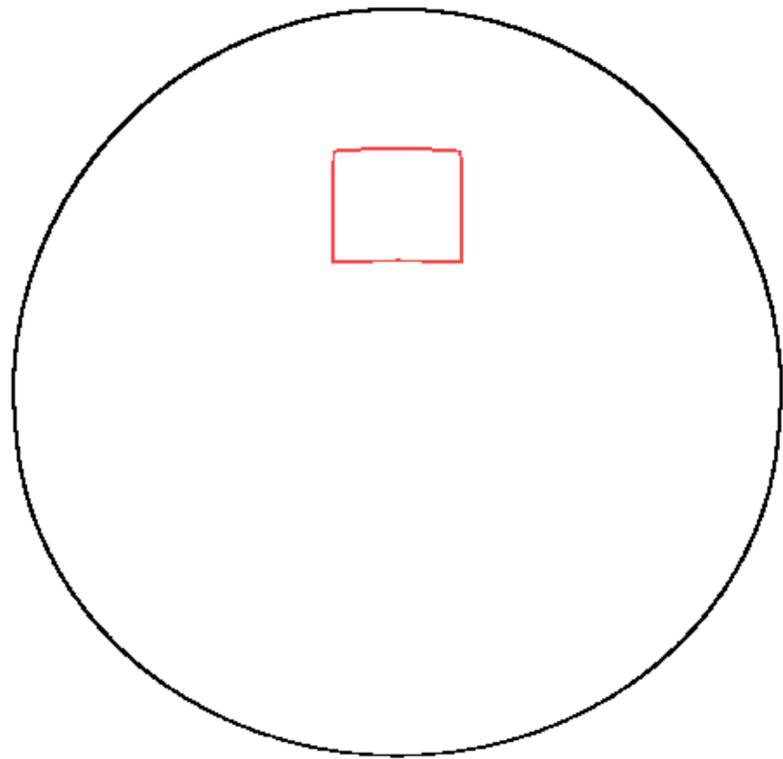
Interior integral
(same as for local parameters)

Boundary integral

Low  High



Integrand $f(\omega_i)$

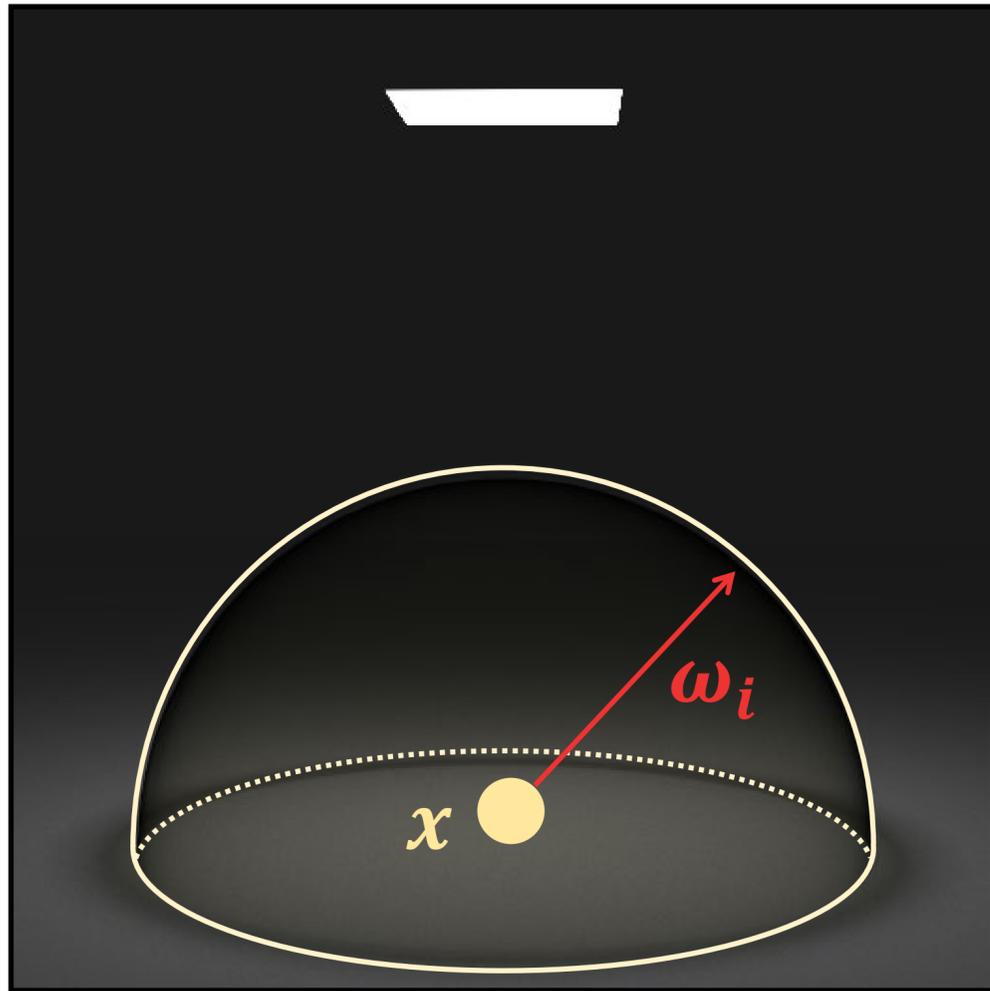


Discontinuous points (π -dependent)

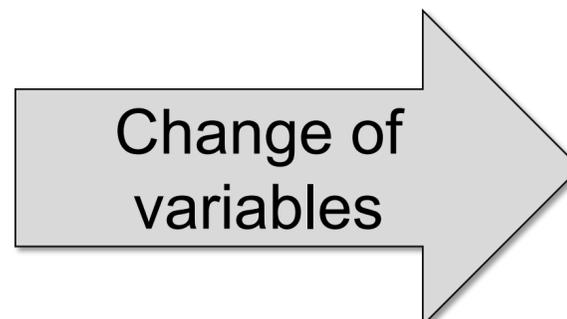
[Ramamoorthi et al. 2007, Li et al. 2019]

Reparameterizing the direct illumination integral

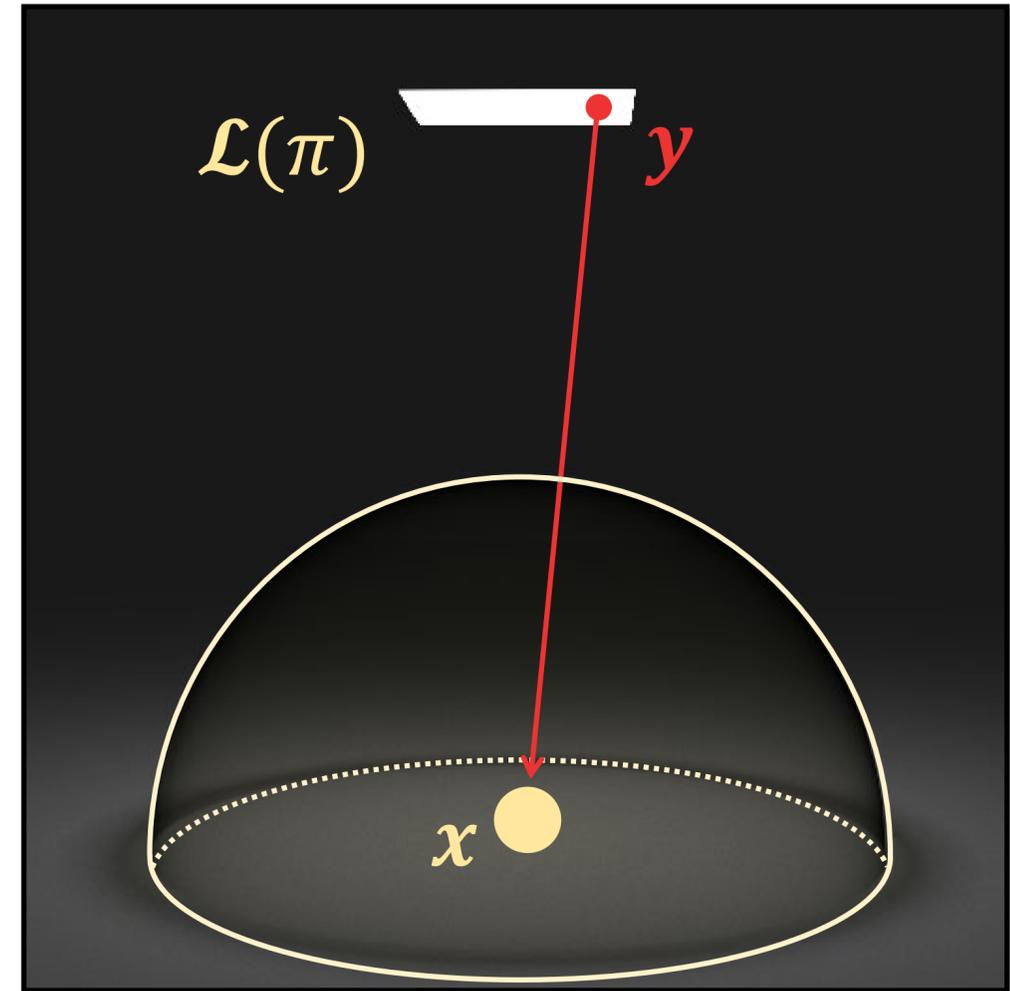
Hemispherical integral



$$I = \int_{\mathbb{H}^2} f(\boldsymbol{\omega}_i) d\sigma(\boldsymbol{\omega}_i)$$



Surface integral

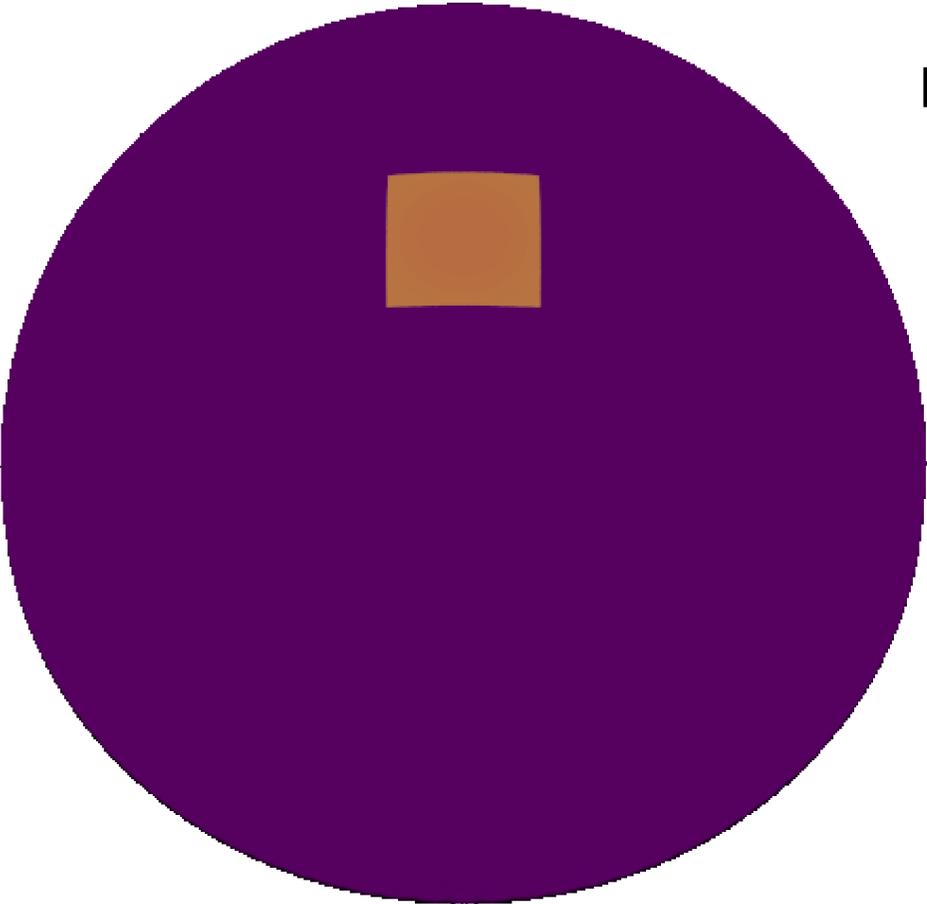


$$I = \int_{\mathcal{L}(\pi)} f(\mathbf{y} \rightarrow \mathbf{x}) G(\mathbf{x}, \mathbf{y}) dA(\mathbf{y})$$

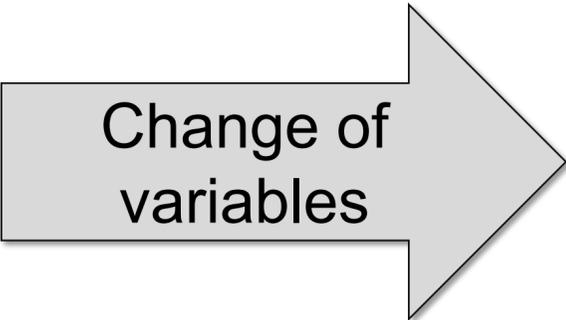
Includes visibility, fall-off,
and foreshortening terms

Reparameterizing the direct illumination integral

Hemispherical integral



Low  High



Surface integral



discontinuous

$$I = \int_{\mathbb{H}^2} f(\omega_i) d\sigma(\omega_i)$$

constant domain

continuous

$$I = \int_{\mathcal{L}(\pi)} f(y \rightarrow x) G(x, y) dA(y)$$

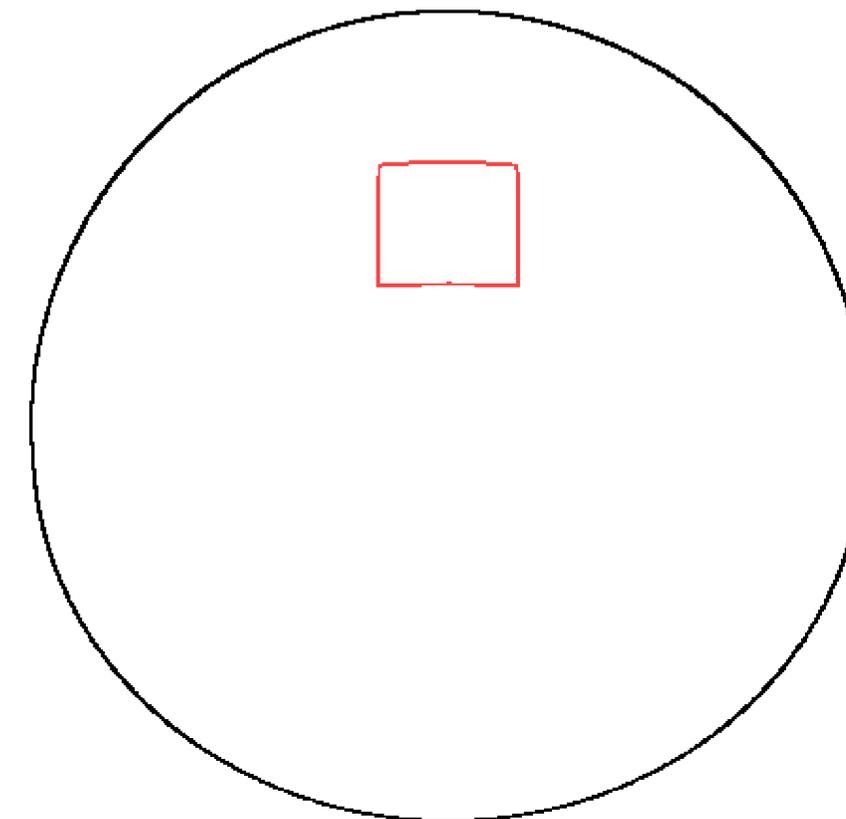
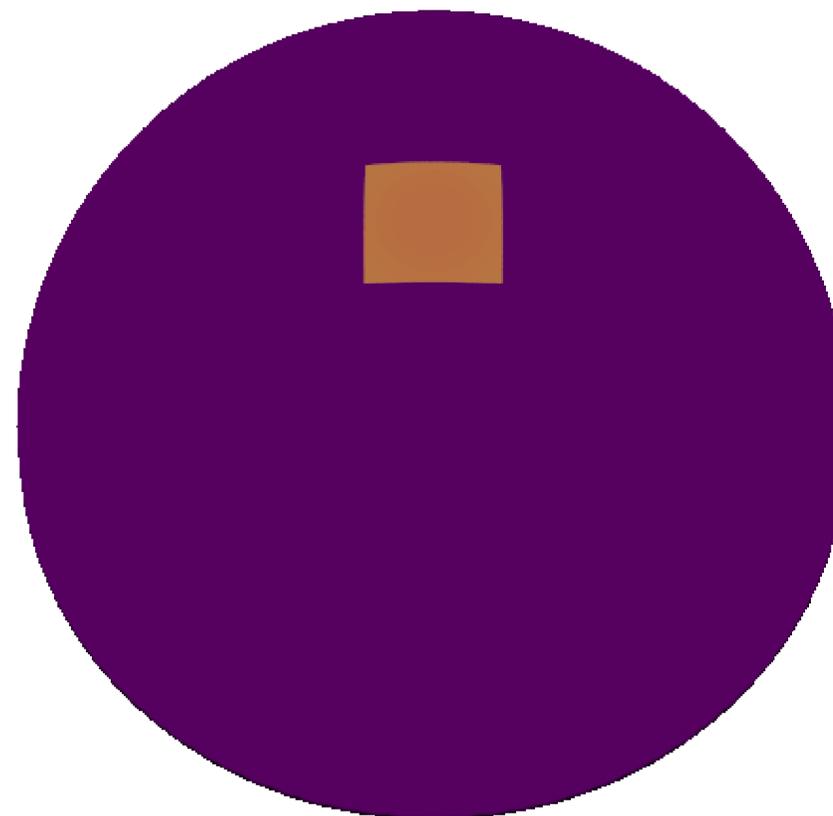
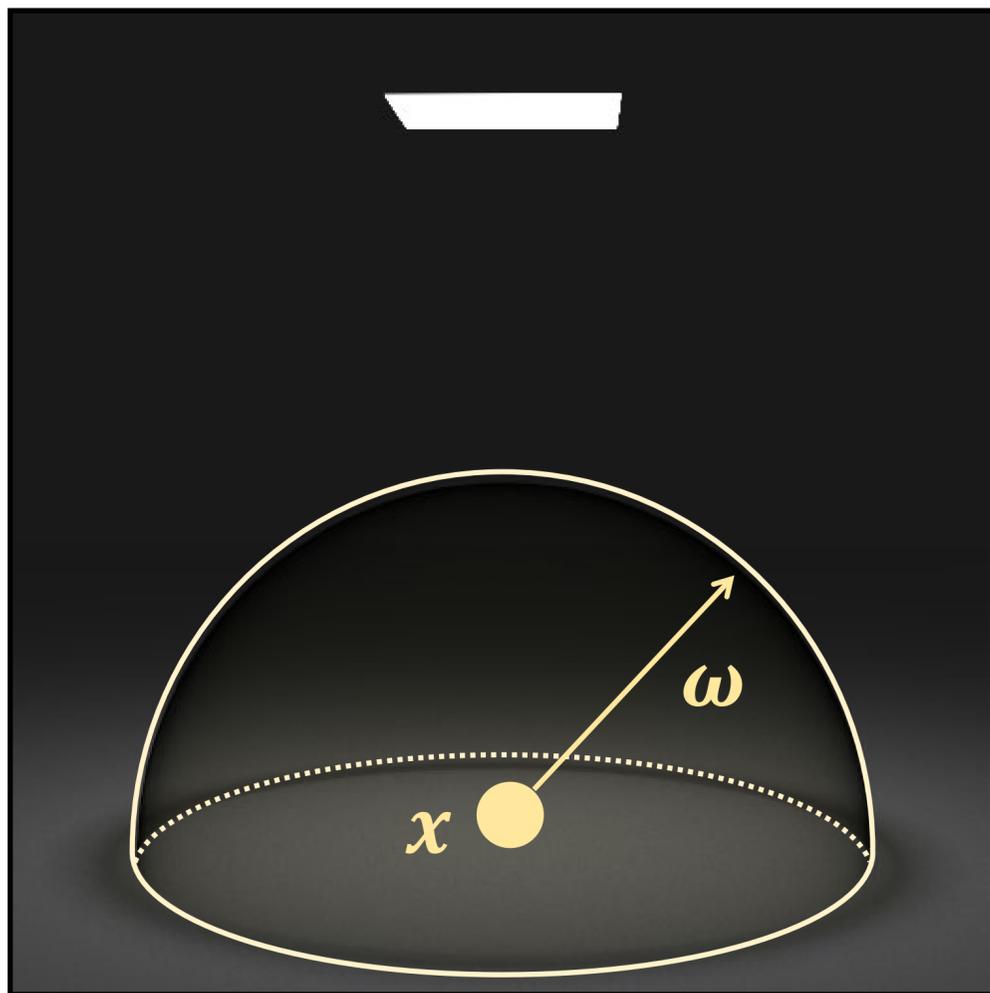
evolving domain

Differentiating the hemispherical integral

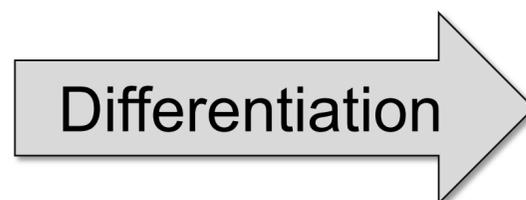
π : size of the emitter

Low  High

Discontinuities of f



$$I = \int_{\mathbb{H}^2} f(\omega_i) d\sigma(\omega_i)$$



Reynolds transport theorem

$$\frac{dI}{d\pi} = \int_{\mathbb{H}^2} \frac{d(f)}{d\pi} d\sigma + \int_{\partial\mathbb{H}^2} g dl$$

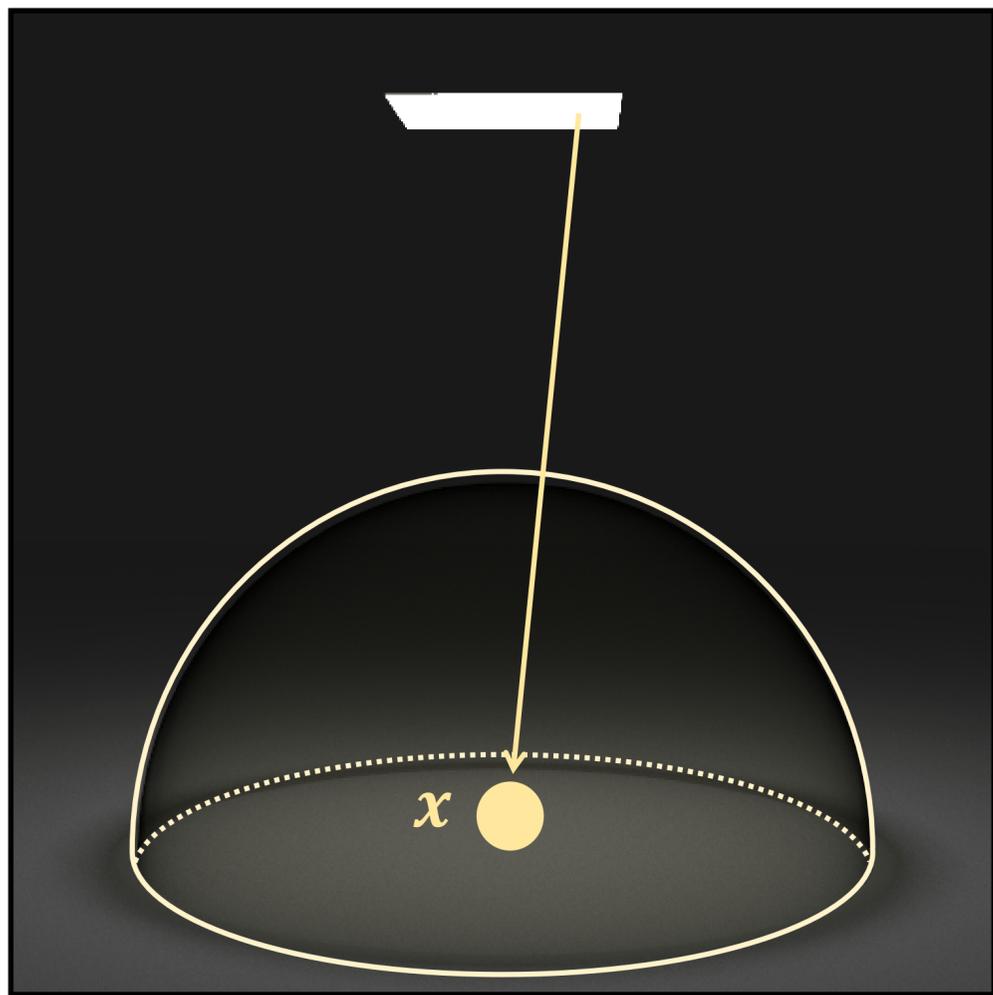
Interior Boundary

Differentiating the area integral

π : size of the emitter

Low  High

Boundary of $\mathcal{L}(\pi)$



$$I = \int_{\mathcal{L}(\pi)} f(y \rightarrow x) G(x, y) dA(y)$$



Reynolds transport theorem

$$\frac{dI}{d\pi} = \int_{\mathcal{L}(\pi)} \frac{d(fG)}{d\pi} dA + \int_{\partial\mathcal{L}(\pi)} g dl$$

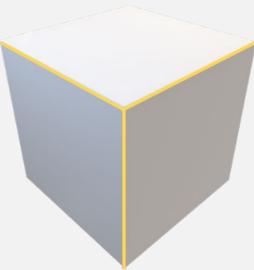
Interior
Boundary

Sources of discontinuities

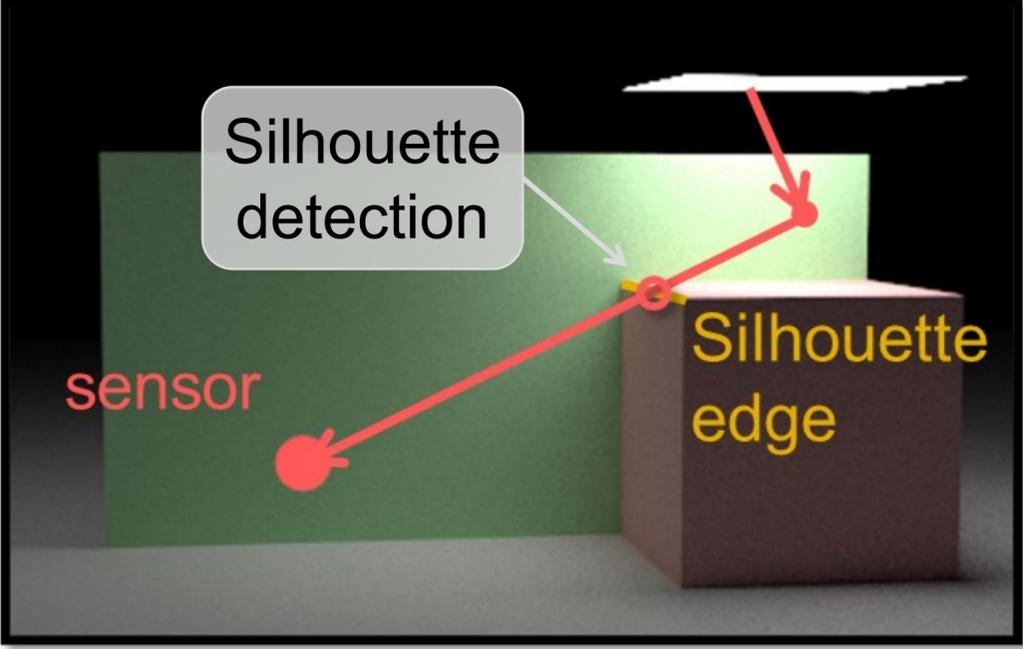
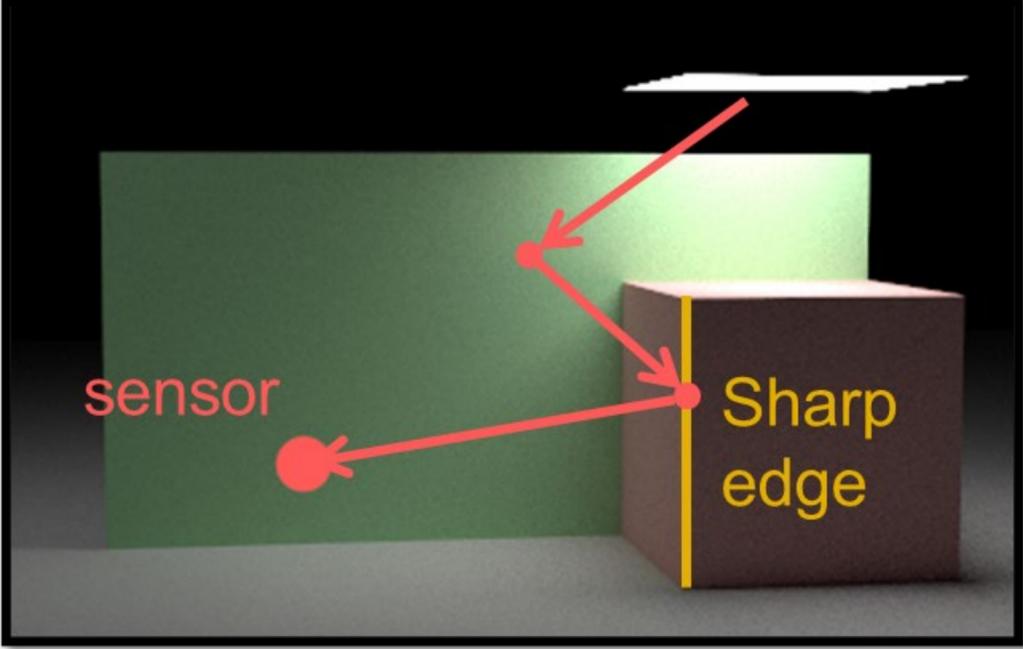
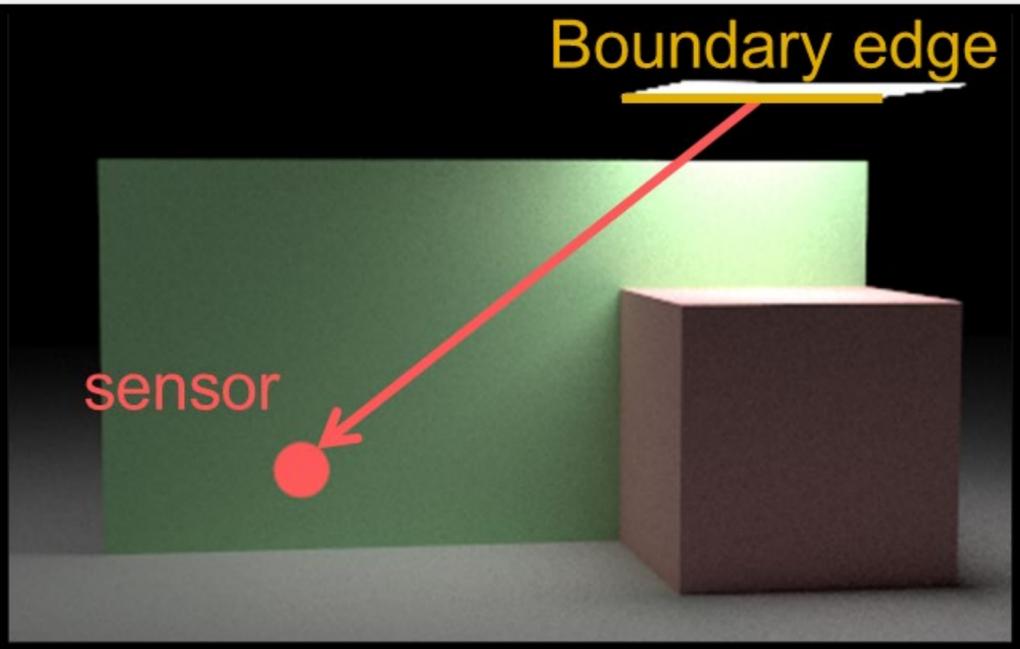
Boundary edge



Sharp edge



Silhouette edge

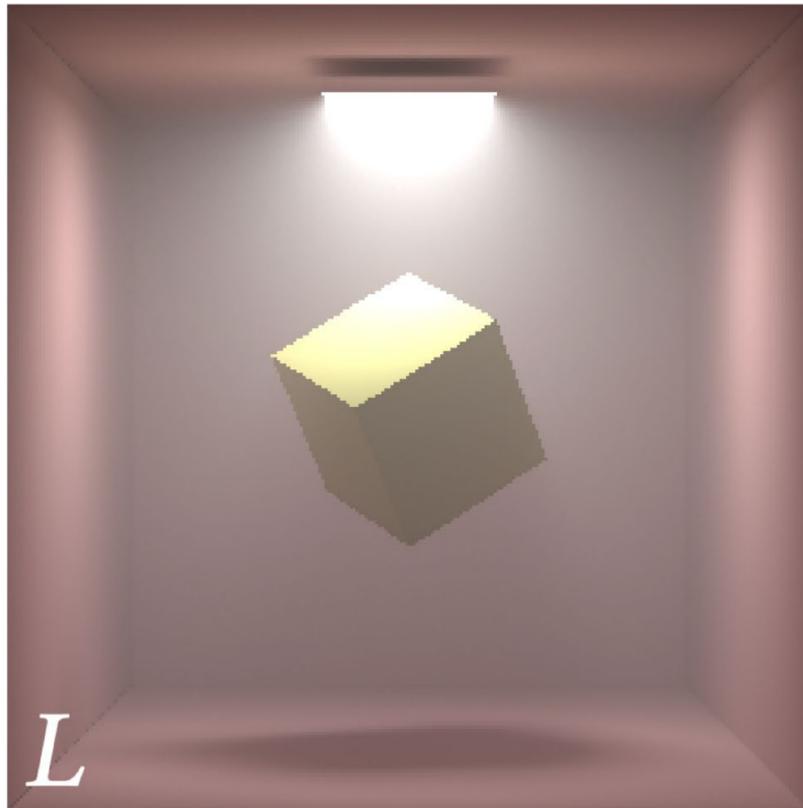


Topology-driven

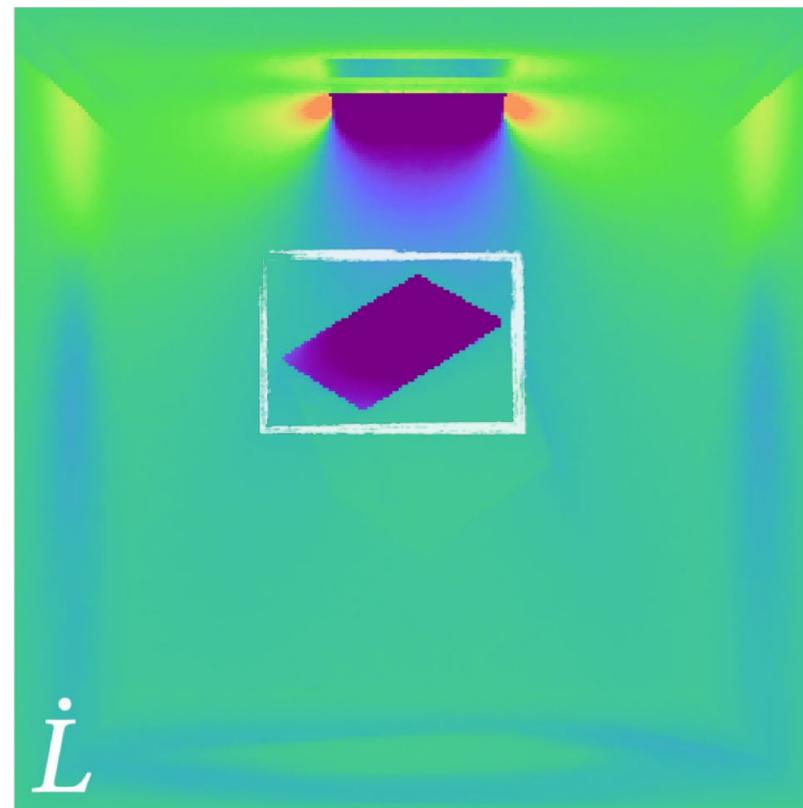
Visibility-driven

Significance of the boundary integral

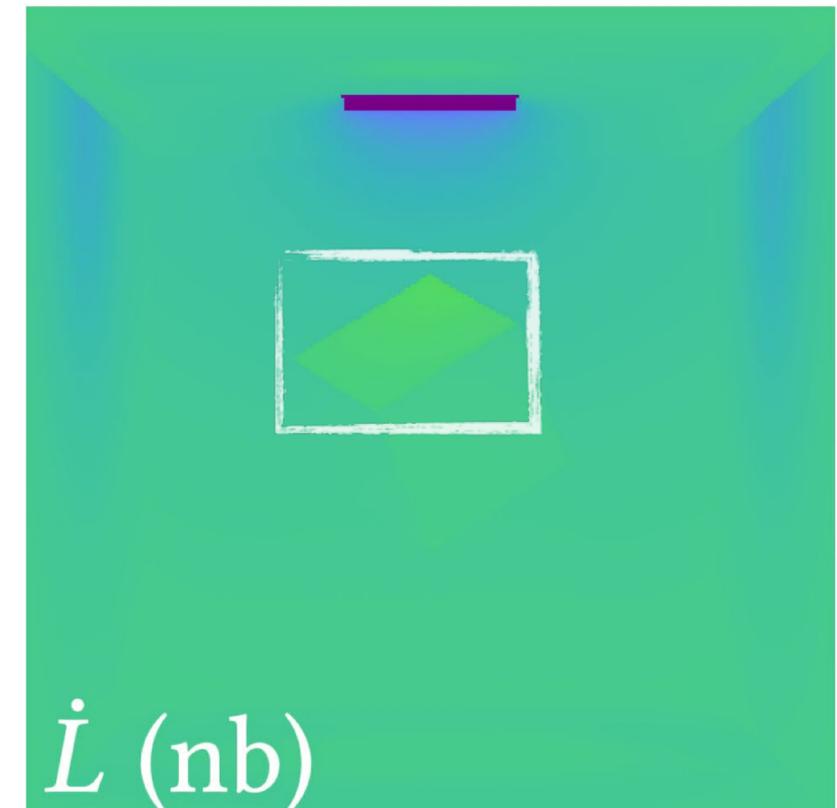
Negative  Zero  Positive



Original image



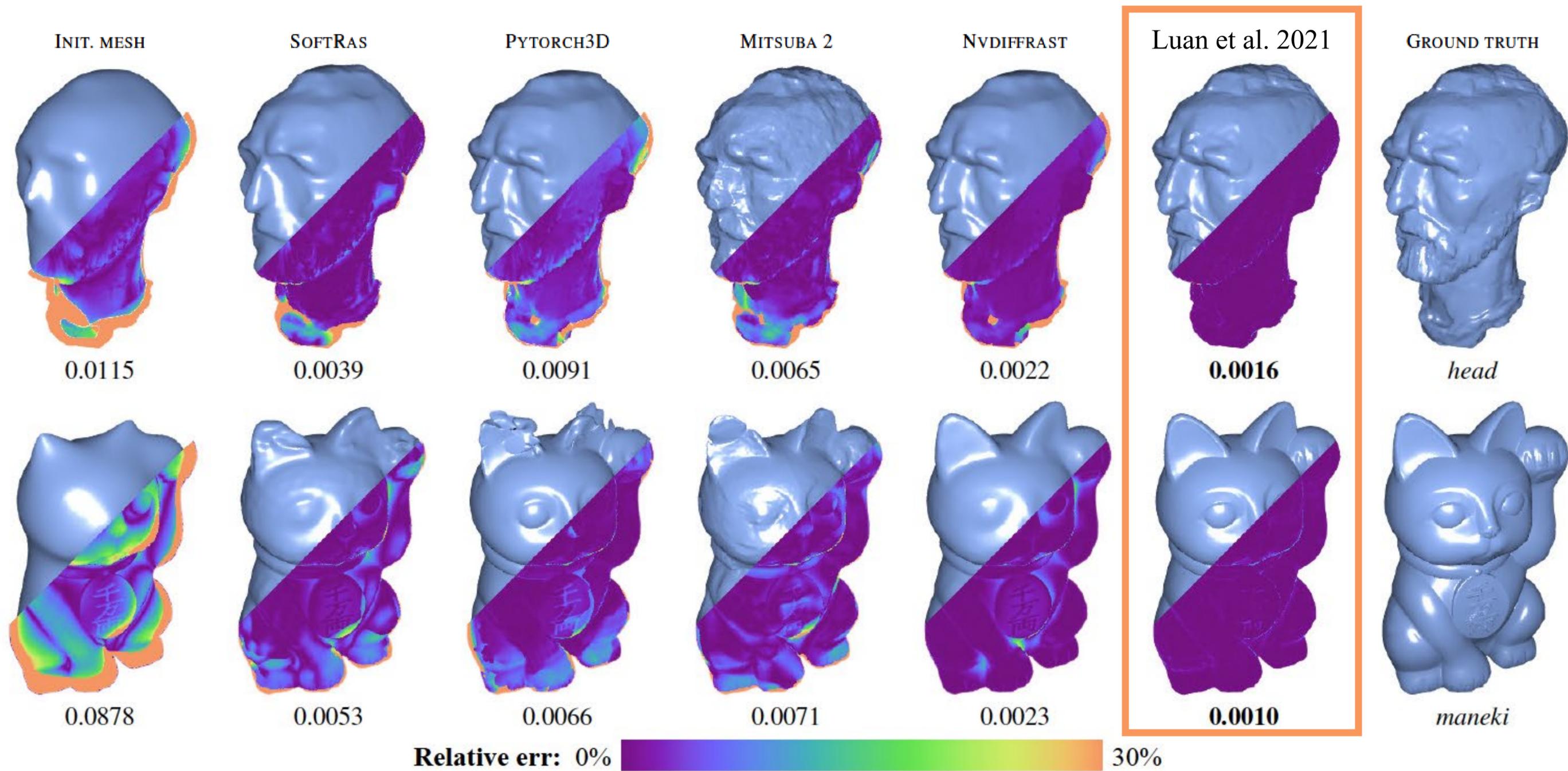
Derivative image
w.r.t. vertical offset of
the area light and the cube



Derivative image
w/o boundary integral

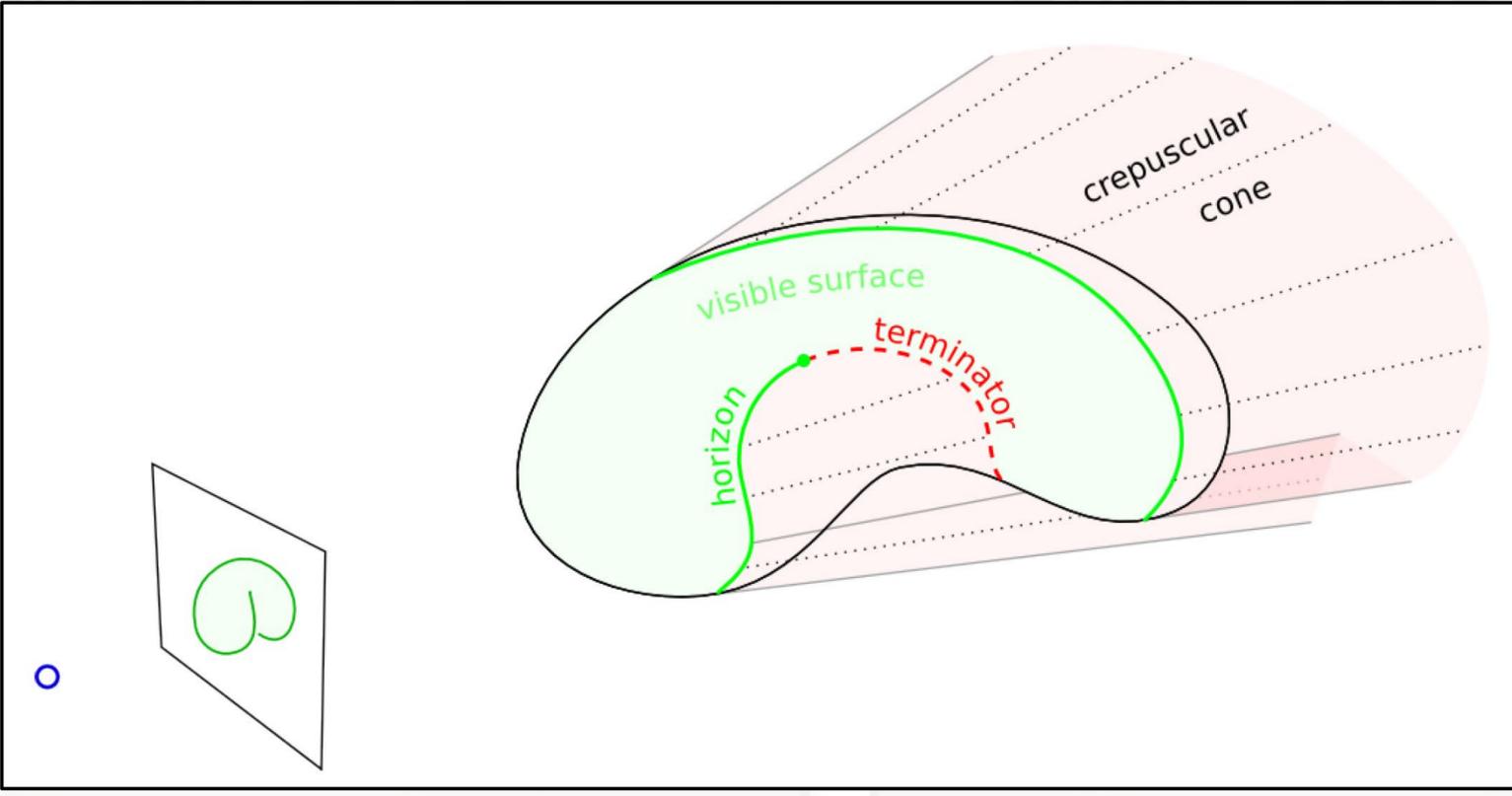
Gradient Accuracy Matters

Inverse-rendering results with *identical* optimization settings



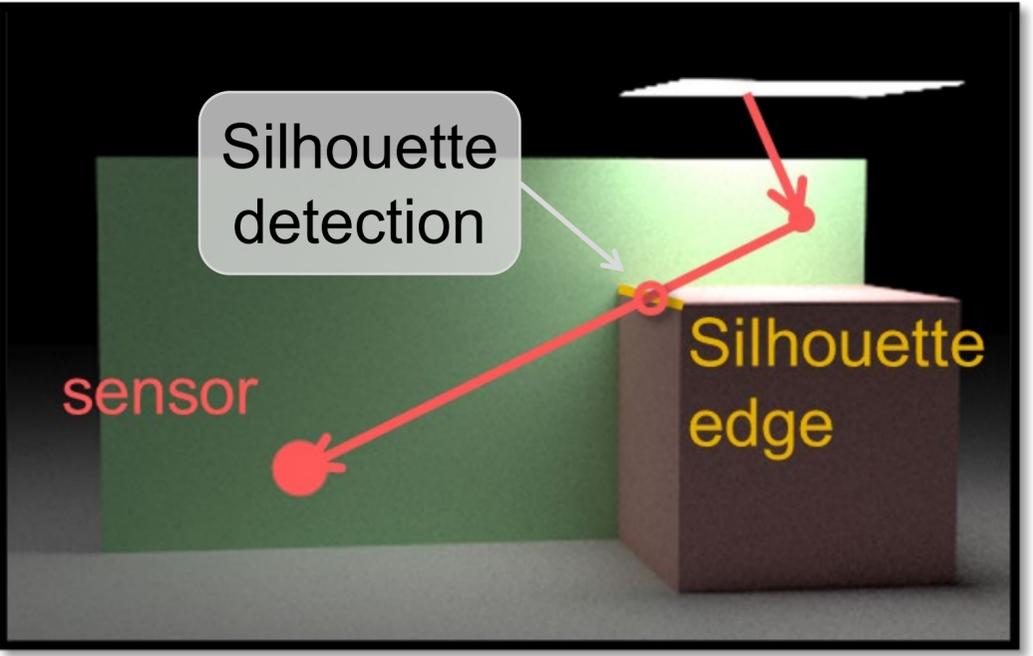
Sources of discontinuities

- We still need to account for visibility discontinuities when using smooth closed surfaces (e.g., neural SDFs)



[Gargallo et al., ICCV 2007]

Silhouette edge



Visibility-driven

Handling Global Illumination

Background: Path Integral for Global Illumination

$$I = \int_{\Omega} f(\bar{x}) d\mu(\bar{x})$$

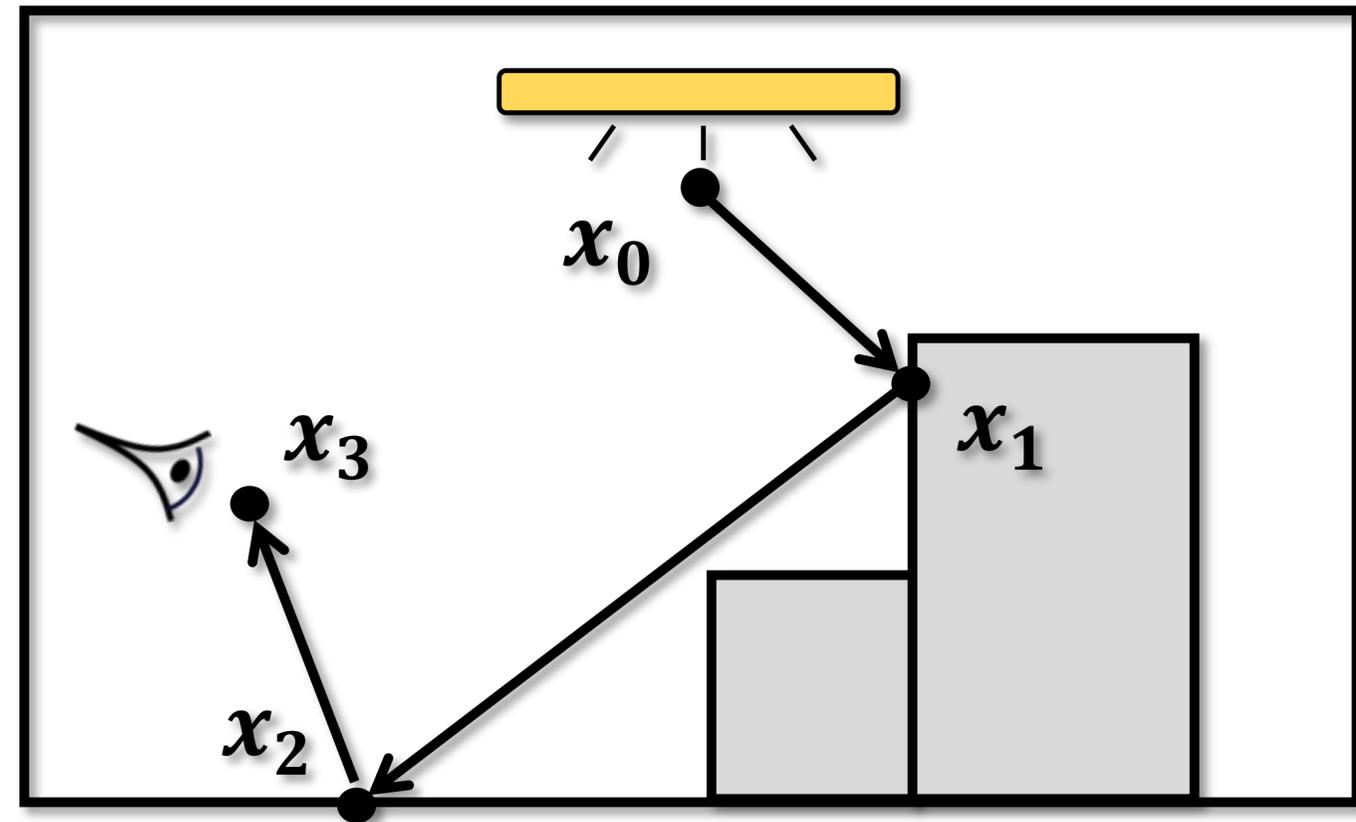
Pixel value

Measurement contribution

Path space

Area-product measure

- Introduced by Veach [1997] and extended by Pauly et al. [2000]
- Can capture both surface reflection/refraction and volumetric (i.e., subsurface) scattering
- Theoretical foundation of most modern forward rendering techniques



Light path $\bar{x} = (x_0, x_1, x_2, x_3)$

Background: Estimating Path Integrals

Pixel value

$$I = \int_{\Omega} f(\bar{x}) d\mu(\bar{x})$$

Measurement contribution

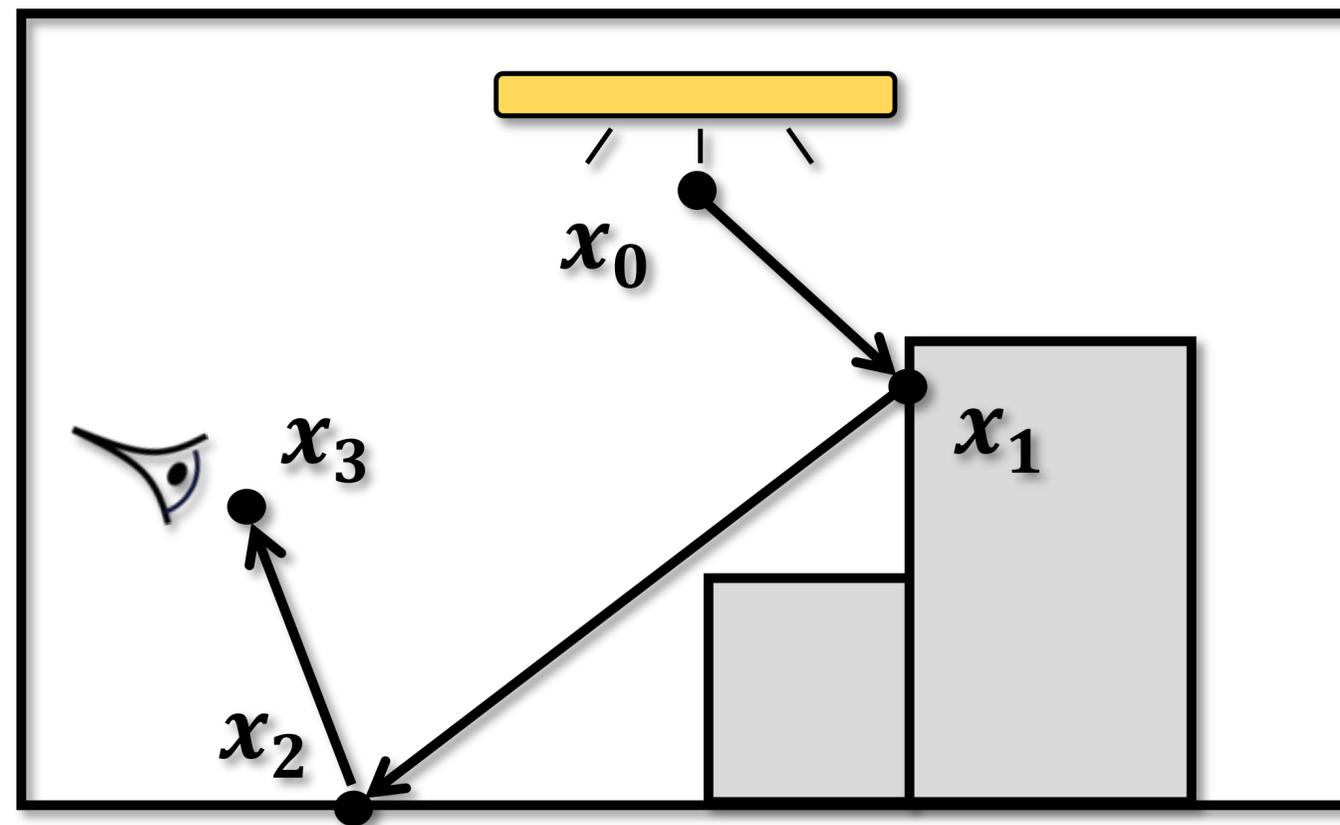
Path space

Area-product measure

Monte Carlo estimator:

$$\langle I \rangle = \frac{f(\bar{x})}{p(\bar{x})}$$

Probability density for sampling path \bar{x}



Light path $\bar{x} = (x_0, x_1, x_2, x_3)$

Differential Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral **Boundary** integral

We now derive $\partial I_N / \partial \pi$ in Eq. (25) using the recursive relations provided by Eqs. (21) and (24). Let

$$h_n^{(0)} := \left[\prod_{n'=n+1}^N g(\mathbf{x}_{n'}; \mathbf{x}_{n'-2}, \mathbf{x}_{n'-1}) \right] W_e(\mathbf{x}_N \rightarrow \mathbf{x}_{N-1}), \quad (52)$$

$$h_n^{(1)} := \sum_{n'=n+1}^N \kappa(\mathbf{x}_{n'}) V(\mathbf{x}_{n'}), \quad (53)$$

$$\Delta h_{n,n'}^{(0)} := h_n^{(0)} \Delta g(\mathbf{x}_{n'}; \mathbf{x}_{n'-2}, \mathbf{x}_{n'-1}) / g(\mathbf{x}_{n'}; \mathbf{x}_{n'-2}, \mathbf{x}_{n'-1}), \quad (54)$$

for $0 \leq n < n' \leq N$. We omit the dependencies of $h_n^{(0)}$, $h_n^{(1)}$, and $\Delta h_{n,n'}^{(0)}$ on $\mathbf{x}_{n+1}, \dots, \mathbf{x}_N$ for notational convenience.

We now show that, for all $0 \leq n < N$, it holds that

$$h_n(\mathbf{x}_n; \mathbf{x}_{n-1}) = \int_{\mathcal{M}^{N-n}} h_n^{(0)} \prod_{n'=n+1}^N dA(\mathbf{x}_{n'}), \quad (55)$$

and

$$\dot{h}_n(\mathbf{x}_n; \mathbf{x}_{n-1}) = \int_{\mathcal{M}^{N-n}} \left[\left(h_n^{(0)} \right)' - h_n^{(0)} h_n^{(1)} \right] \prod_{n'=n+1}^N dA(\mathbf{x}_{n'}) + \sum_{n'=n+1}^N \int \Delta h_{n,n'}^{(0)} V_{\partial \mathcal{M}_{n'}}(\mathbf{x}_{n'}) d\ell(\mathbf{x}_{n'}) \prod_{\substack{n < i \leq N \\ i \neq n'}} dA(\mathbf{x}_i), \quad (56)$$

where the integral domain of the second term on the right-hand side, which is omitted for notational clarity, is $\mathcal{M}(\pi)$ for each \mathbf{x}_i with $i \neq n'$ and $\partial \mathcal{M}_{n'}(\pi)$, which depends on $\mathbf{x}_{n'-1}$, for $\mathbf{x}_{n'}$.

It is easy to verify that Eqs. (55) and (56) hold for $n = N - 1$. We now show that, if they hold for some $0 < n < N$, then it is also the case for $n - 1$. Let $g_{n-1} := g(\mathbf{x}_n; \mathbf{x}_{n-2}, \mathbf{x}_{n-1})$ for all $0 < n \leq N$. Then,

$$h_{n-1}(\mathbf{x}_{n-1}; \mathbf{x}_{n-2}) = \int_{\mathcal{M}} g_{n-1} \int_{\mathcal{M}^{N-n}} h_n^{(0)} \prod_{n'=n+1}^N dA(\mathbf{x}_{n'}) dA(\mathbf{x}_n) = \int_{\mathcal{M}^{N-n+1}} h_{n-1}^{(0)} \prod_{n'=n}^N dA(\mathbf{x}_{n'}), \quad (57)$$

and

$$\begin{aligned} & \dot{h}_{n-1}(\mathbf{x}_{n-1}; \mathbf{x}_{n-2}) \\ &= \int_{\mathcal{M}} \left[\dot{g}_{n-1} h_n + g_{n-1} (\dot{h}_n - h_n \kappa(\mathbf{x}_n) V(\mathbf{x}_n)) \right] dA(\mathbf{x}_n) \\ & \quad + \int_{\partial \mathcal{M}_n} \Delta g_{n-1} h_n V_{\partial \mathcal{M}_n} d\ell(\mathbf{x}_n) \\ &= \int_{\mathcal{M}^{N-n+1}} \left\{ \dot{g}_{n-1} h_n^{(0)} + g_{n-1} \left[\left(h_n^{(0)} \right)' - h_n^{(0)} h_n^{(1)} \right] \right\} \prod_{n'=k}^N dA(\mathbf{x}_{n'}) \\ & \quad + \sum_{n'=n+1}^N \int g_{n-1} \Delta h_{n,n'}^{(0)} V_{\partial \mathcal{M}_{n'}}(\mathbf{x}_{n'}) d\ell(\mathbf{x}_{n'}) \prod_{\substack{n \leq i \leq N \\ i \neq n'}} dA(\mathbf{x}_i) \\ & \quad + \int \Delta g_{n-1} h_n^{(0)} V_{\partial \mathcal{M}_n} d\ell(\mathbf{x}_n) \prod_{n'=n+1}^N dA(\mathbf{x}_{n'}) \\ &= \int_{\mathcal{M}^{N-n+1}} \left[\left(h_{n-1}^{(0)} \right)' - h_{n-1}^{(0)} h_{n-1}^{(1)} \right] \prod_{n'=n}^N dA(\mathbf{x}_{n'}) \\ & \quad + \sum_{n'=n}^N \int \Delta h_{n-1,n'}^{(0)} V_{\partial \mathcal{M}_{n'}}(\mathbf{x}_{n'}) d\ell(\mathbf{x}_{n'}) \prod_{\substack{n \leq i \leq N \\ i \neq n'}} dA(\mathbf{x}_i). \quad (58) \end{aligned}$$

Thus, using mathematical induction, we know that Eqs. (55) and (56) hold for all $0 \leq n < N$.

Notice that $h_0^{(0)} = f$ and $\Delta h_{0,n'}^{(0)} = \Delta f_{n'}$, where $\Delta f_{n'}$ follows the definition in Eq. (28). Letting $n = 0$ in Eq. (56) yields

$$\dot{h}_0(\mathbf{x}_0) = \int_{\mathcal{M}^N} \left[\dot{f}(\bar{\mathbf{x}}) - f(\bar{\mathbf{x}}) \sum_{n'=1}^N \kappa(\mathbf{x}_{n'}) V(\mathbf{x}_{n'}) \right] \prod_{n'=1}^N dA(\mathbf{x}_{n'}) + \sum_{n'=1}^N \int \Delta f_{n'}(\bar{\mathbf{x}}) V_{\partial \mathcal{M}_{n'}} d\ell(\mathbf{x}_{n'}) \prod_{\substack{0 < i \leq N \\ i \neq n'}} dA(\mathbf{x}_i). \quad (59)$$

Lastly, based on the assumption that h_0 is continuous in \mathbf{x}_0 , Eq. (25) can be obtained by differentiating Eq. (23):

$$\begin{aligned} \frac{\partial I_N}{\partial \pi} &= \frac{\partial}{\partial \pi} \int_{\mathcal{M}} h_0(\mathbf{x}_0) dA(\mathbf{x}_0) \\ &= \int_{\mathcal{M}} \left[\dot{h}_0(\mathbf{x}_0) - h_0(\mathbf{x}_0) \kappa(\mathbf{x}_0) V(\mathbf{x}_0) \right] dA(\mathbf{x}_0) \\ & \quad + \int_{\partial \mathcal{M}_0} h_0(\mathbf{x}_0) V_{\partial \mathcal{M}_0}(\mathbf{x}_0) d\ell(\mathbf{x}_0) \quad (60) \\ &= \int_{\Omega_N} \left[\dot{f}(\bar{\mathbf{x}}) - f(\bar{\mathbf{x}}) \sum_{K=0}^N \kappa(\mathbf{x}_K) V(\mathbf{x}_K) \right] d\mu(\bar{\mathbf{x}}) \\ & \quad + \sum_{K=0}^N \int_{\Omega_{N,K}} \Delta f_K(\bar{\mathbf{x}}) V_{\partial \mathcal{M}_K} d\mu'_{N,K}(\bar{\mathbf{x}}). \end{aligned}$$

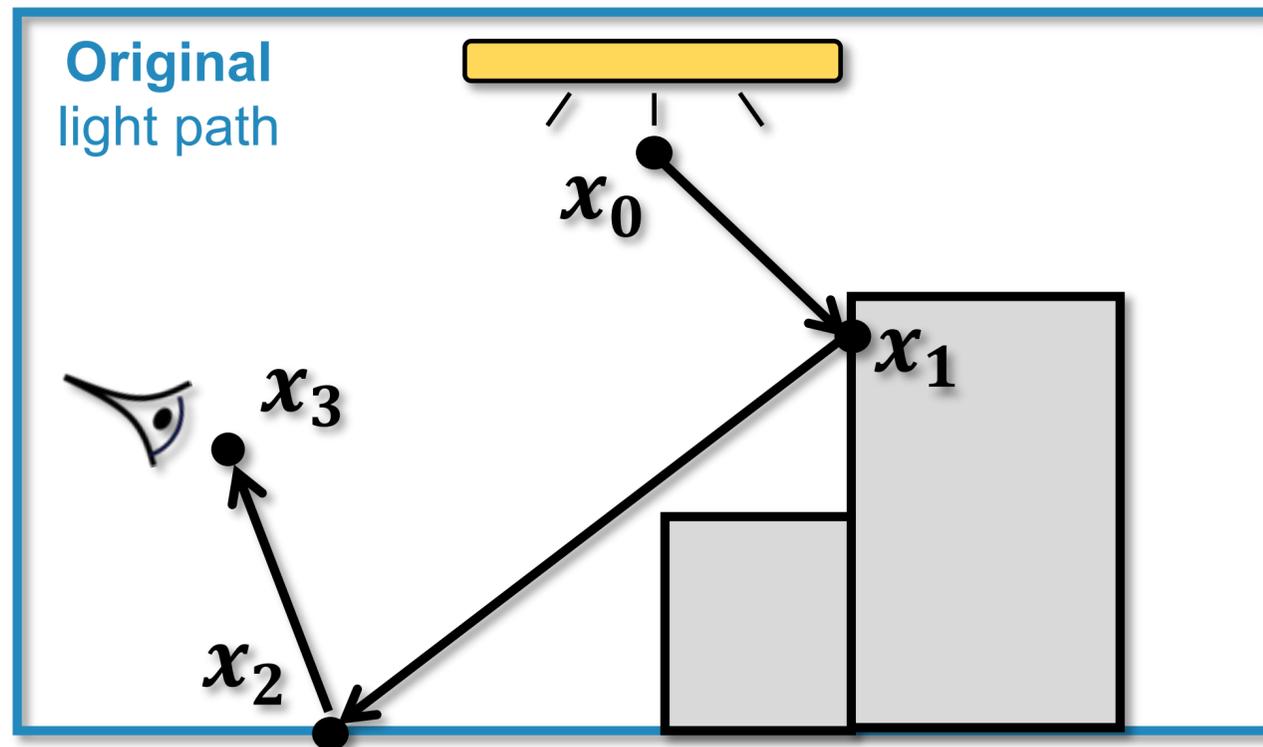
(The full derivation is quite involved...)

Differential Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral



Interior integral

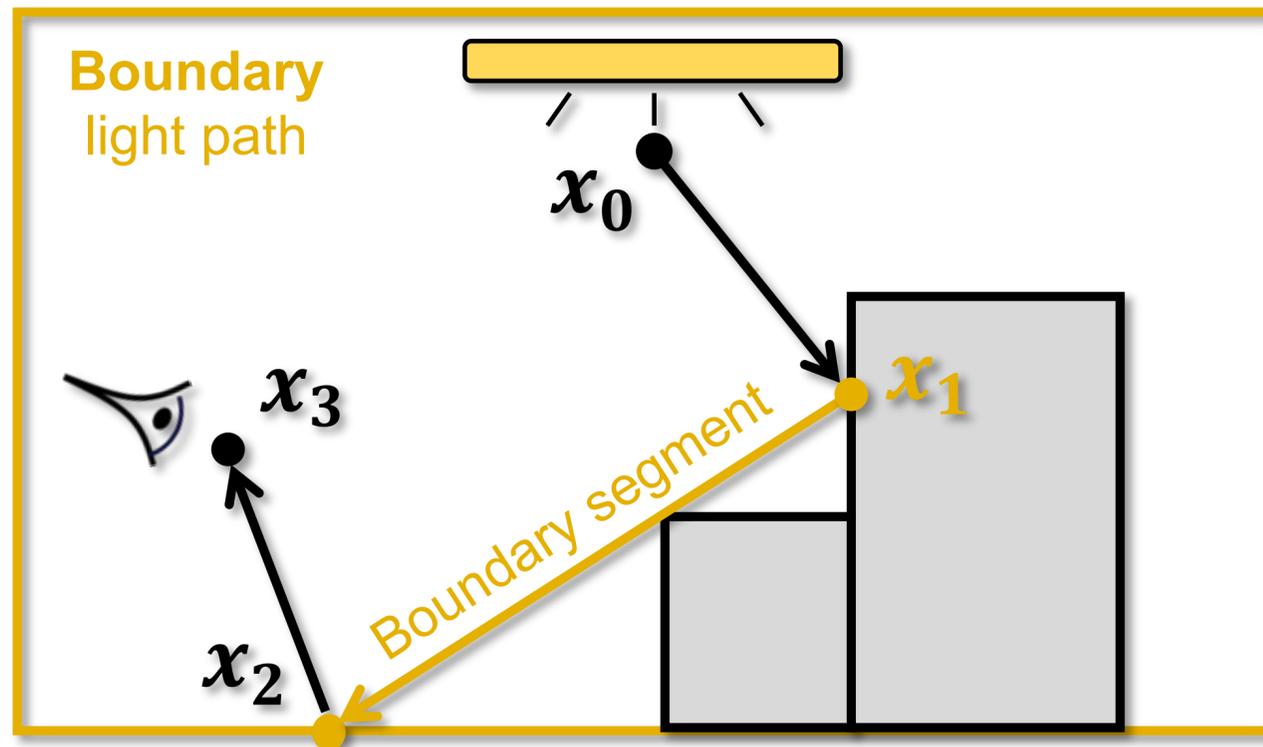
- Defined on the ordinary path space Ω
- The integrand \dot{f} can be obtained by differentiating the ordinary *measurement contribution function* f

Differential Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Boundary integral



Boundary integral

- Defined on the boundary path space $\partial\Omega$
- A **boundary** light path is the same as an original one except having exactly one **boundary** segment

Differential Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral **Boundary** integral

Physics-based **differentiable** rendering generally requires estimating **both integrals**

Challenges:

- Differentiating f w.r.t. many parameters (**interior**)
- Handling discontinuities (**boundary**)

Differential Interior Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral

- Computing \dot{f} requires differentiating f w.r.t. θ
- This can be done via **automatic differentiation**, but ...
 - We have many (e.g., 10^6) path integrals to evaluate (one per pixel)
 - There can be many (e.g., 10^6) parameters
 - Huge gradient matrices (e.g., with 10^{12} entries), not enough memory!

Specialized *computational differentiation* methods have been developed [Nimier-David et al. 2020, Vicini et al. 2021]



Differential Path Integral

Path-space differentiable rendering [Zhang et al. 2020, 2021]

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right) = \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral **Boundary** integral

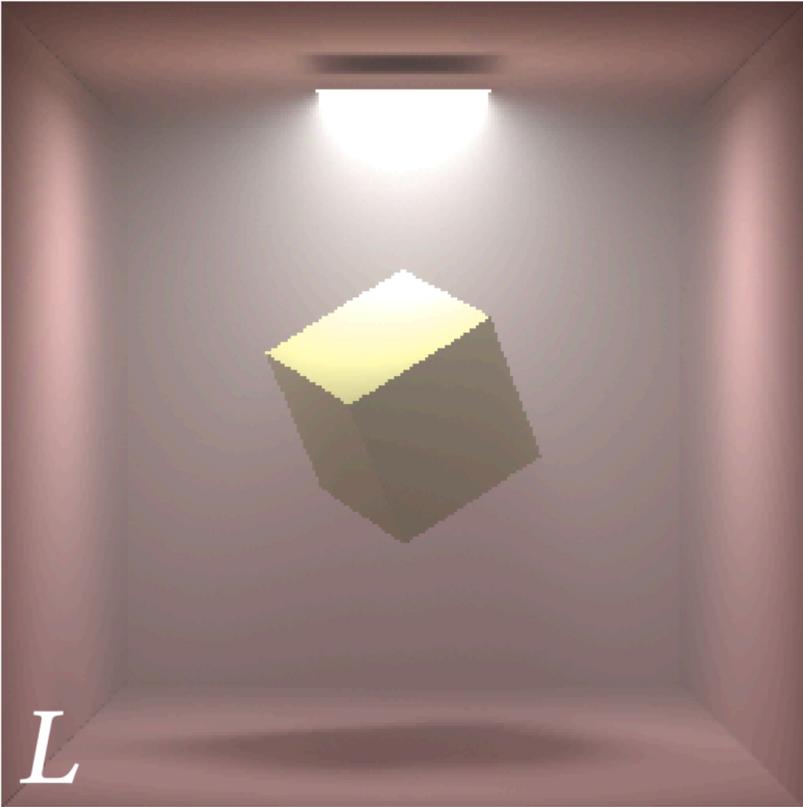
Physics-based **differentiable** rendering generally requires estimating **both integrals**

Challenges:

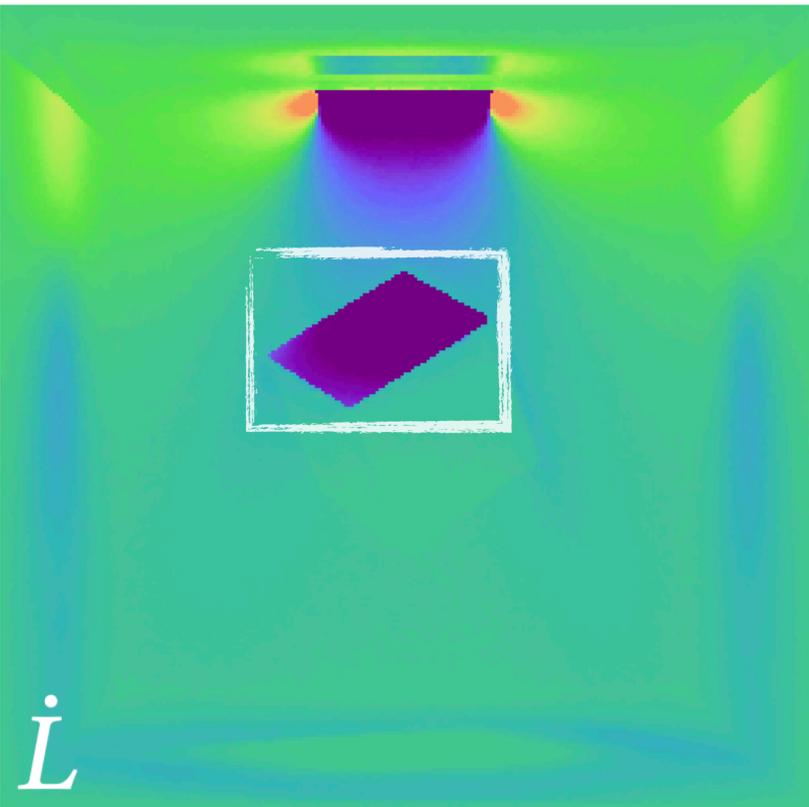
- Differentiating f w.r.t. many parameters (**interior**)
- Handling discontinuities (**boundary**)

Recap: Significance of the Boundary Integral

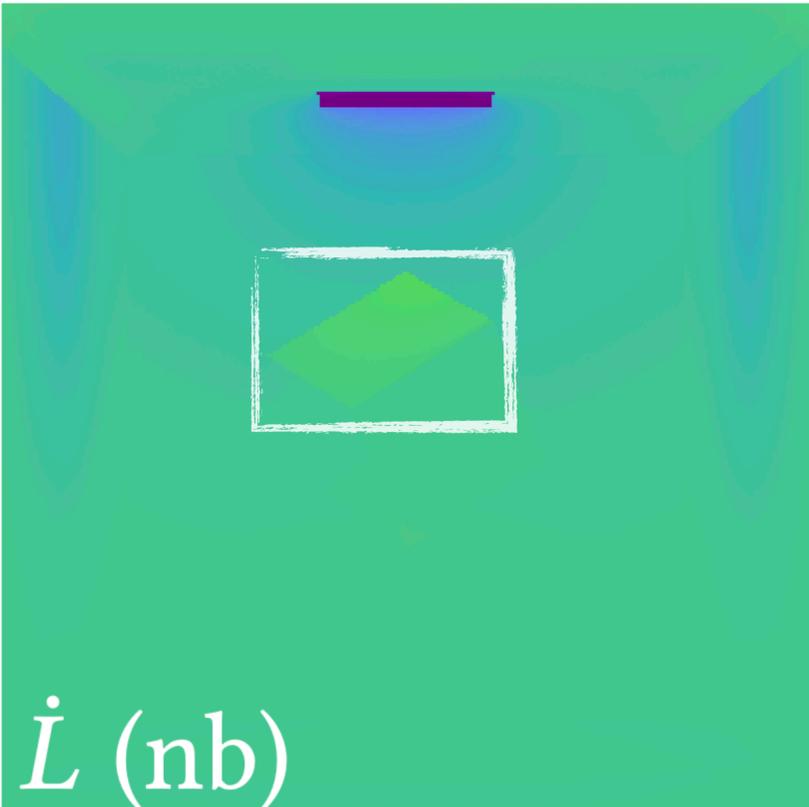
Negative  Zero  Positive



Original image



Derivative image
w.r.t. vertical offset of
the area light and the cube



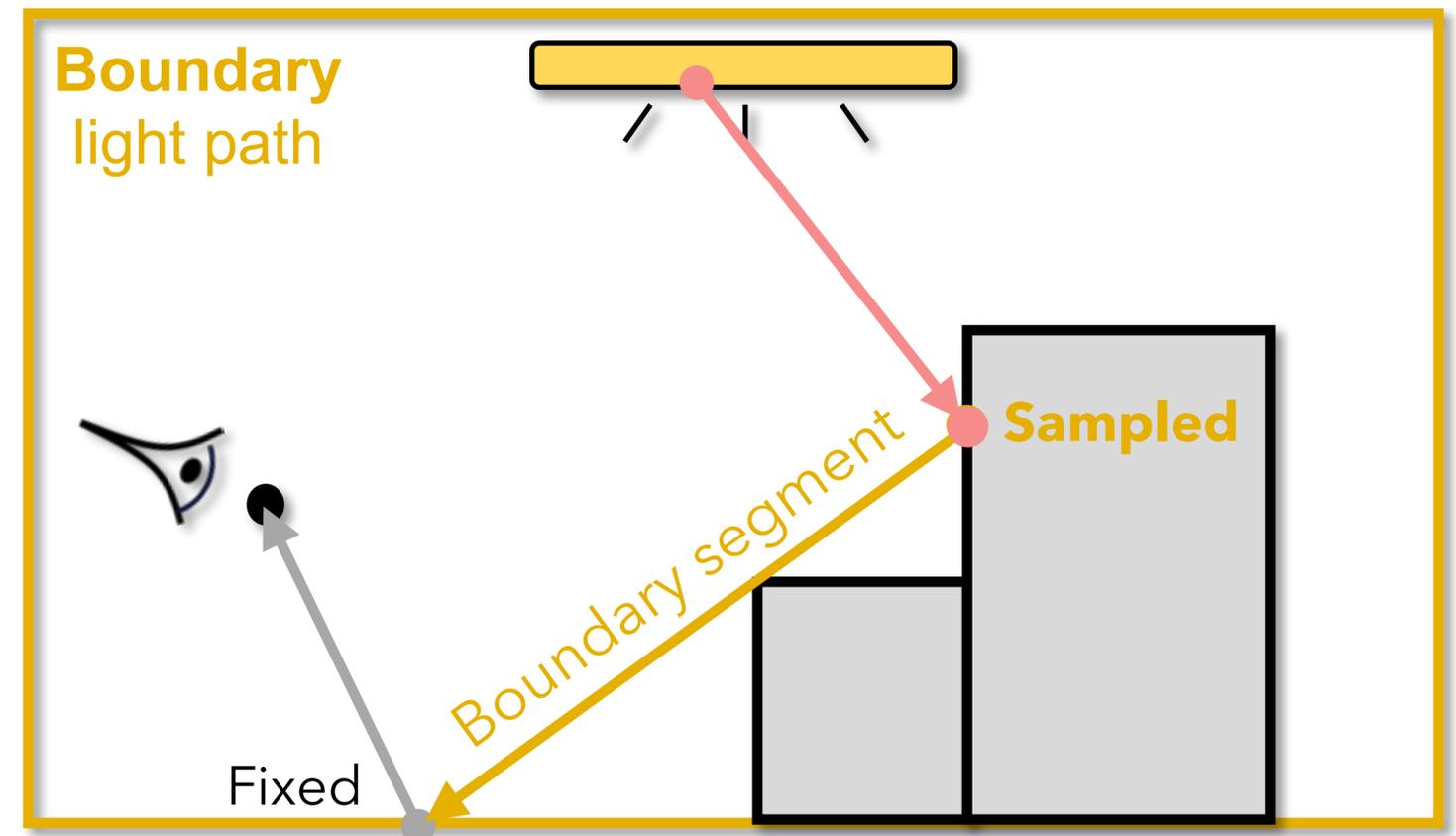
Derivative image
w/o boundary integral

Handling Discontinuities

- **Objective:** estimating the integral over all **boundary** light paths (that are the same as an **original one** except having exactly one **boundary segment**)
- (Solution 1) Monte Carlo **edge sampling**
 - Introduced by Li et al. [2018]
 - Also used by Zhang et al. [2019]

To sample a **boundary segment**:

- Fix one endpoint
- Sample the other from **discontinuity boundaries**



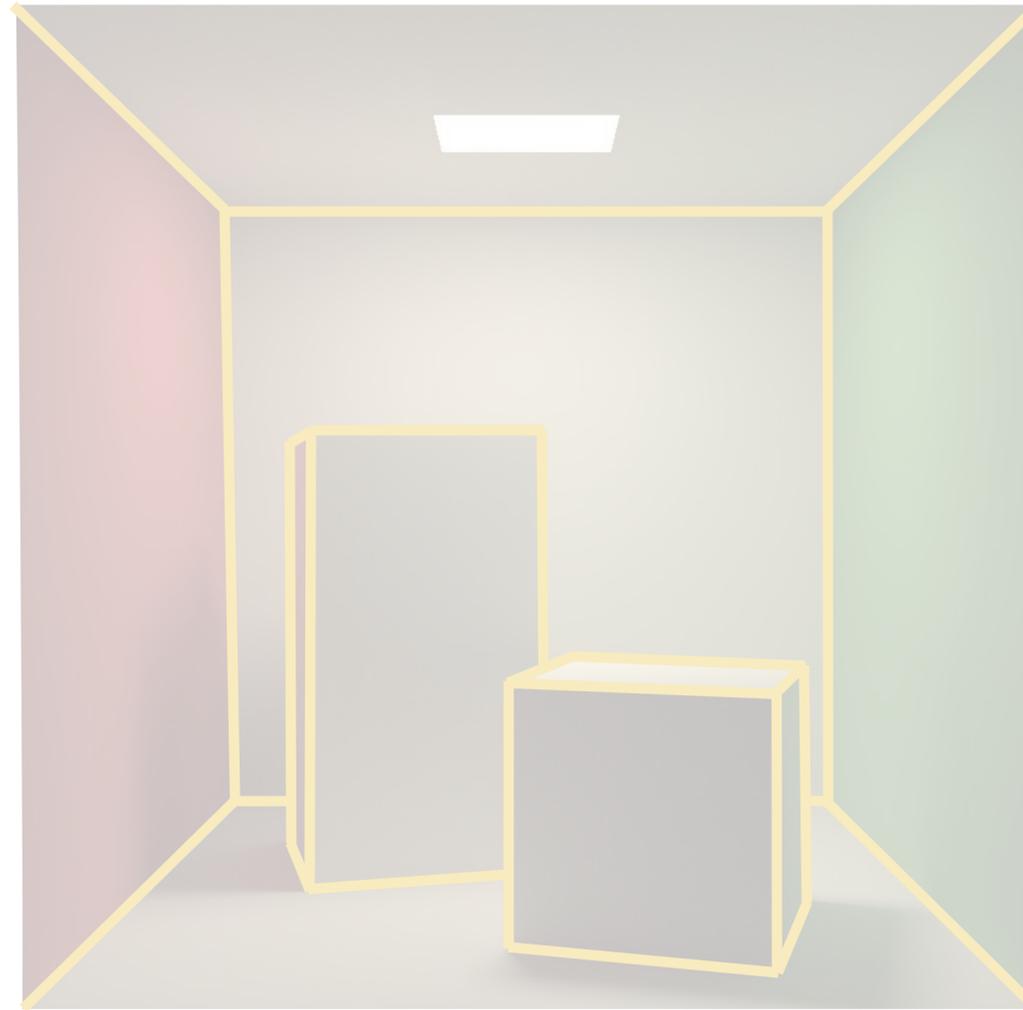
Recap: Sources of Discontinuities

Boundary edges



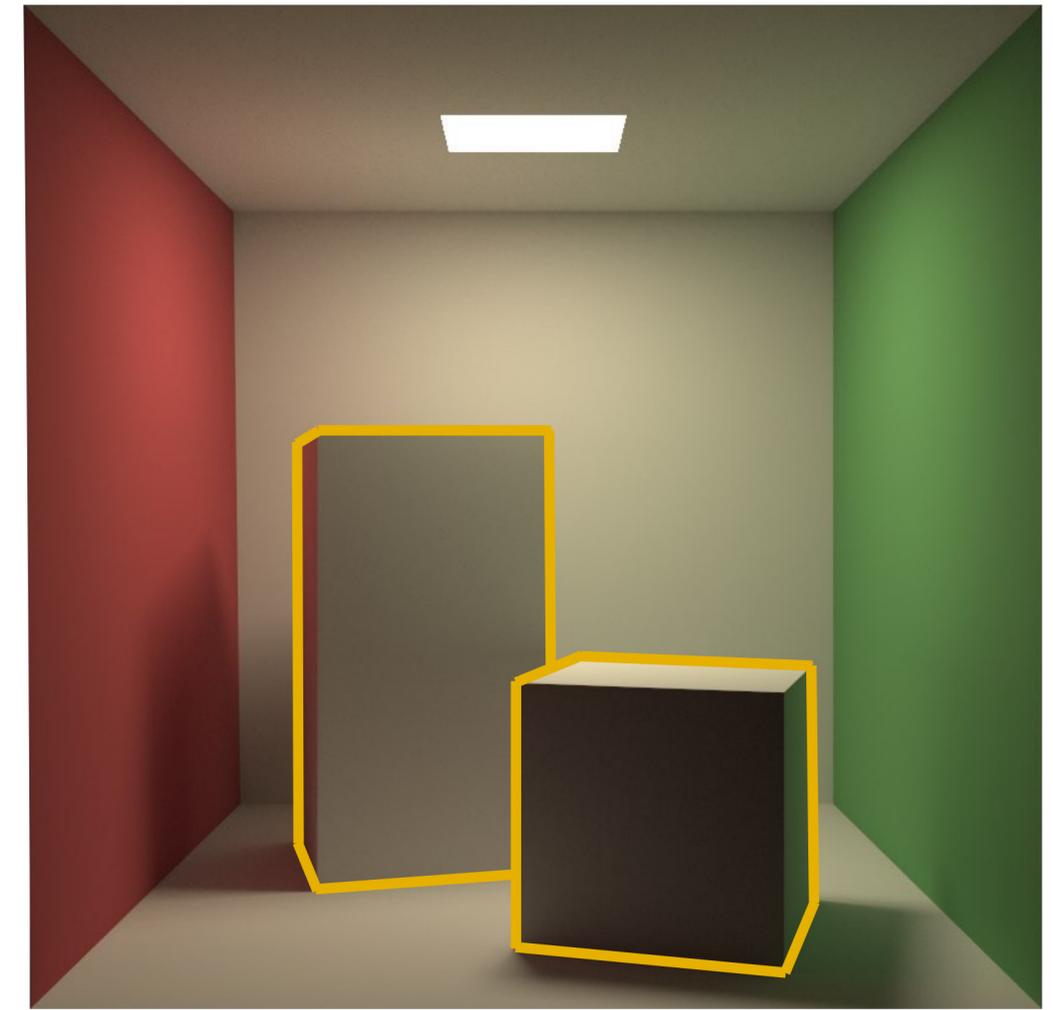
(Topological) boundary of an object

Sharp edges



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

Silhouette edges



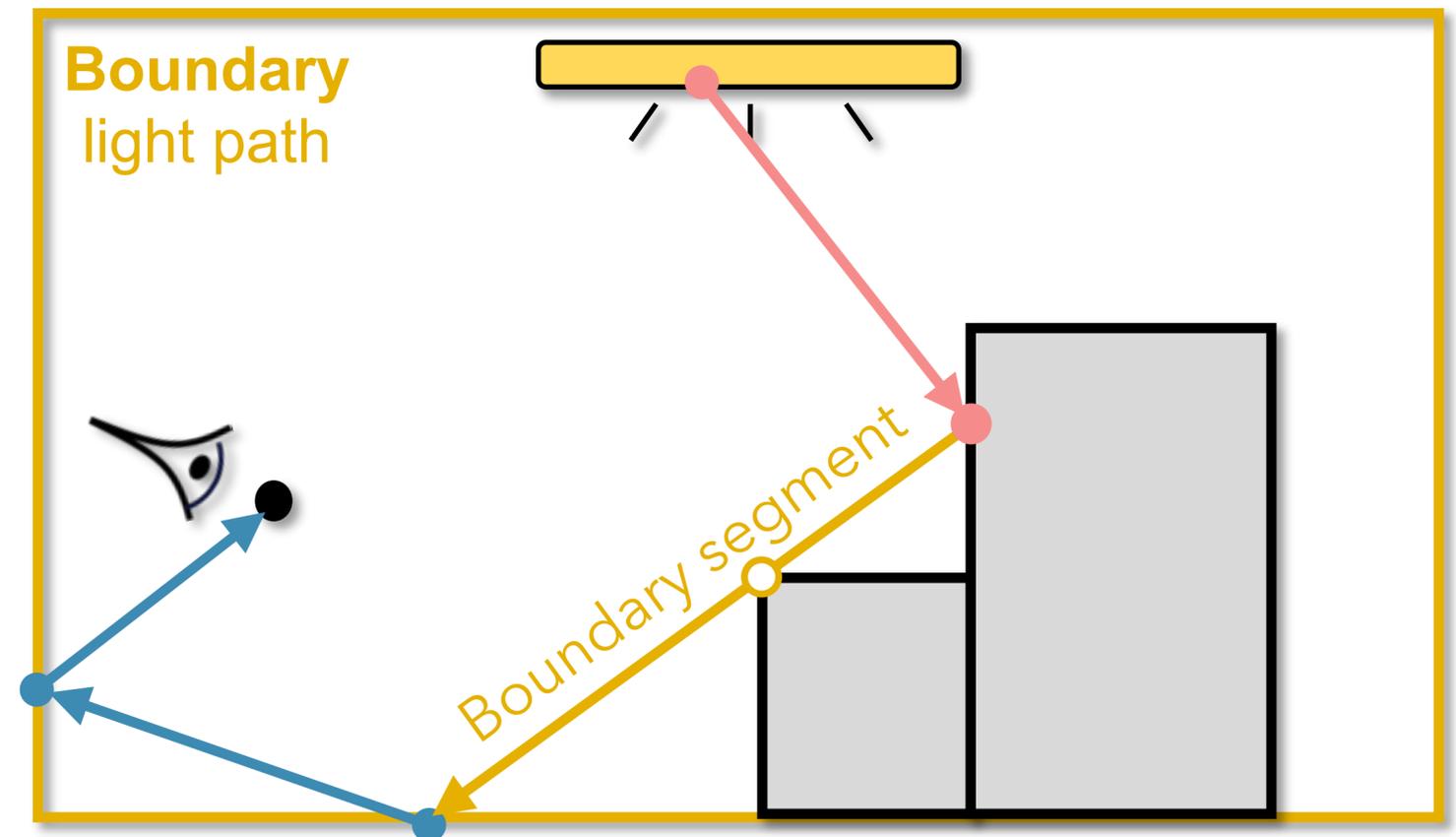
View-dependent object silhouettes

Handling Discontinuities

- **Objective:** estimating the integral over all **boundary light paths** (that are the same as an **original one** except having exactly one **boundary segment**)
- (Solution 2) **multi-directional sampling** of **boundary paths**
 - Enabled by the path-integral formulation [Zhang et al. 2020, 2021]

To sample a **boundary path**:

- Start from the **boundary segment** in the middle
- Then construct the **source** and **sensor** subpaths



Physics-Based Differentiable Rendering Algorithms

- **Boundary-sampling differentiable** rendering
 - Path tracing with **edge sampling** [Li et al. 2018, Zhang et al. 2019] ([solution 1](#))
 - **Path-space differentiable** rendering [Zhang et al. 2020, 2021] ([solution 2](#))
- **Area-sampling differentiable** rendering
 - Avoids boundary integrals altogether (Sai will cover this later)

To be
discussed
next

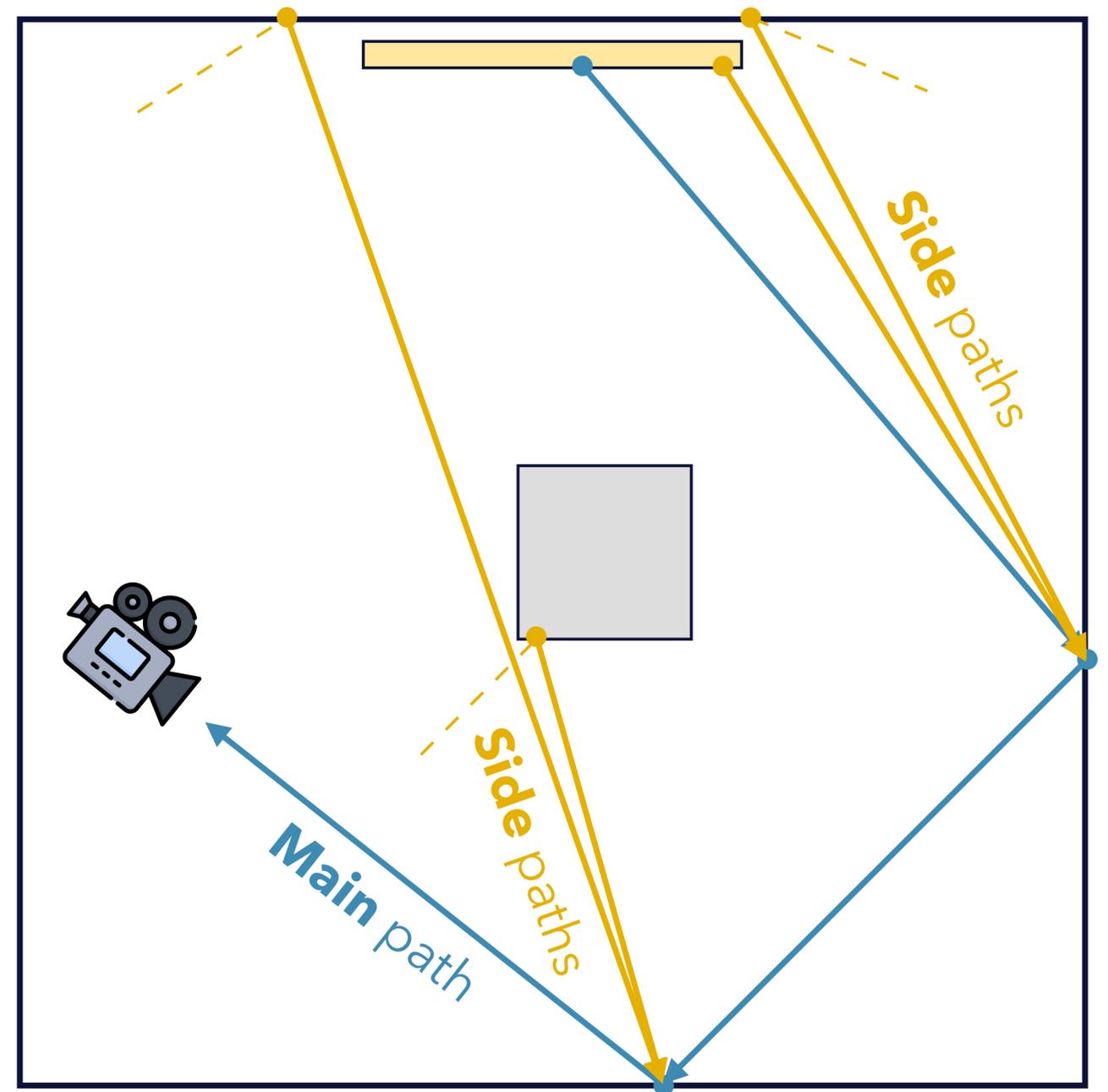
Differentiable Path Tracing with Edge Sampling

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right)$$
$$= \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

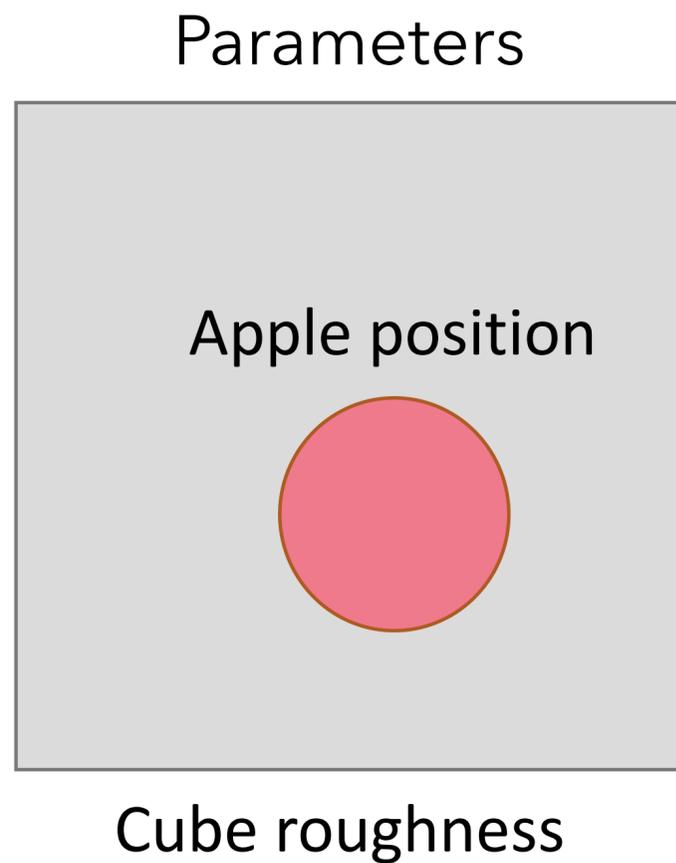
Interior integral **Boundary** integral

Differentiable path tracing with edge sampling

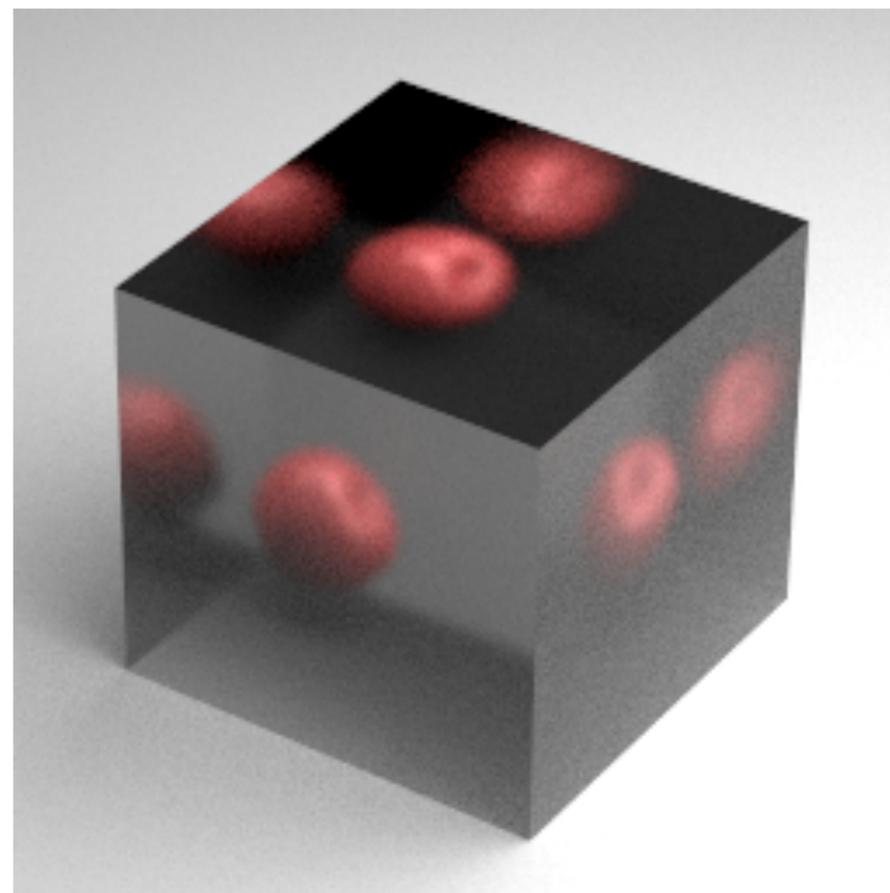
- Trace **main paths** to estimate the **interior** integral
 - Same as ordinary path tracing (for forward rendering)
- Trace additional **side paths** for the **boundary** integral
 - Each **side path** begins with a **boundary** segment (obtained with edge sampling)



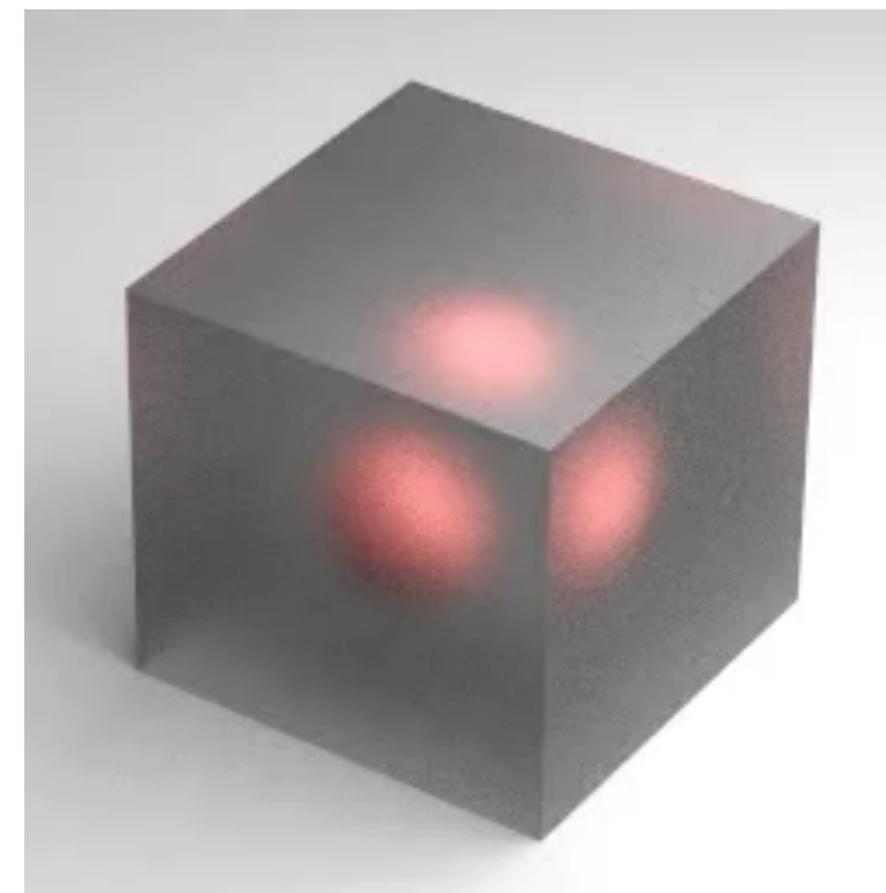
Inverse-Rendering Result [Zhang et al. 2019]



Target



Optimization process



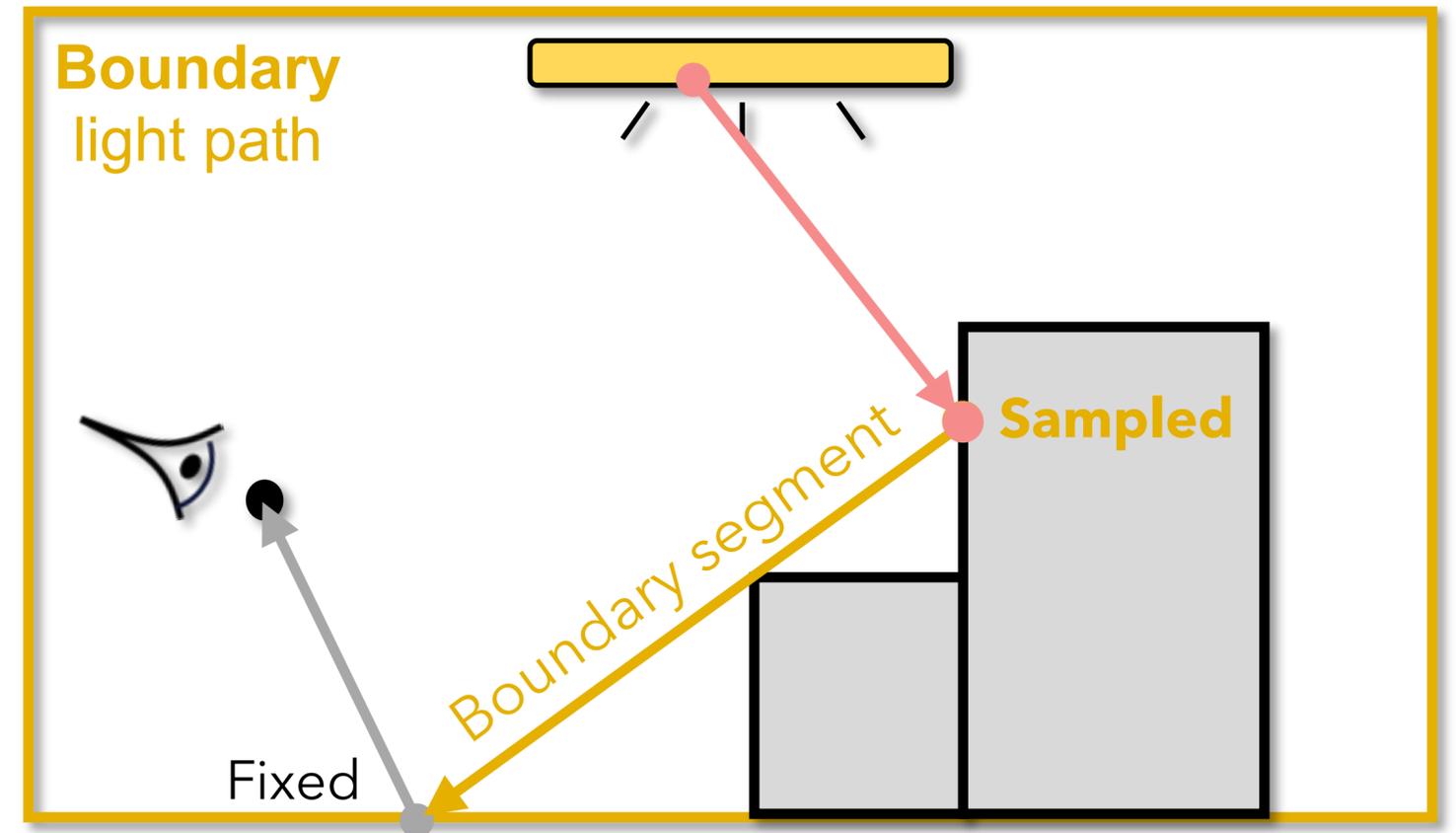
Light-transport phenomena:

rough reflection and refraction
subsurface scattering

Differentiable Path Tracing with Edge Sampling

To sample a **boundary** segment:

- Fix one endpoint
- Sample the other from **discontinuity boundaries**



Requires **silhouette detection**, which can be **expensive!**

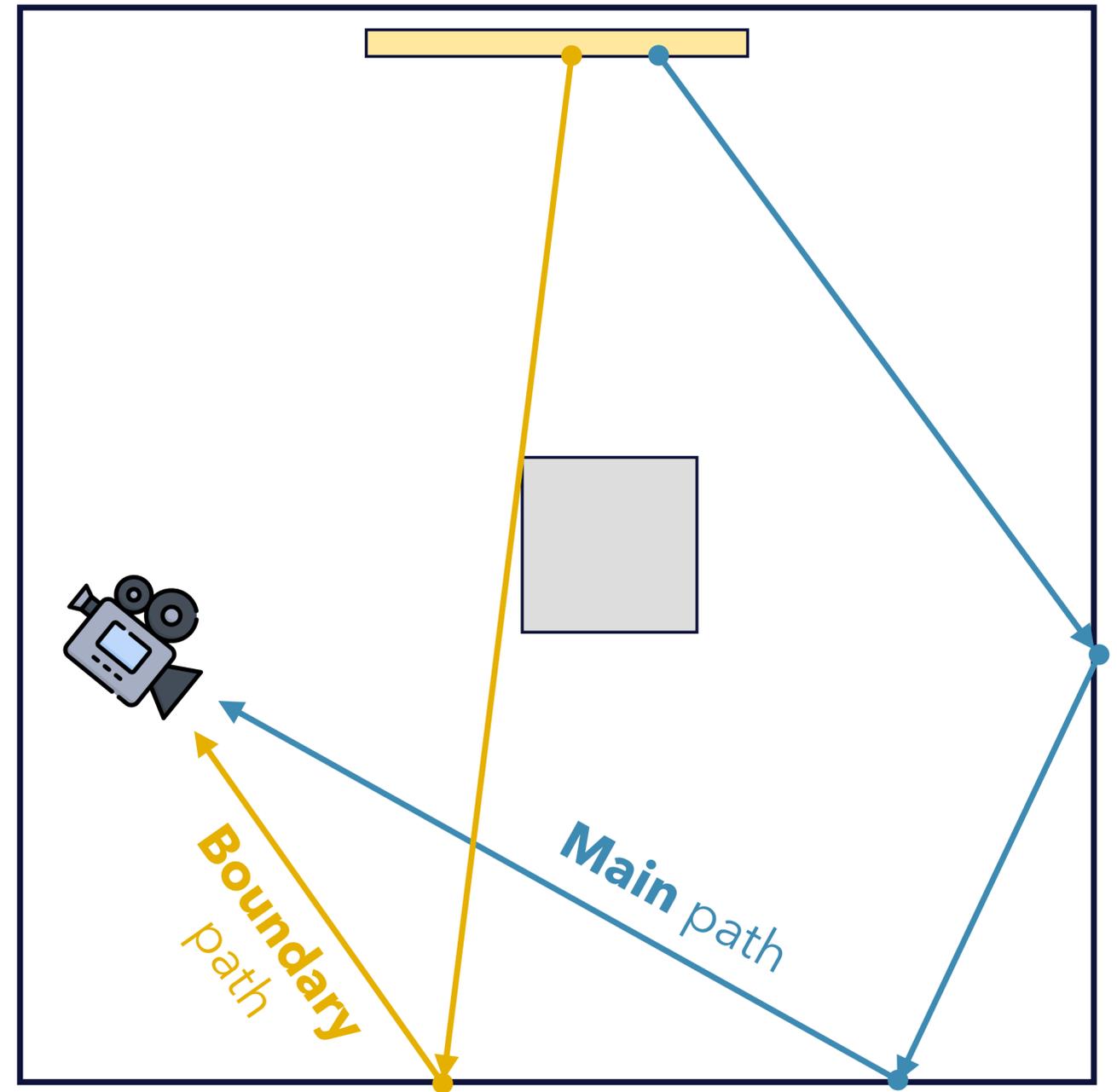
Path-Space Differentiable Path Tracing

$$\frac{d}{d\theta} \left(\int_{\Omega} f(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) \right)$$
$$= \int_{\Omega} \dot{f}(\bar{\mathbf{x}}) d\mu(\bar{\mathbf{x}}) + \int_{\partial\Omega} g(\bar{\mathbf{x}}) d\mu'(\bar{\mathbf{x}})$$

Interior integral **Boundary** integral

Path-space **differentiable** path tracing

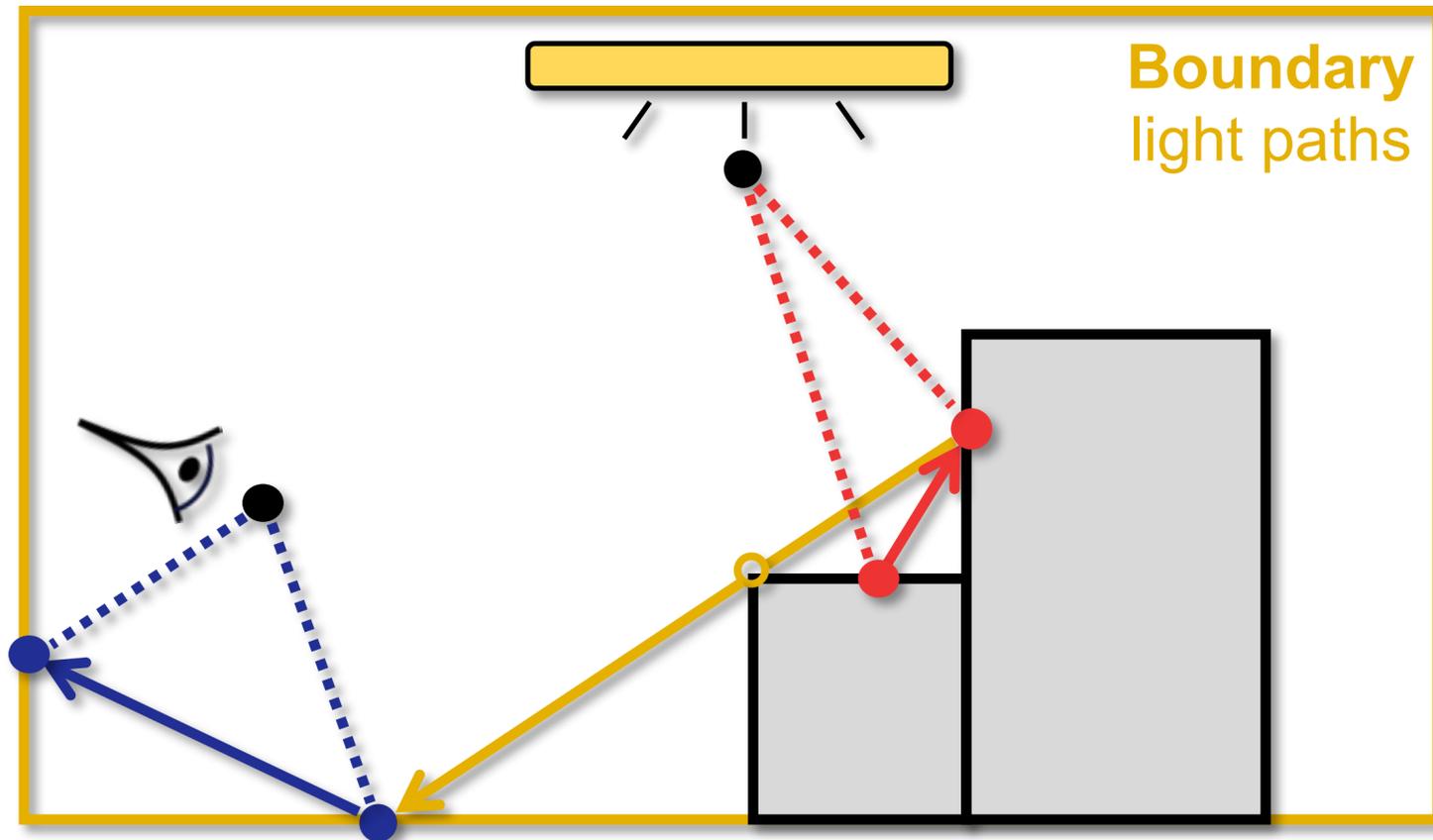
- Trace **main** paths to estimate the **interior** integral
 - Same as forward rendering
- Trace additional **boundary** paths for the **boundary** integral separately (using multi-directional sampling)



Path-Space Differentiable Path Tracing

Unidirectional estimator

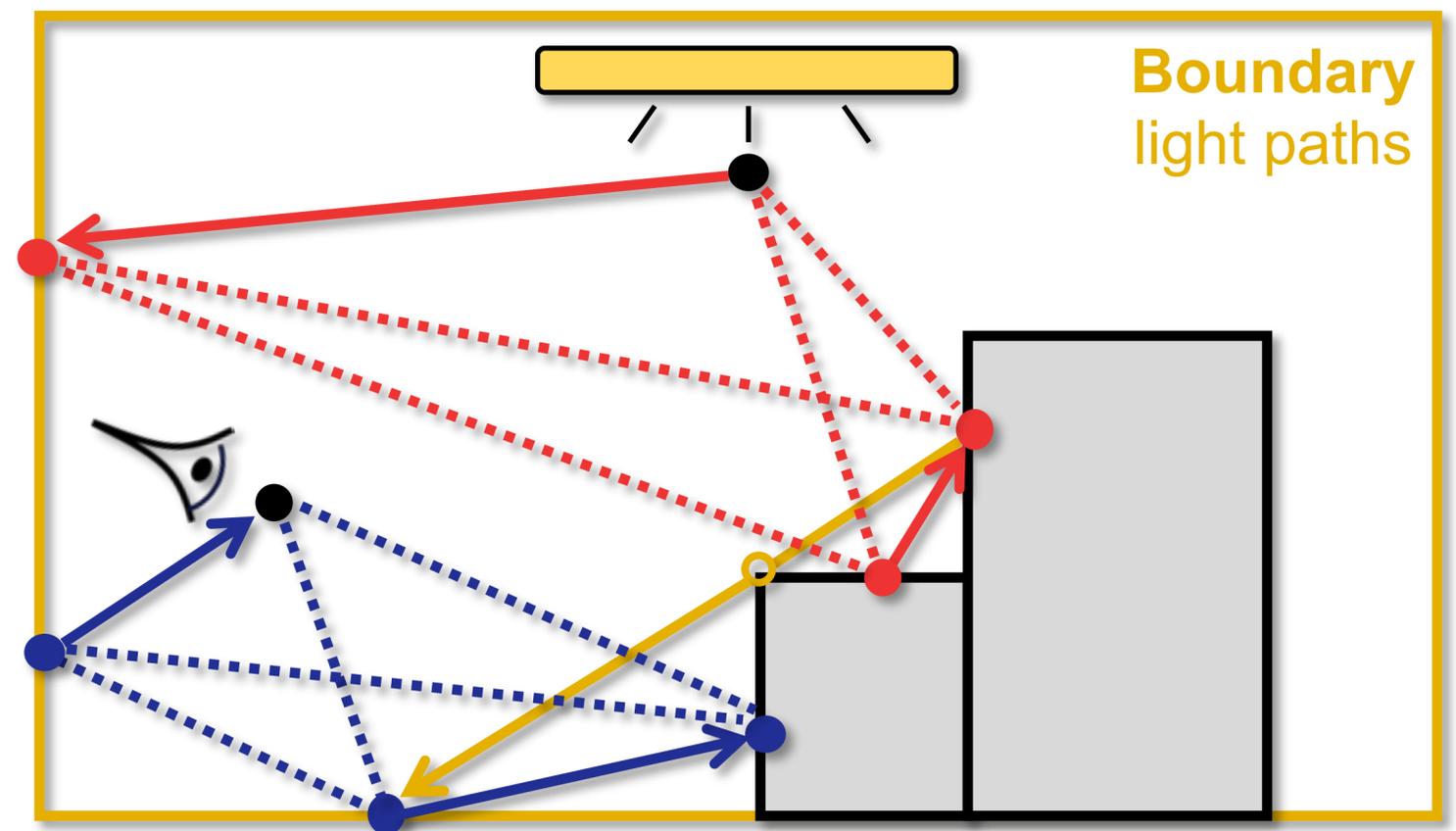
- **Interior**: *unidirectional* path tracing
- **Boundary**: *unidirectional* sampling of subpaths



Unidirectional path tracing + NEE

Bidirectional estimator

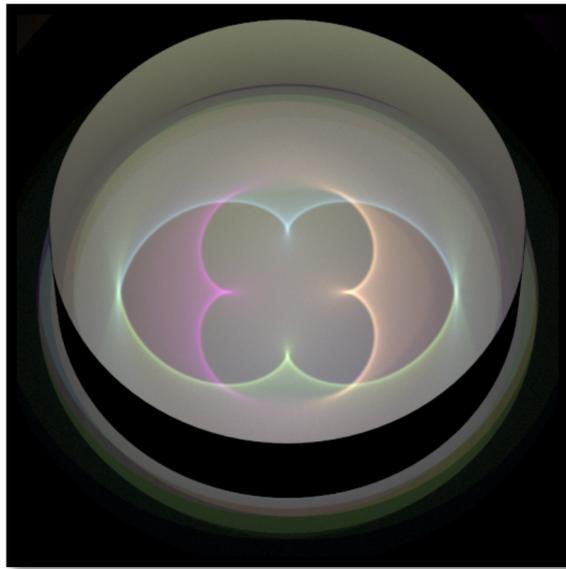
- **Interior**: *bidirectional* path tracing
- **Boundary**: *bidirectional* sampling of subpaths



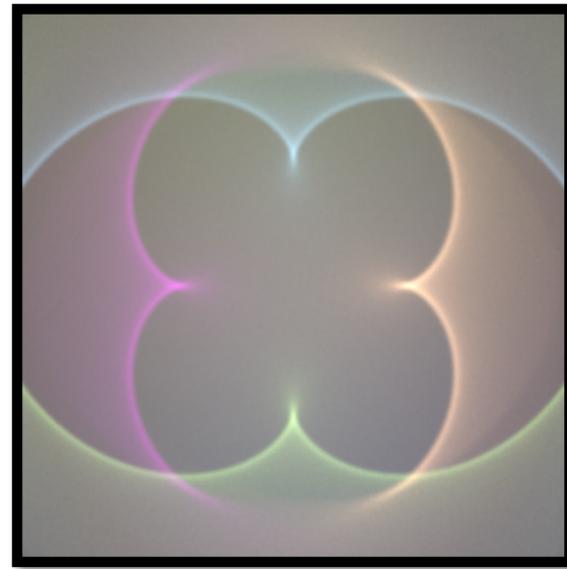
Bidirectional path tracing

Inverse-Rendering Result [Zhang et al. 2020]

Config.



Initial



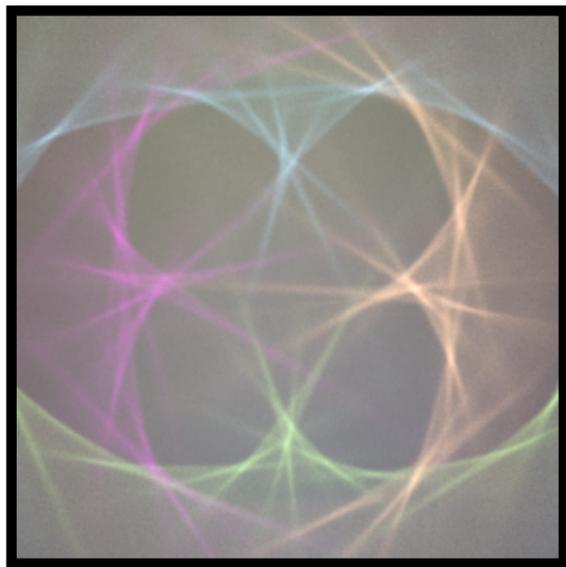
Scene configuration:

- A glossy ring lit by four colored light sources
- Optimize **cross-sectional shape** of the ring

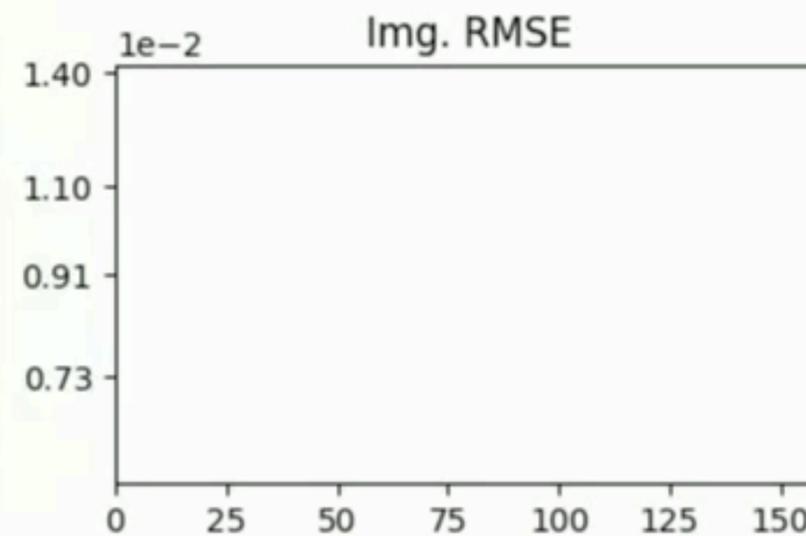
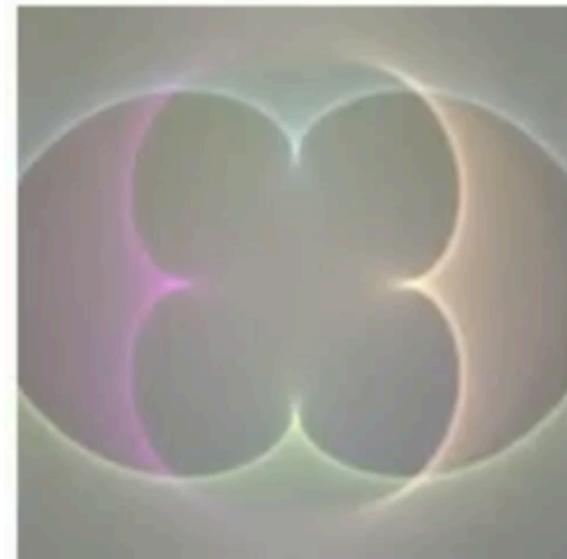
Light-transport phenomenon:

- Caustics

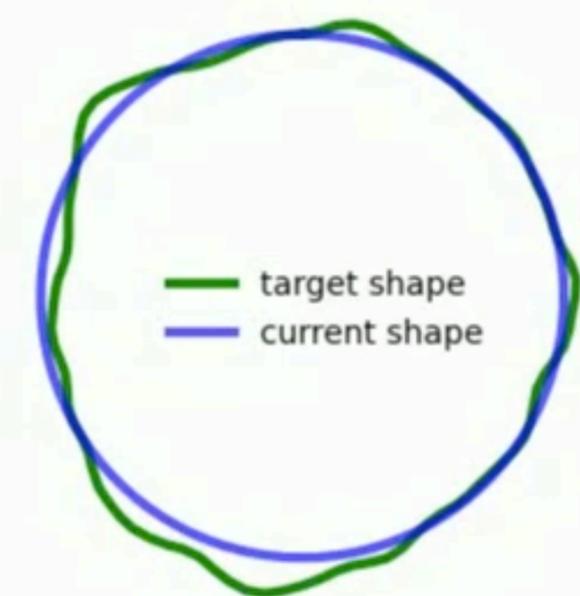
Target



Iter #0

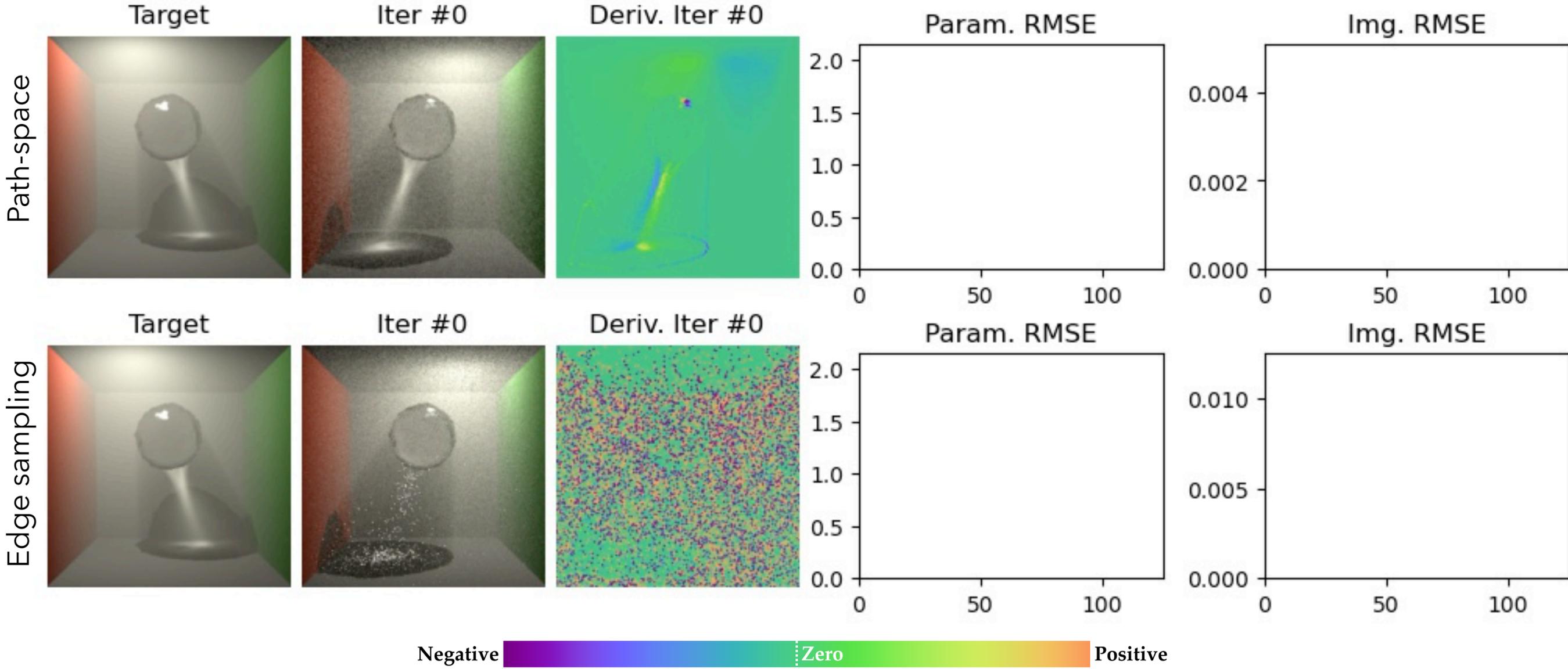


Cross-sectional shape
(displacement x 20)



Inverse-Rendering Comparison [Zhang et al. 2021]

Optimizing the position of a small area light
(identical inverse-rendering configurations, equal-time per iteration)

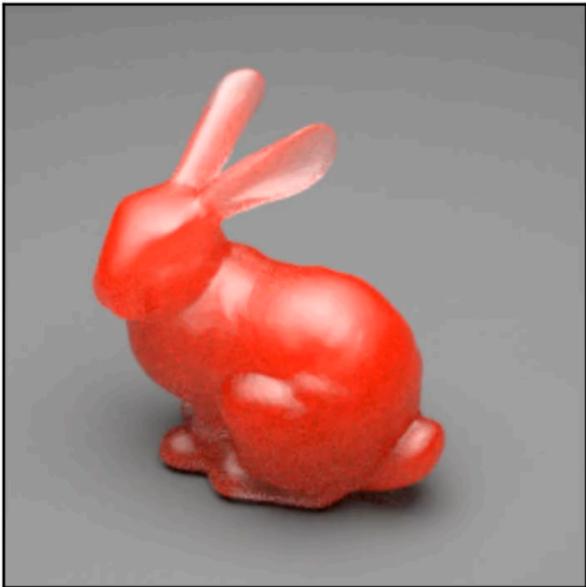


Inverse-Rendering Result [Zhang et al. 2021]

Initial



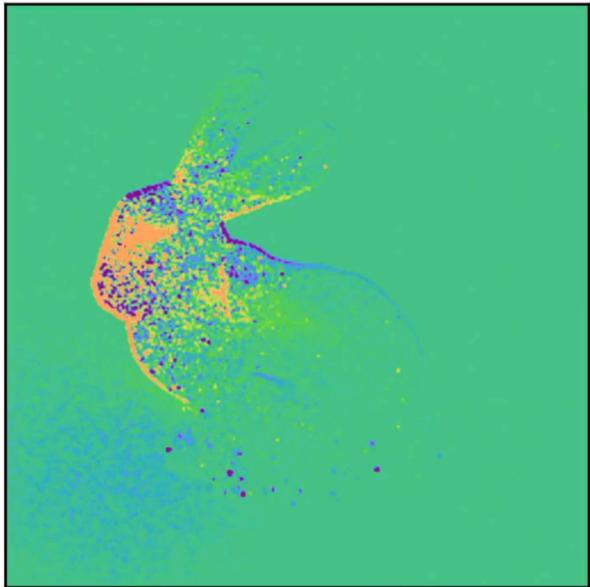
Iter #0



Target



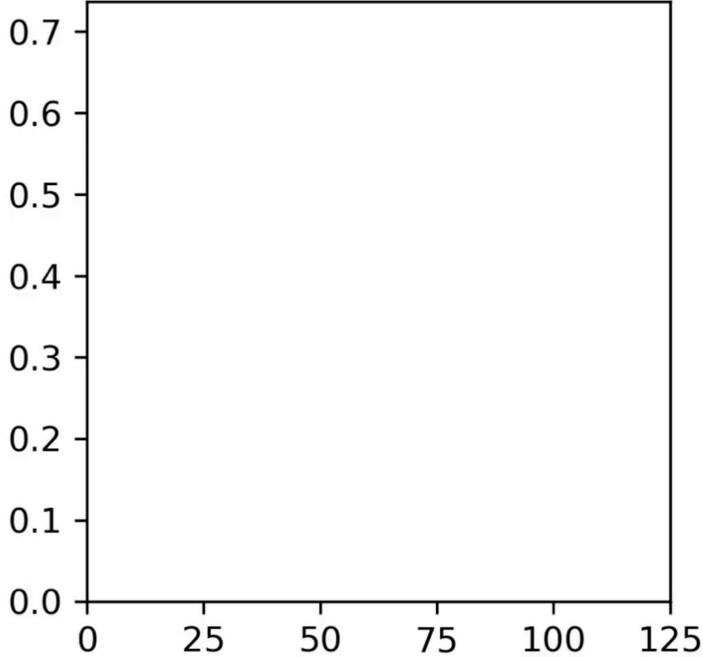
Deriv. Iter #0



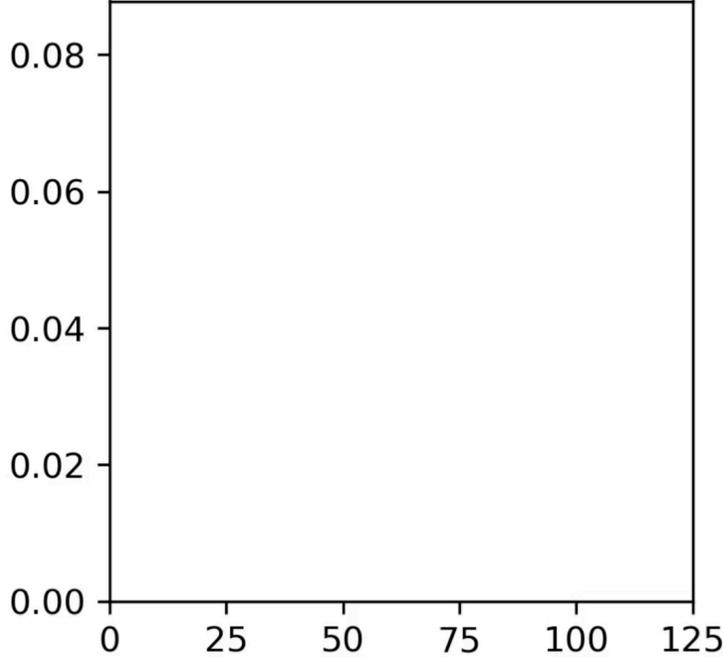
Jointly optimizing of the bunny's:

- Shape
- Surface roughness
- Optical density

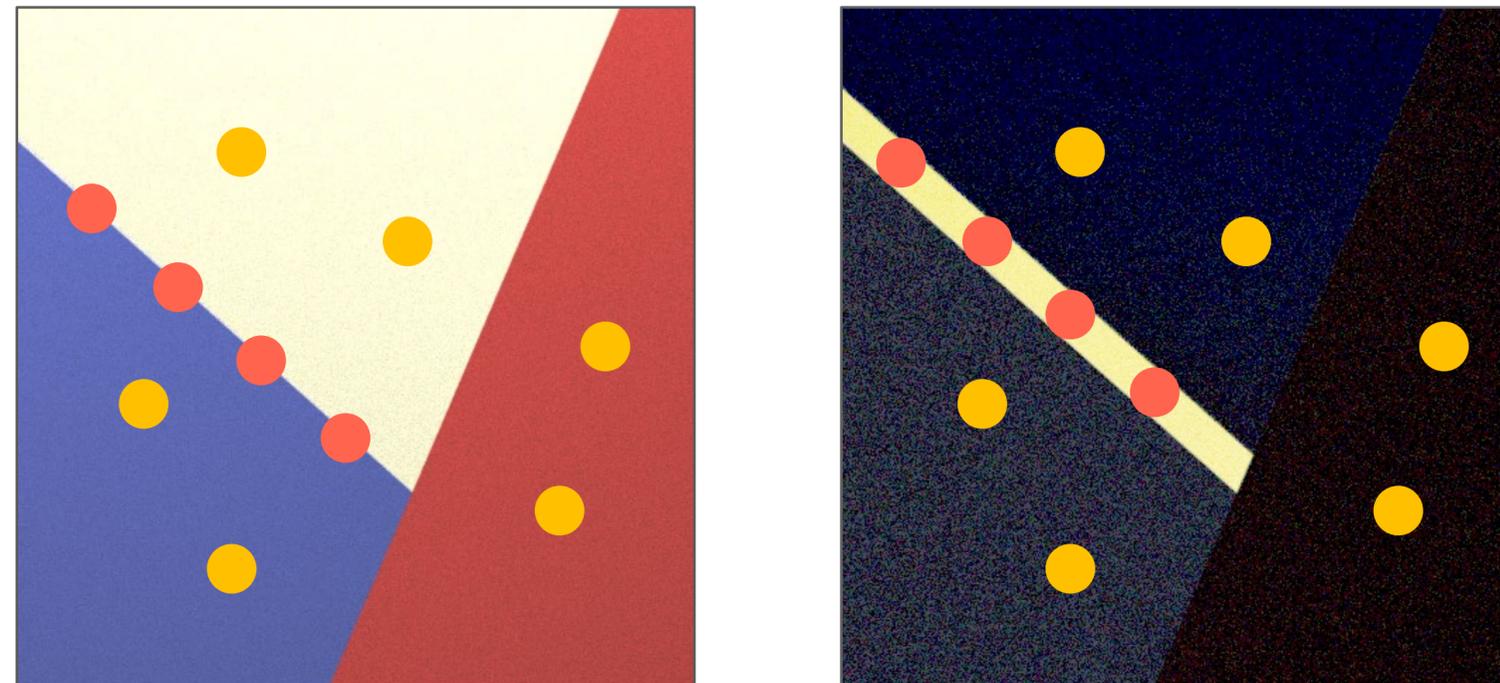
Param. RMSE



Img. RMSE



Boundary methods can run into some problems

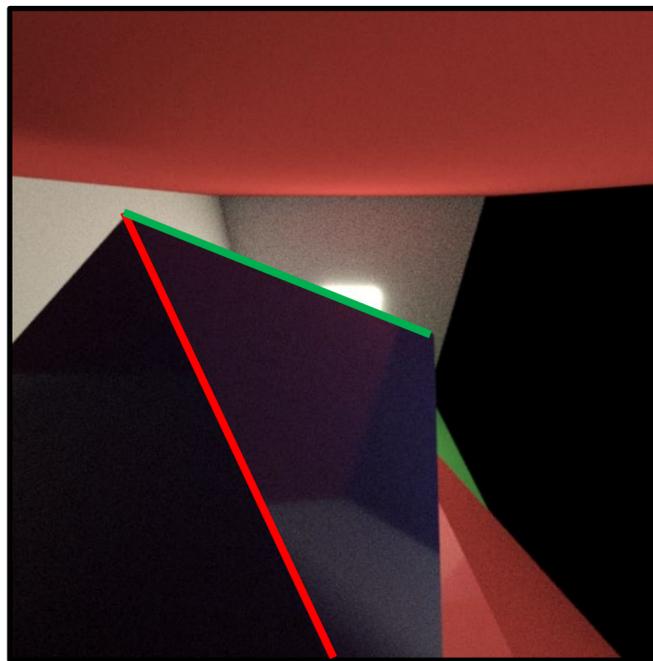


● : Boundary samples

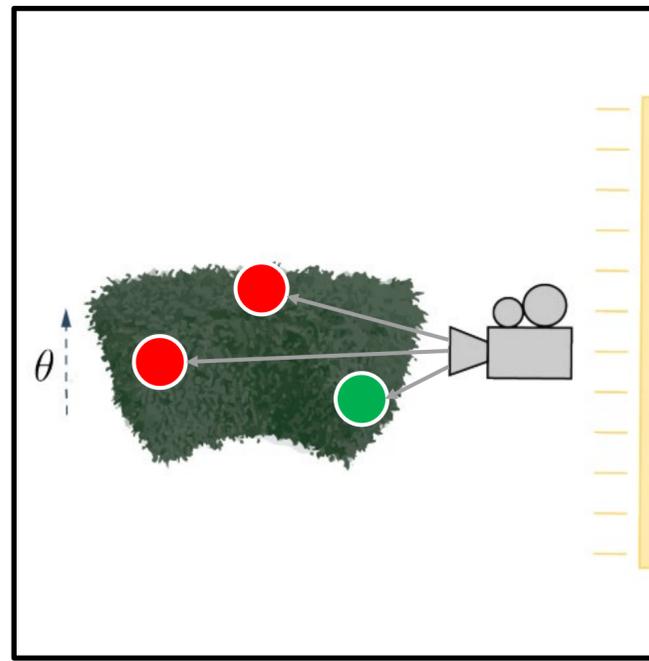
● : Area samples

Boundary methods can run into some problems

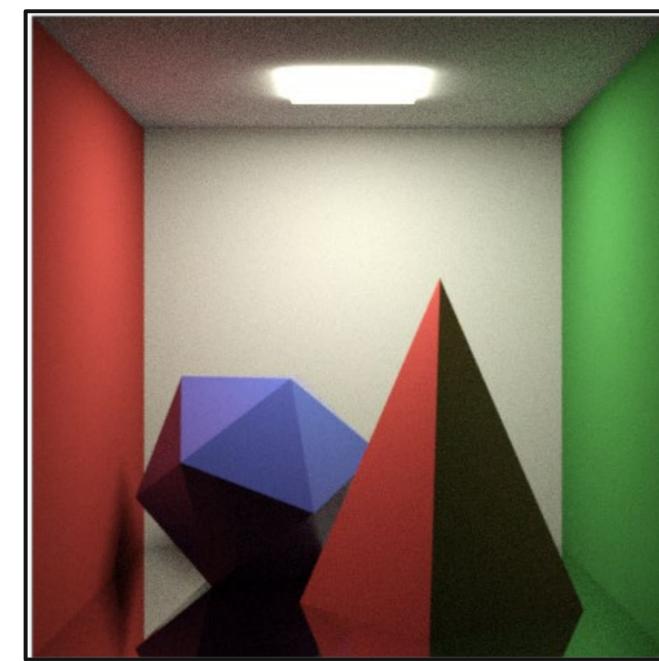
Some challenges faced



Silhouette sampling



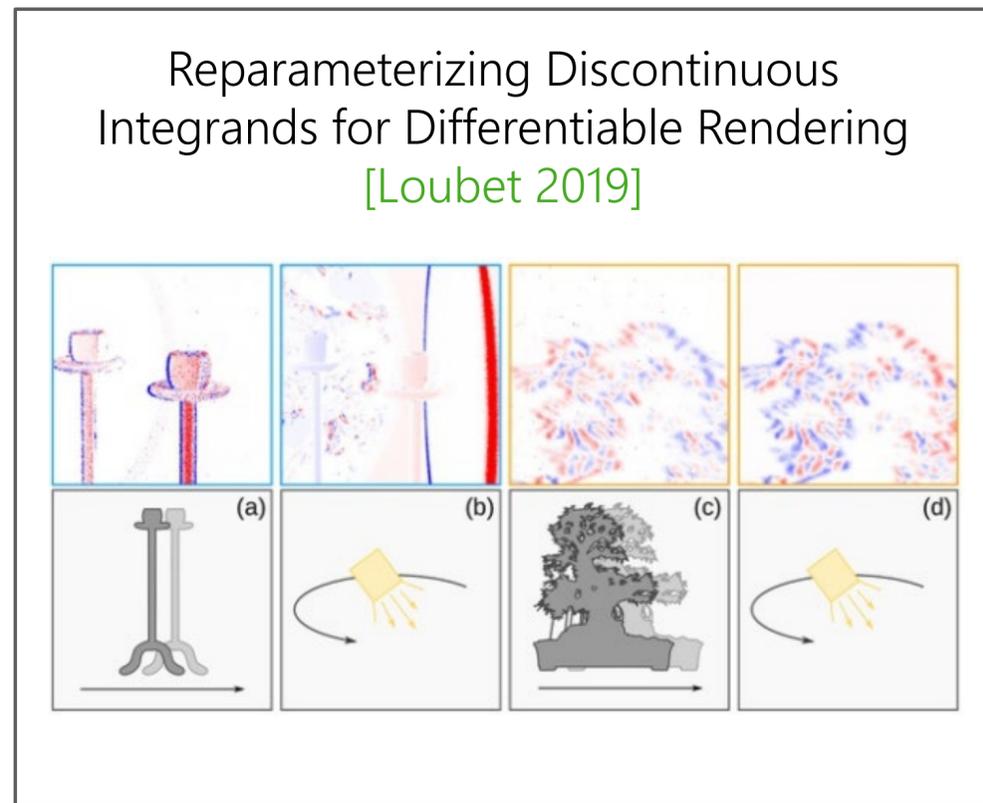
Depth complexity



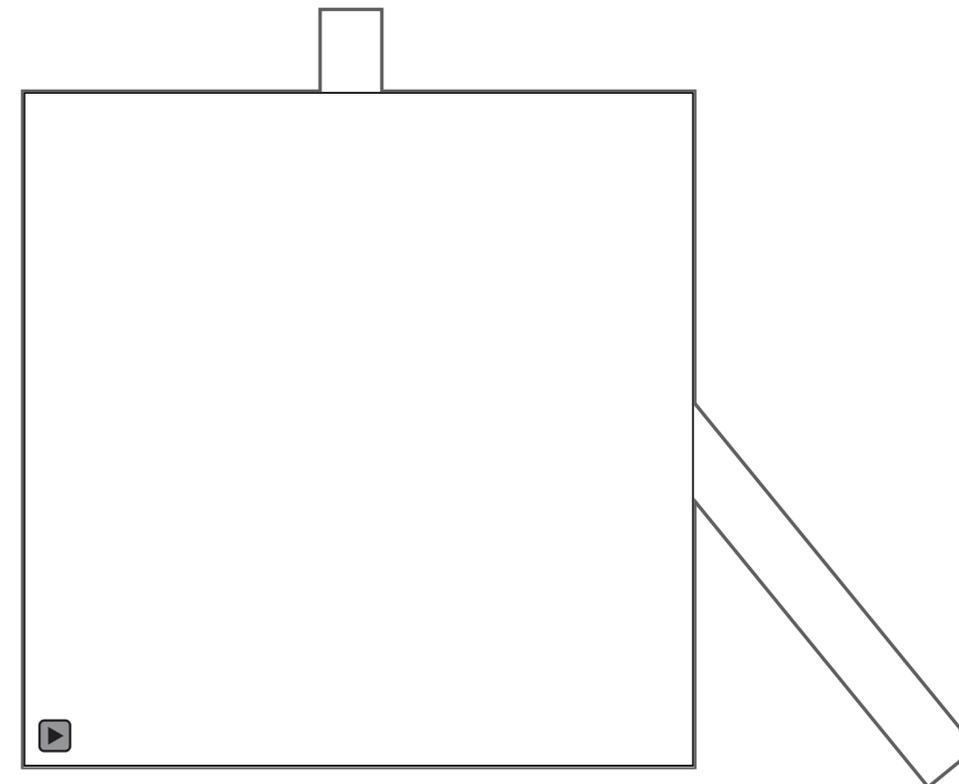
Perfect specularities

Integrating over arbitrary discontinuities can be tricky

Avoid discontinuities through reparameterization



Transform the space with θ .
"Cancels" discontinuities.



Heuristic Approximation!
May not work for all samples.

Unbiased Warped-Area Sampling for Differentiable Rendering

SAI PRAVEEN BANGARU, Massachusetts Institute of Technology
TZU-MAO LI, Massachusetts Institute of Technology
FRÉDO DURAND, Massachusetts Institute of Technology

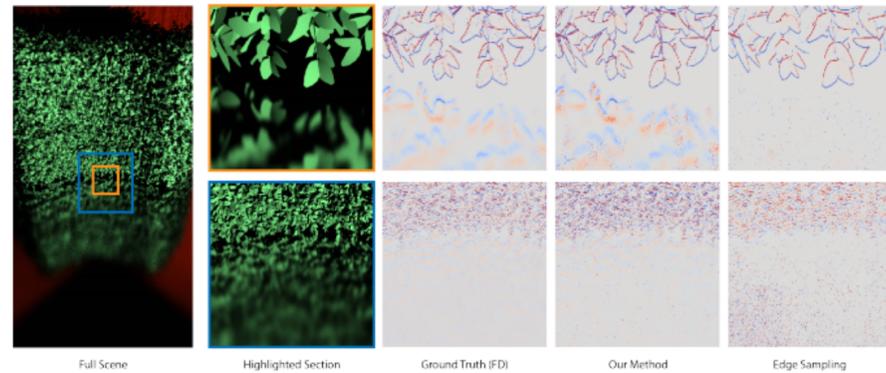


Fig. 1. Differentiable rendering computes derivatives of the light transport equation. To differentiate with the existence of visibility, recent physically-based differentiable renderers require either explicitly finding boundary points [Li et al. 2018; Zhang et al. 2020], or approximating the boundary contribution through heuristics [Loubet et al. 2019]. We develop from first principles an unbiased estimator that computes the boundary contribution from interior (area) samples. Our approach can be easily integrated with existing importance sampling methods and computes accurate and low variance gradients. For instance, the *edge sampling method* [Li et al. 2018] finds it difficult to consistently sample boundary points that contribute to the derivative in the soft reflection, especially because of the high complexity of the scene. Our method, on the other hand, uses samples from a standard path tracer and takes advantage of BSDF and light source importance sampling to compute a robust estimate for the derivative. We validate our derivatives against the finite difference image computed w.r.t the hedge’s translation in the upward direction. Both our method and edge sampling used an equal number of samples.

Differentiable rendering computes derivatives of the light transport equation with respect to arbitrary 3D scene parameters, and enables various applications in inverse rendering and machine learning. We present an unbiased and efficient differentiable rendering algorithm that does not require explicit boundary sampling. We apply the divergence theorem to the derivative of the rendering integral to convert the boundary integral into an area integral. We rewrite the converted area integral to a form that is suitable for Monte Carlo rendering. We then develop an efficient Monte Carlo sampling algorithm for solving the area integral. Our method can be easily plugged into a traditional path tracer and does not require dedicated data structures for sampling boundaries.

Authors’ addresses: Sai Praveen Bangaru, Massachusetts Institute of Technology, Cambridge, MA, sbangaru@mit.edu; Tzu-Mao Li, Massachusetts Institute of Technology, Cambridge, MA, trumao@mit.edu; Frédo Durand, Massachusetts Institute of Technology, Cambridge, MA, fredod@mit.edu.

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2020 Copyright held by the owner/author(s).
0730-0301/2020/12-ART245
<https://doi.org/10.1145/3414685.3417833>

We analyze the convergence properties through bias-variance metrics, and demonstrate our estimator’s advantages over existing methods for some synthetic inverse rendering examples.

CCS Concepts: • **Computing methodologies** → *Computer vision; Rendering; Visibility.*

Additional Key Words and Phrases: inverse graphics, differentiable rendering, light transport, differentiating visibility

ACM Reference Format:

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 245 (December 2020), 18 pages. <https://doi.org/10.1145/3414685.3417833>

1 INTRODUCTION

Differentiable rendering – the task of computing derivatives of the light transport equation [Kajiya 1986] with respect to scene parameters such as camera position, triangle mesh positions, and texture parameters, has become increasingly important for solving inverse rendering problems and training 3D deep learning models. The discontinuities introduced by visibility pose a central challenge

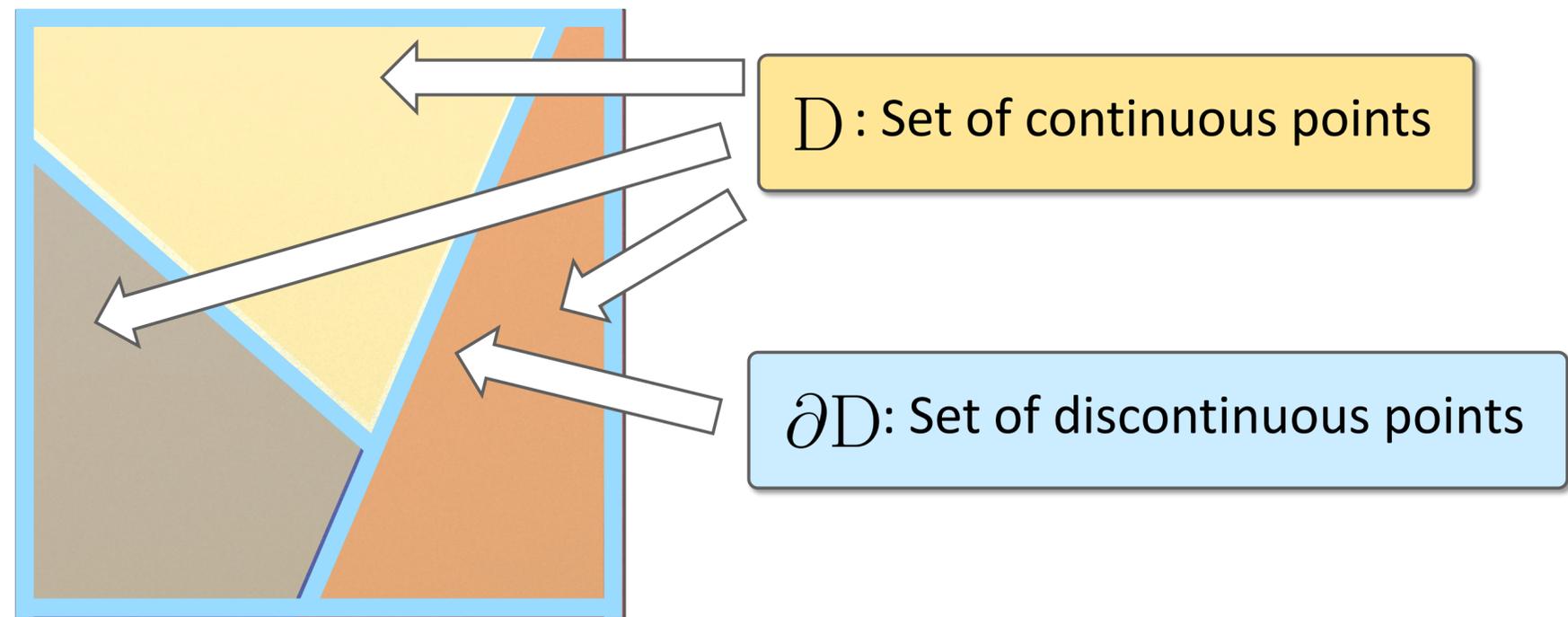
ACM Trans. Graph., Vol. 39, No. 6, Article 245. Publication date: December 2020.

Unbiased Warped-Area Sampling for Differentiable Rendering

Sai Bangaru, Tzu-Mao Li, Fredo Durand
(MIT CSAIL)

SIGGRAPH Asia 2020

The Reynolds Transport Theorem



$$\partial_{\theta} \int_{\mathcal{D}} f = \int_{\mathcal{D}} \partial_{\theta} f + \int_{\partial \mathcal{D}} f \vec{\mathbf{v}} \cdot \vec{\mathbf{n}}$$

Interior term Edge term

Converting Edge-Samples to Area-Samples

$$\int_{\partial D} f \vec{v} \cdot \vec{n}$$

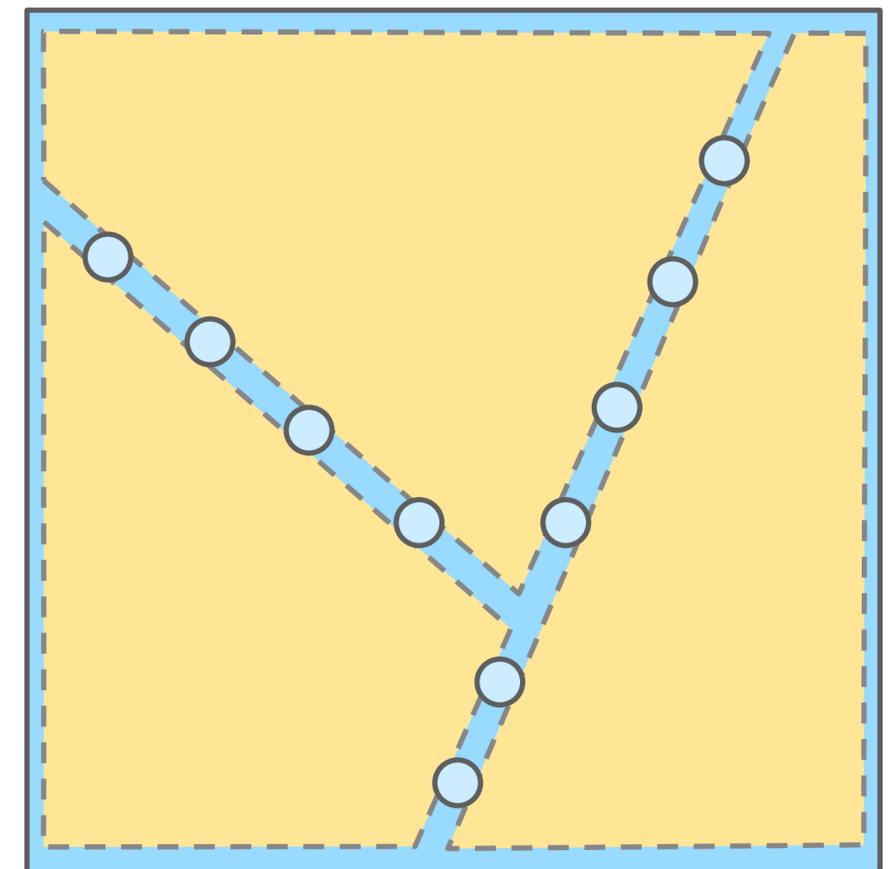
is estimated through edge-samples ○

Goal: Rewrite

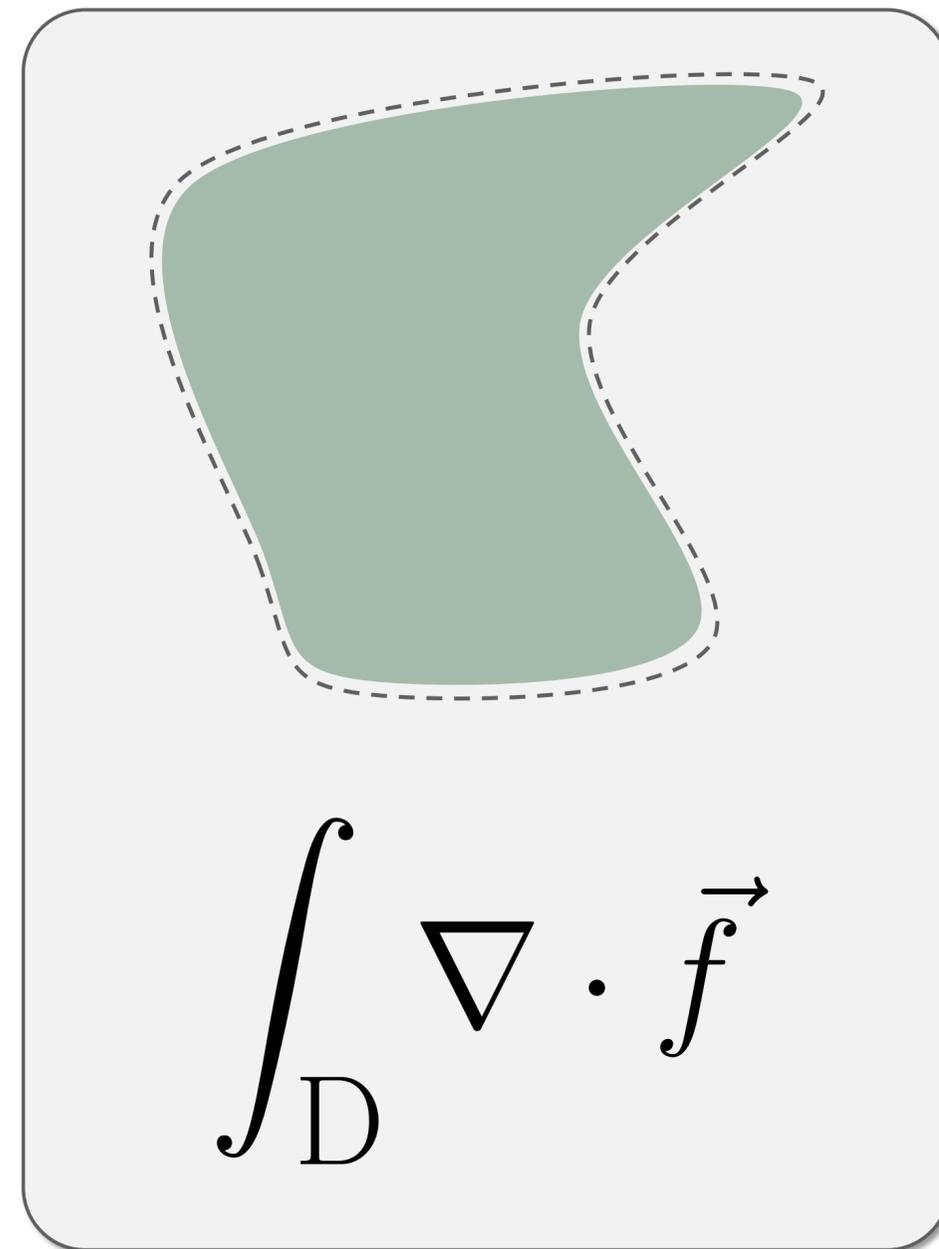
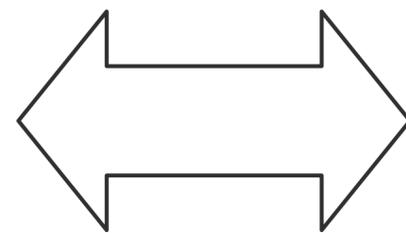
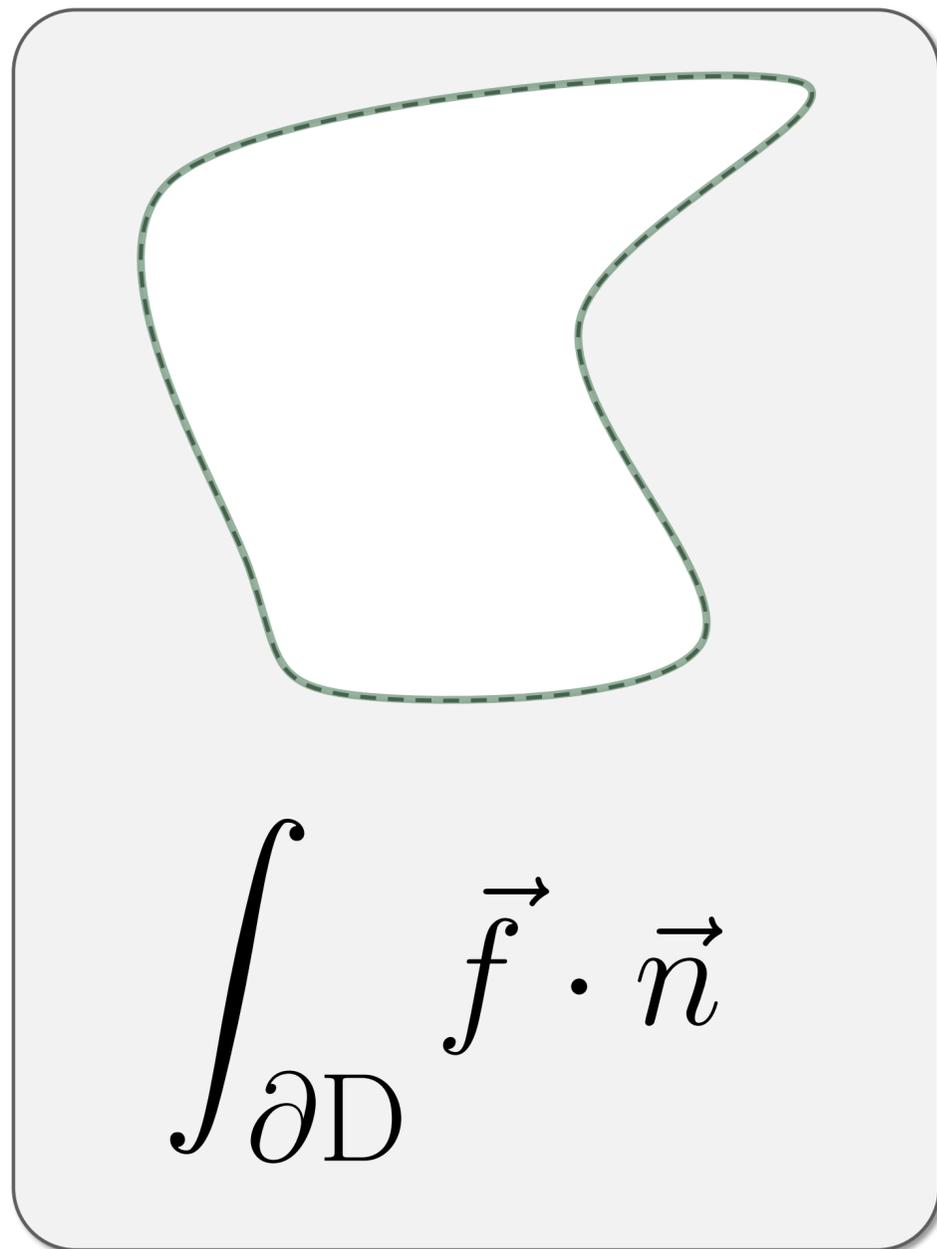
$$\int_{\partial D} f \vec{v} \cdot \vec{n}$$

into area integral

$$\int_D g$$



The Divergence Theorem [Gauss 1813]

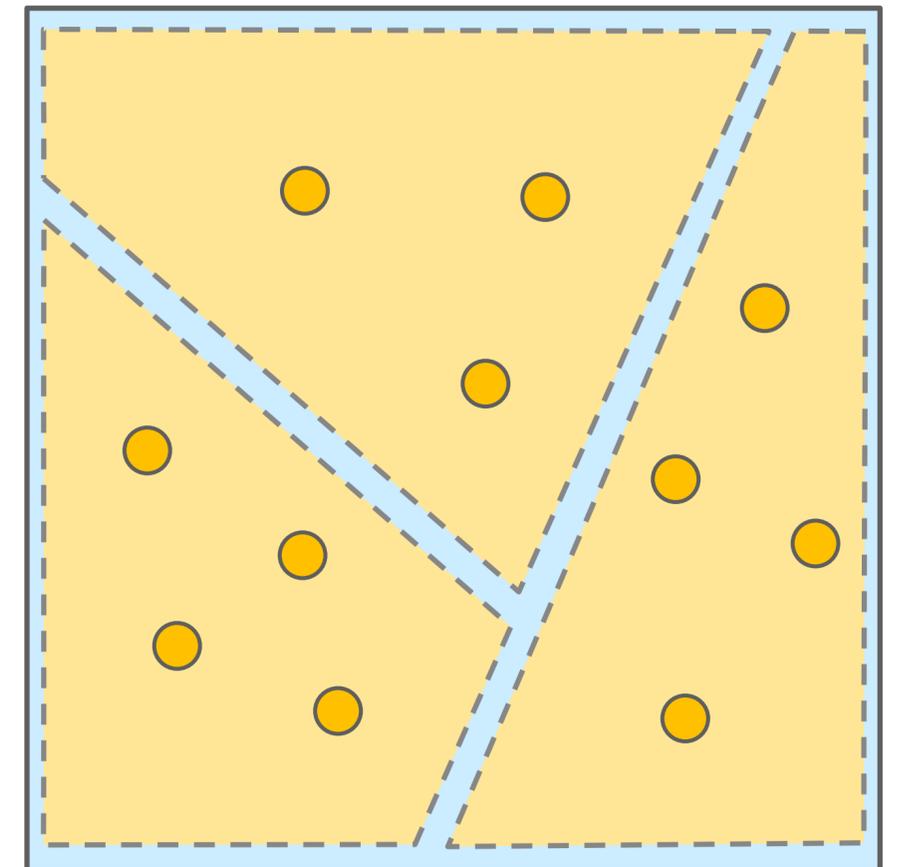


Applying the divergence theorem to the Edge Integral

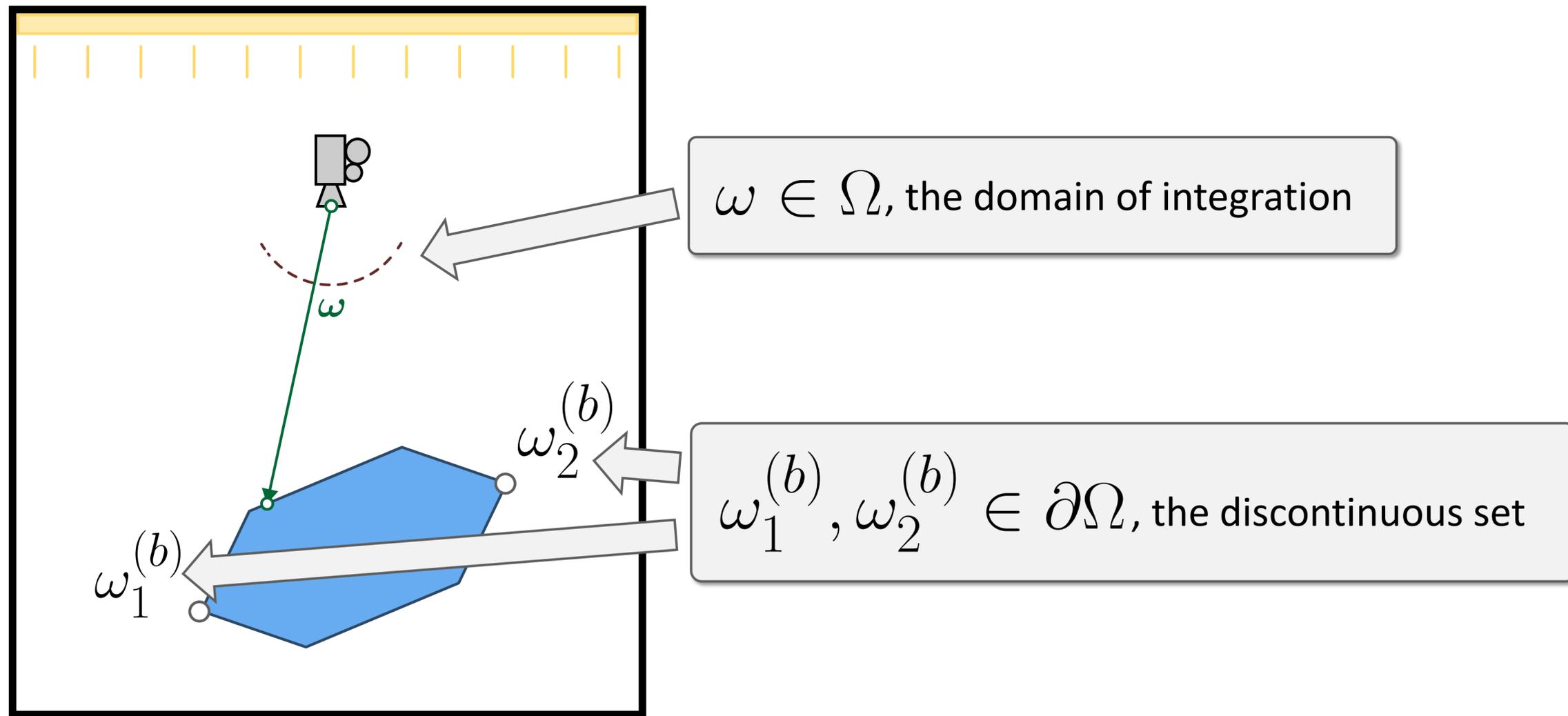
Goal: Rewrite $\int_{\partial D} f \vec{v} \cdot \vec{n}$ into area integral $\int_D g$

Solution: Rewrite $\int_{\partial D} f \vec{v} \cdot \vec{n}$ into $\int_D \nabla \cdot (\vec{v}_\theta f)$

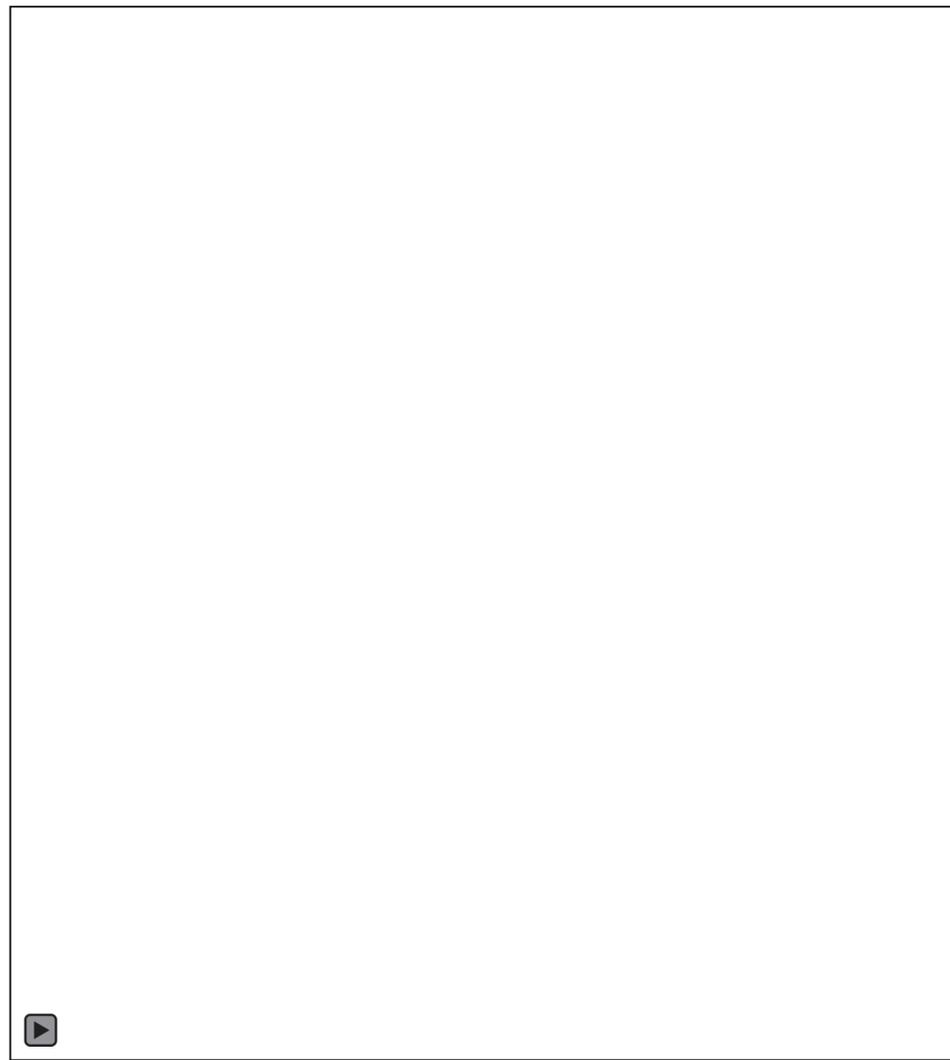
$\int_D \nabla \cdot (\vec{v}_\theta f)$ can be estimated through area-samples ●



A 2D Example Scene



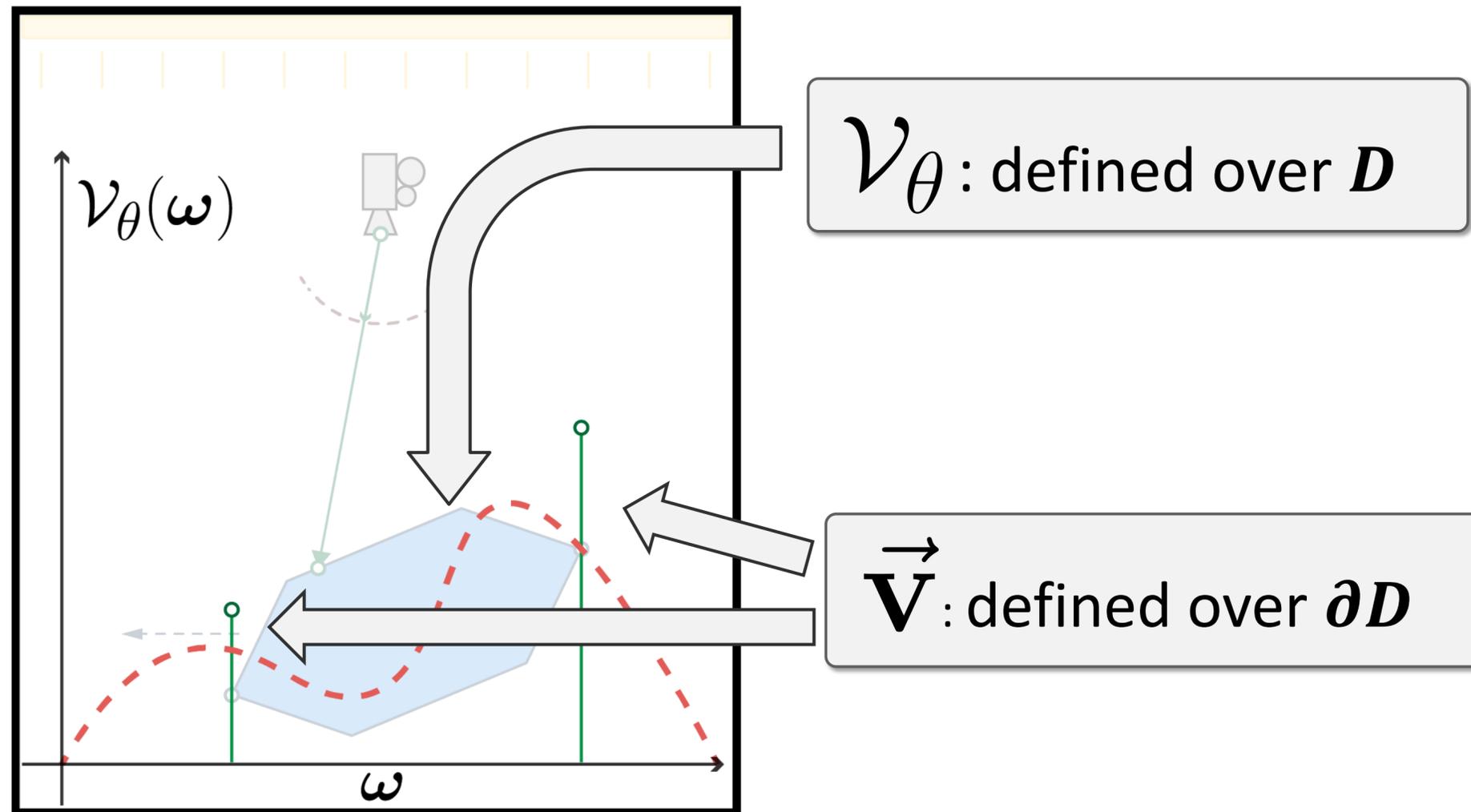
Velocity \vec{v} : the Boundary derivative



$\partial_{\theta} \omega_i^{(b)}$: Derivative of boundary position w.r.t θ

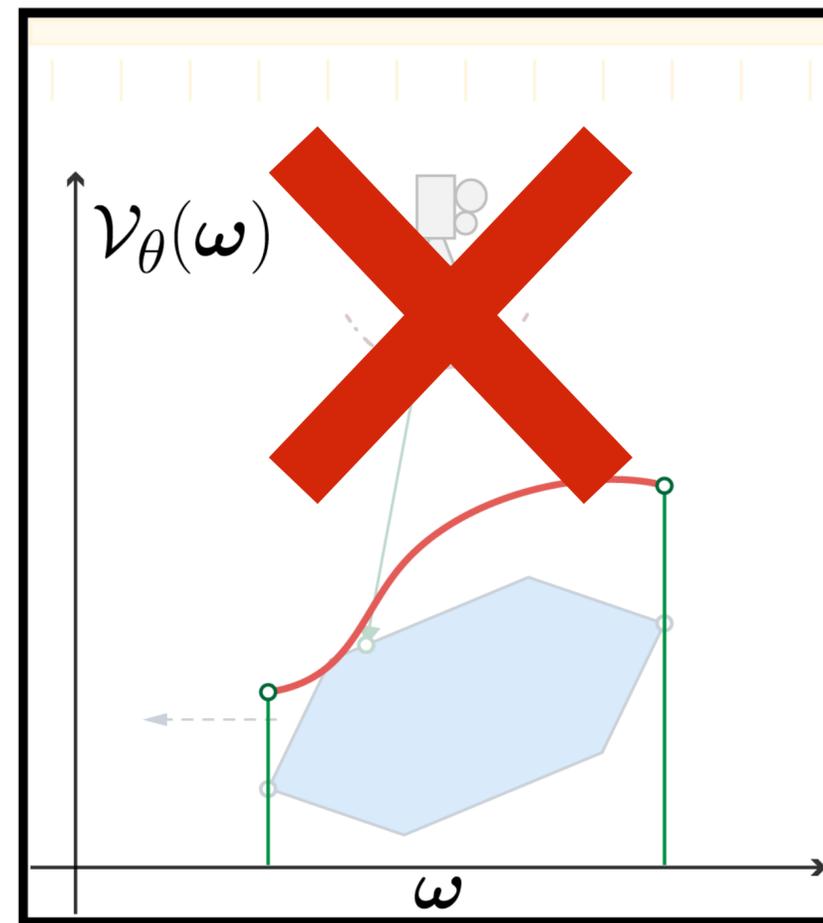
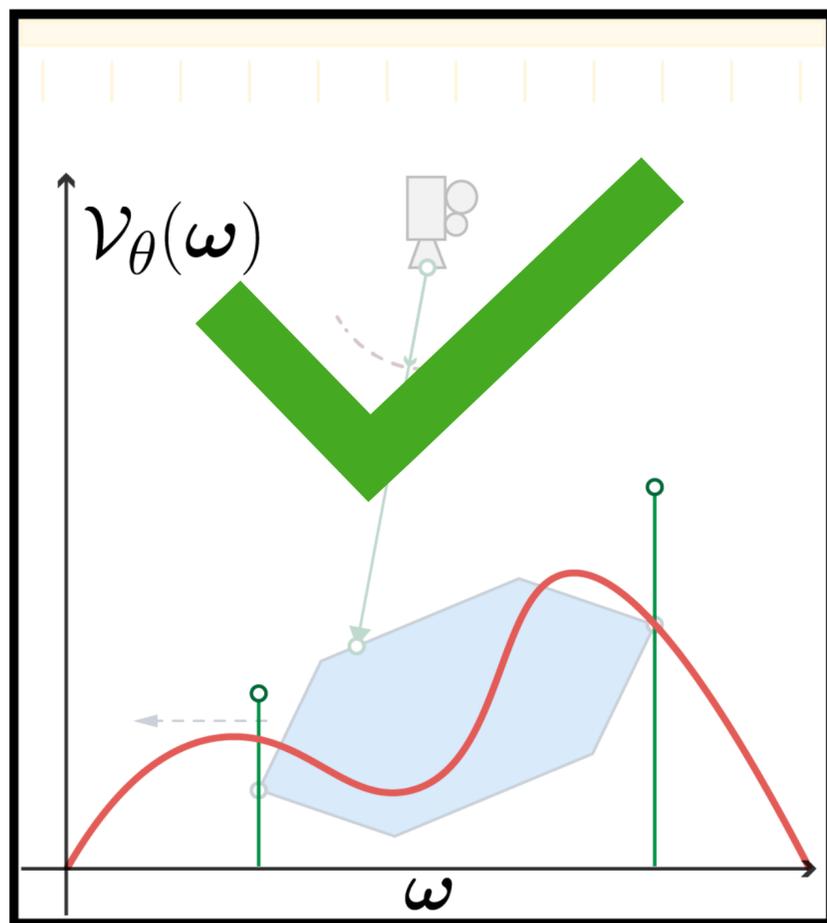
θ

Warp Field $\vec{\mathcal{V}}_\theta$: Extension of \vec{v} to all points



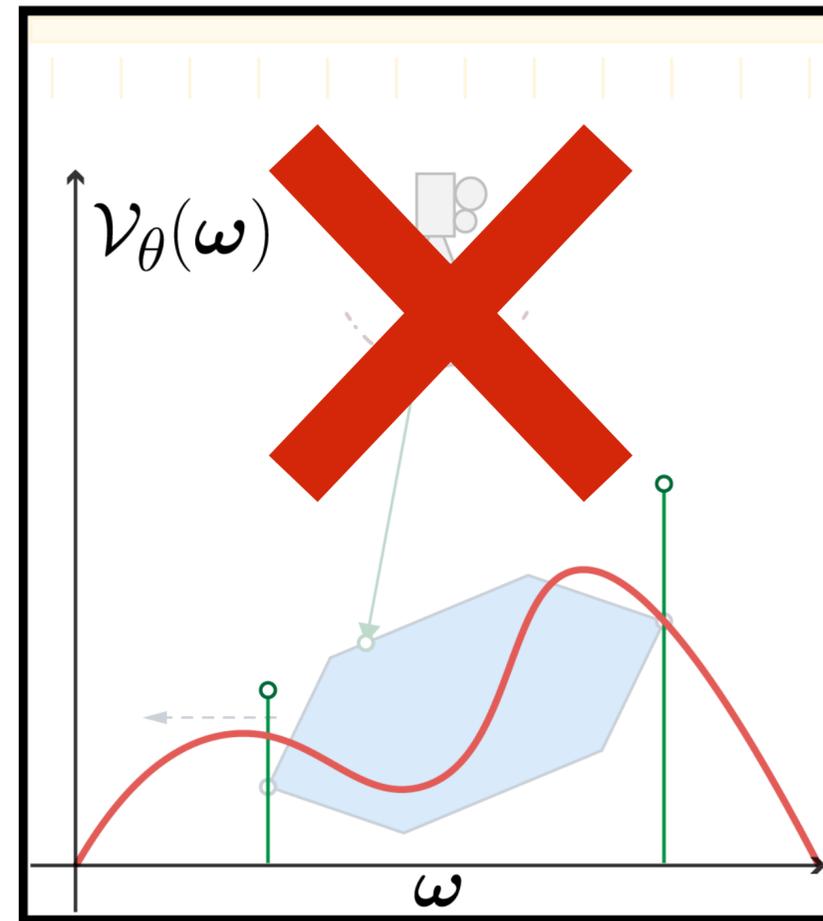
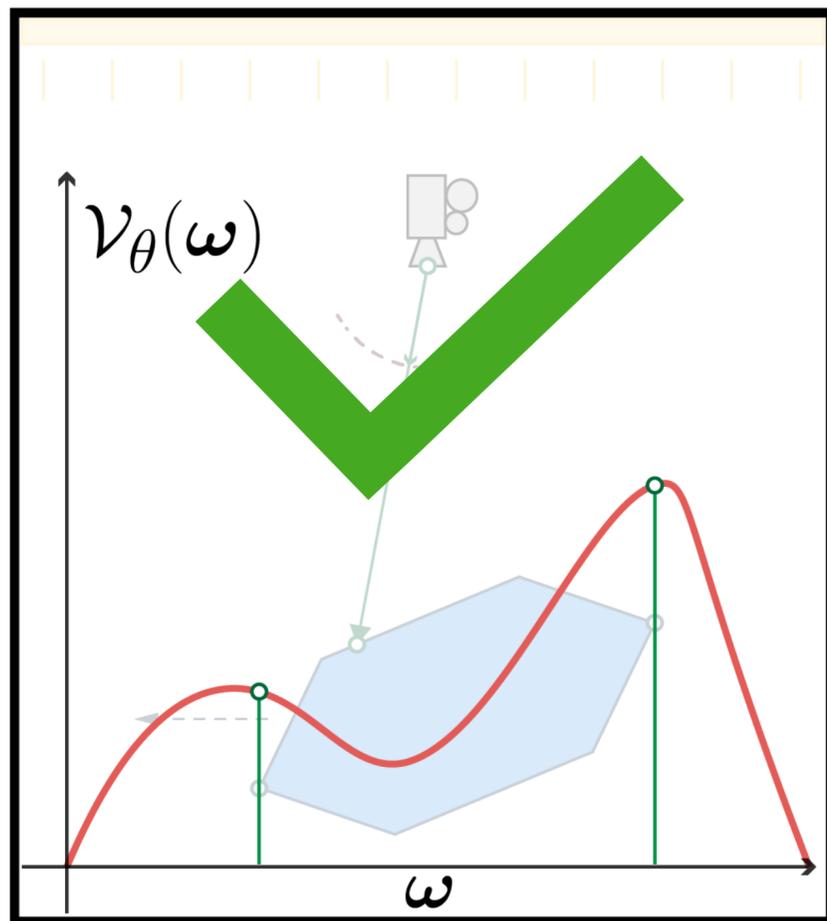
Validity of $\vec{\nu}_\theta$

Rule 1: Continuous

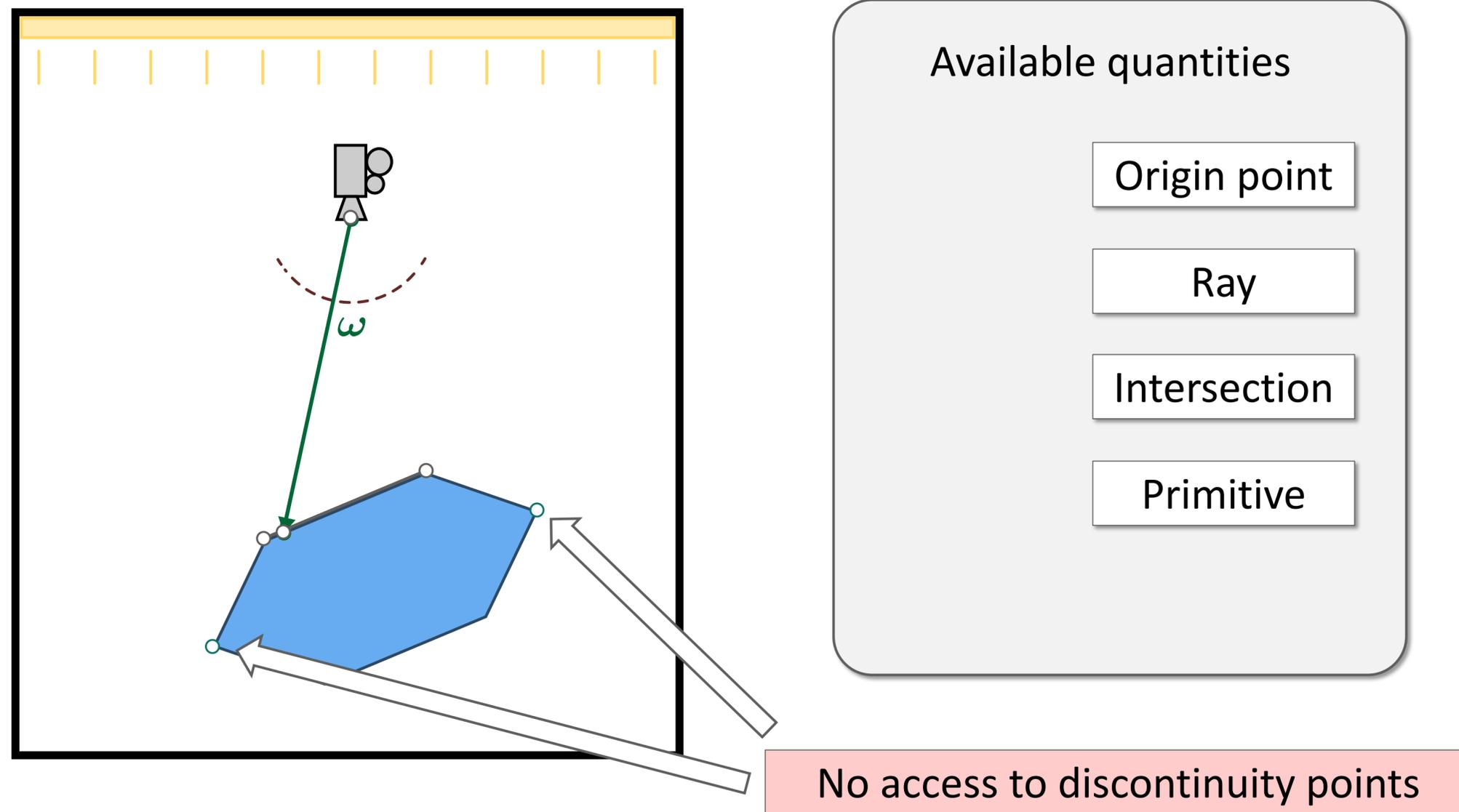


Validity of $\vec{\nu}_\theta$

Rule 2: Boundary Consistent



Interpolation without knowledge of boundaries



Constructing \vec{v}_θ

Attempt 1 \longrightarrow Find $\partial_\theta \omega$ through *implicit derivative*

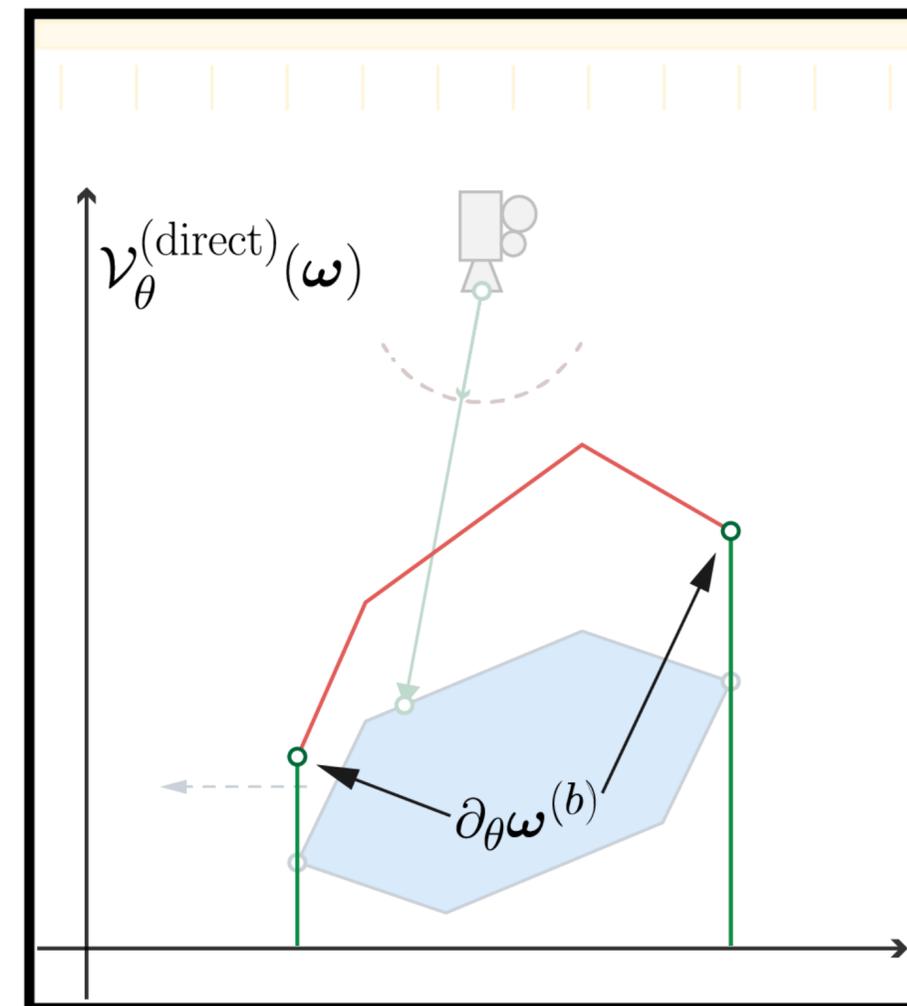
(Incorrect)

$$\mathbf{y} = \text{INTERSECT}(\omega, \theta) \implies \partial_\theta \omega = \frac{\partial_\omega \mathbf{y}}{\partial_\theta \mathbf{y}}$$

At all points (not just boundaries)

+ Boundary consistent

- Not continuous



Constructing $\vec{\nu}_\theta$

Attempt 2 \longrightarrow Filter *Attempt 1* with a Gaussian filter

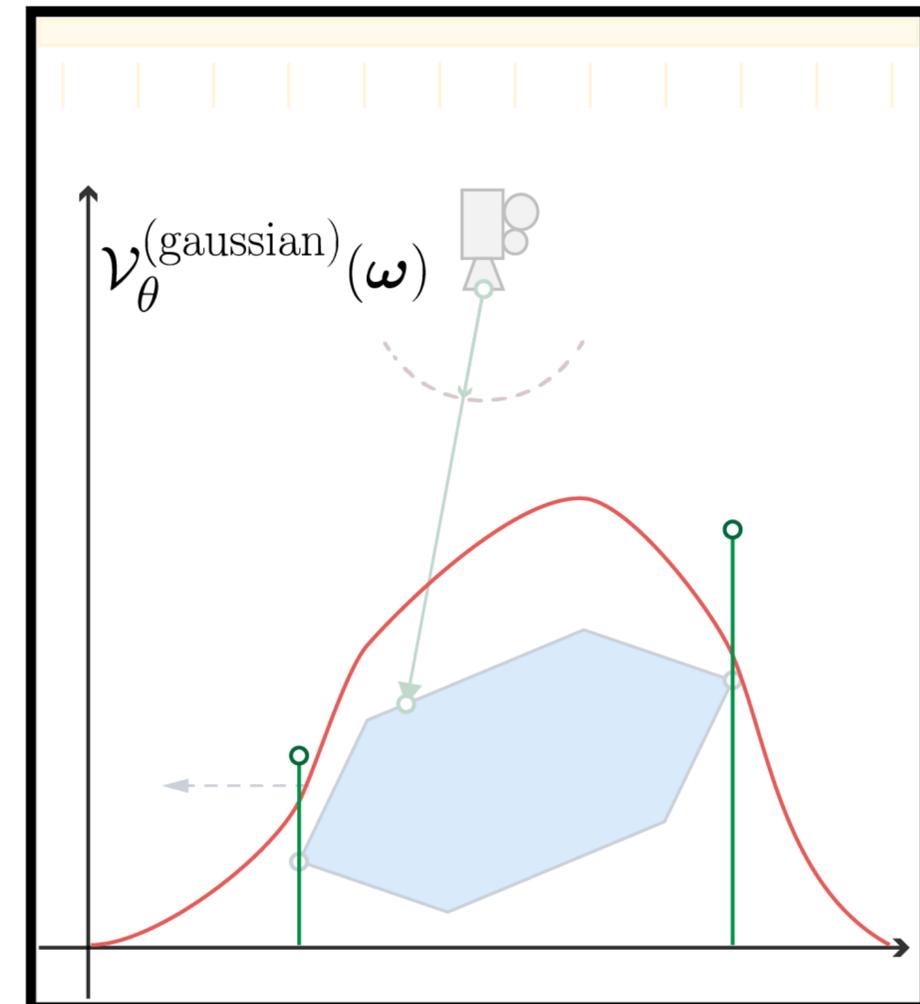
(Incorrect)

$$\int_{\Omega'} k(\omega, \omega') \frac{\partial \omega \mathbf{y}}{\partial \theta \mathbf{y}}$$

$k(.,.) = \text{Gaussian filter}$

+ Continuous

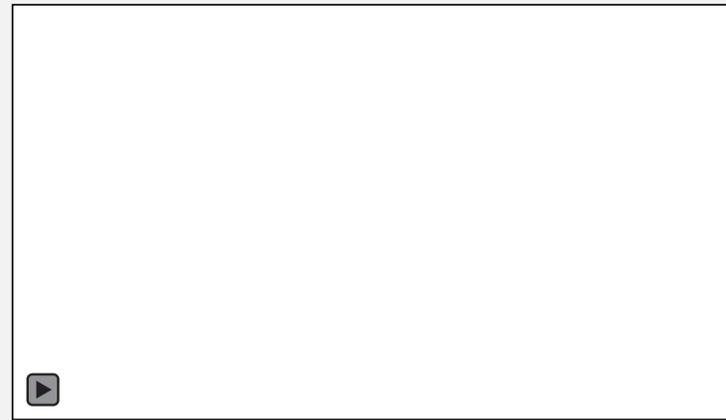
- Not boundary consistent



Boundary-Aware Weighting

Goal: Find weights $k(\omega, \omega')$ s.t. $\vec{V}_\theta = \frac{\partial \omega \mathbf{y}}{\partial \theta \mathbf{y}}$ at boundaries.

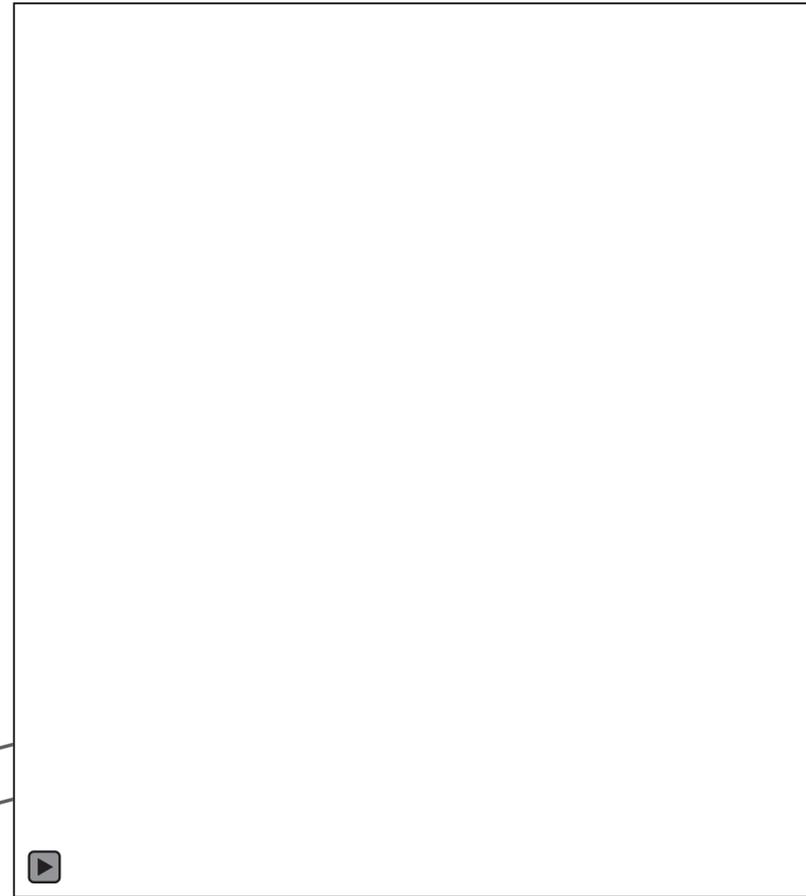
Ideal weighting function



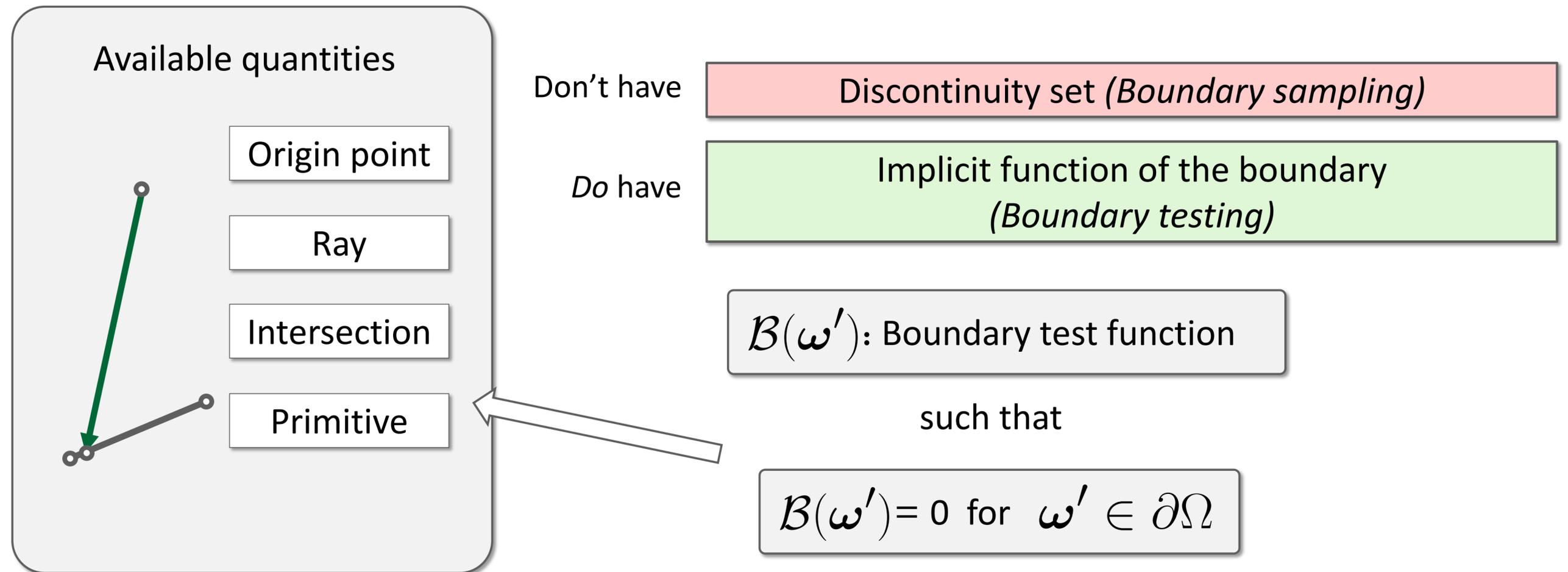
ω



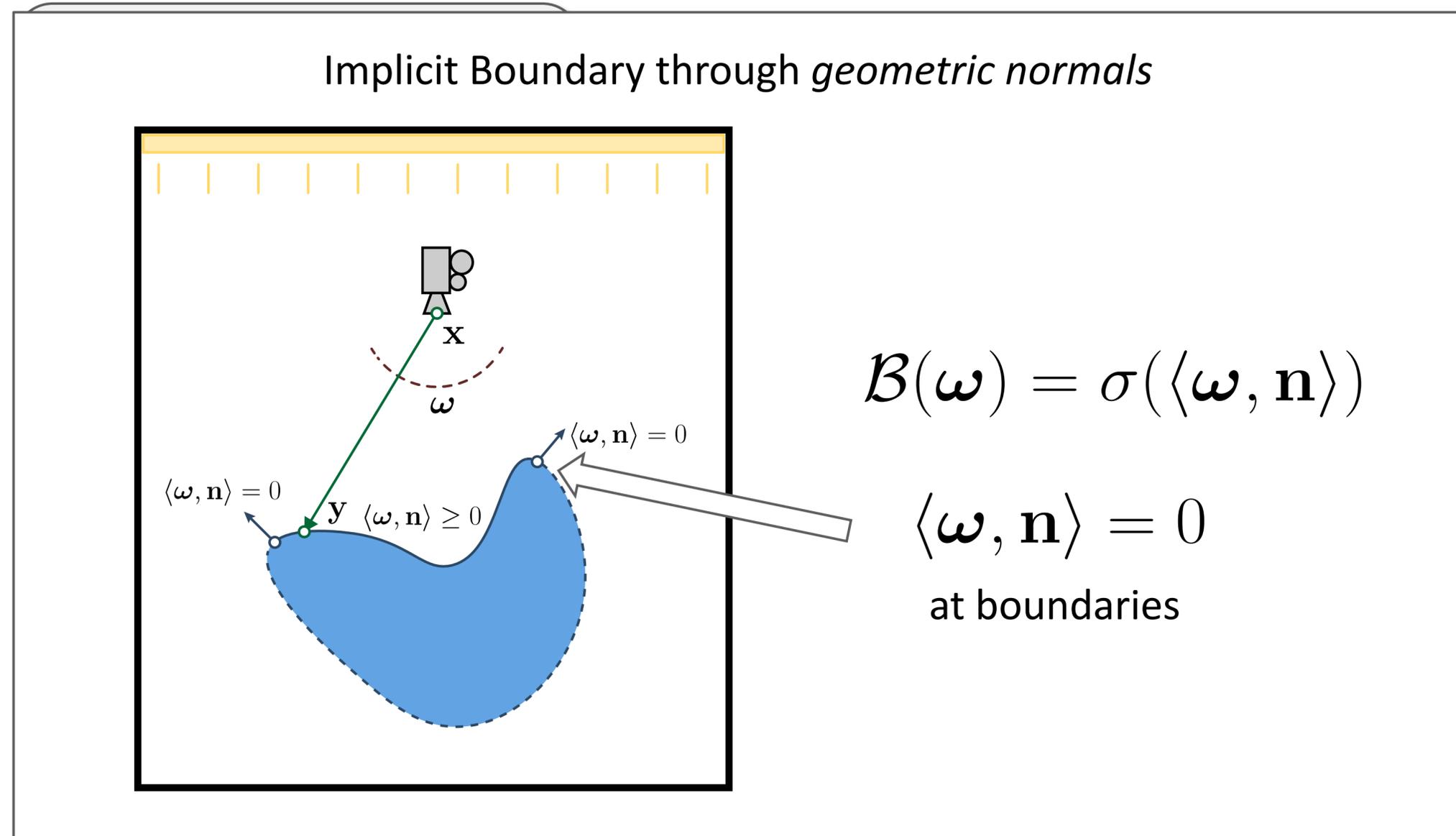
Approach Dirac delta near boundaries



Boundary-Aware Weighting



Boundary-Aware Weighting



ray sampling)

boundary
g)

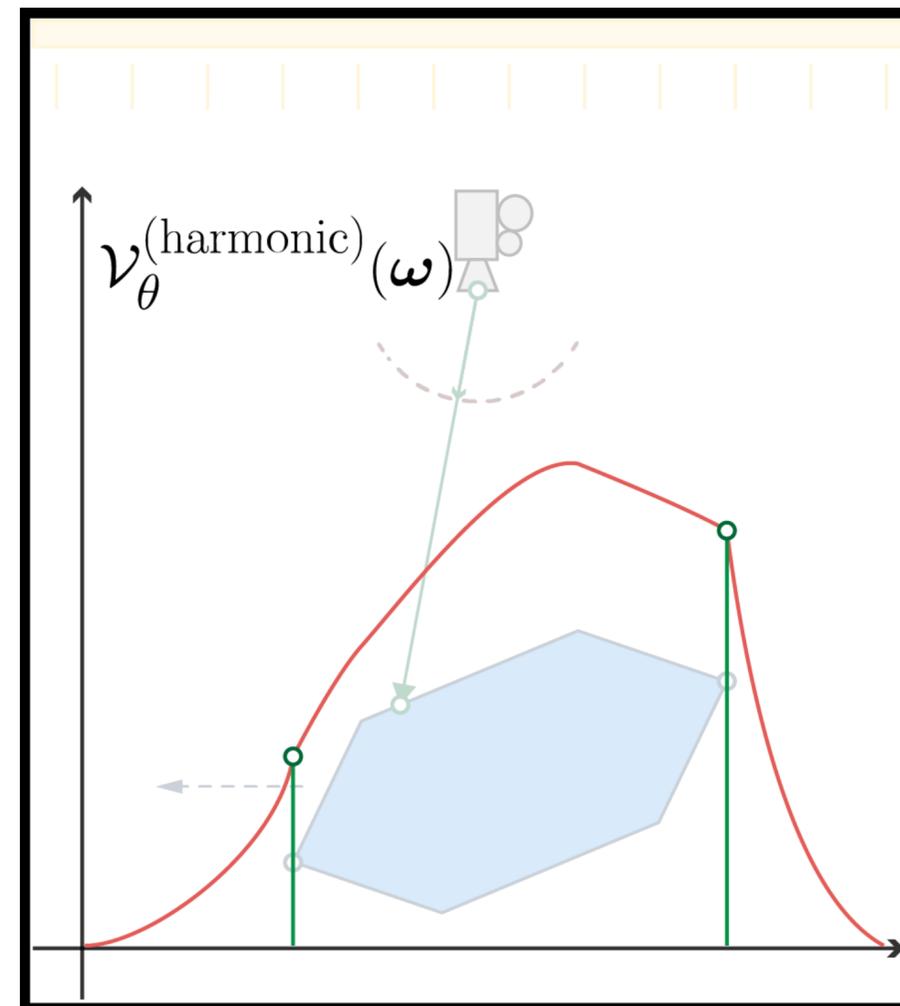
Constructing $\vec{\nu}_\theta$

Our Approach \longrightarrow Filter *Attempt 1* with harmonic weights

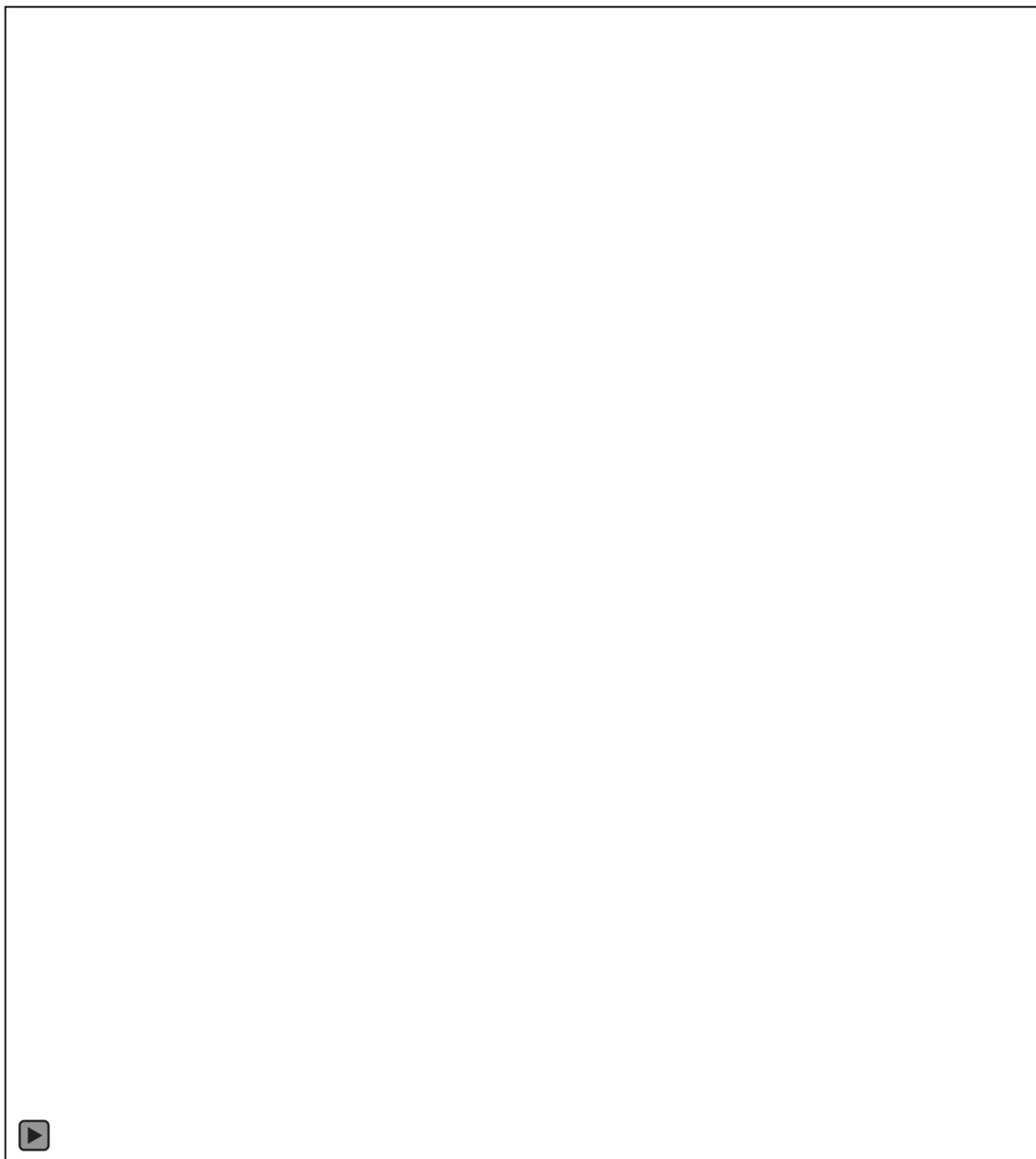
$$k(\omega, \omega') = \frac{1}{\boxed{D(\omega, \omega')} + \boxed{B(\omega')}}}$$

Distance function Boundary test

+ Boundary consistent
+ Continuous



Computing $\vec{\mathcal{V}}_{\theta}$



1. Sample **path** using path tracer *(N paths)*

For each bounce:

2. Sample **auxiliary** rays *(N' rays)*

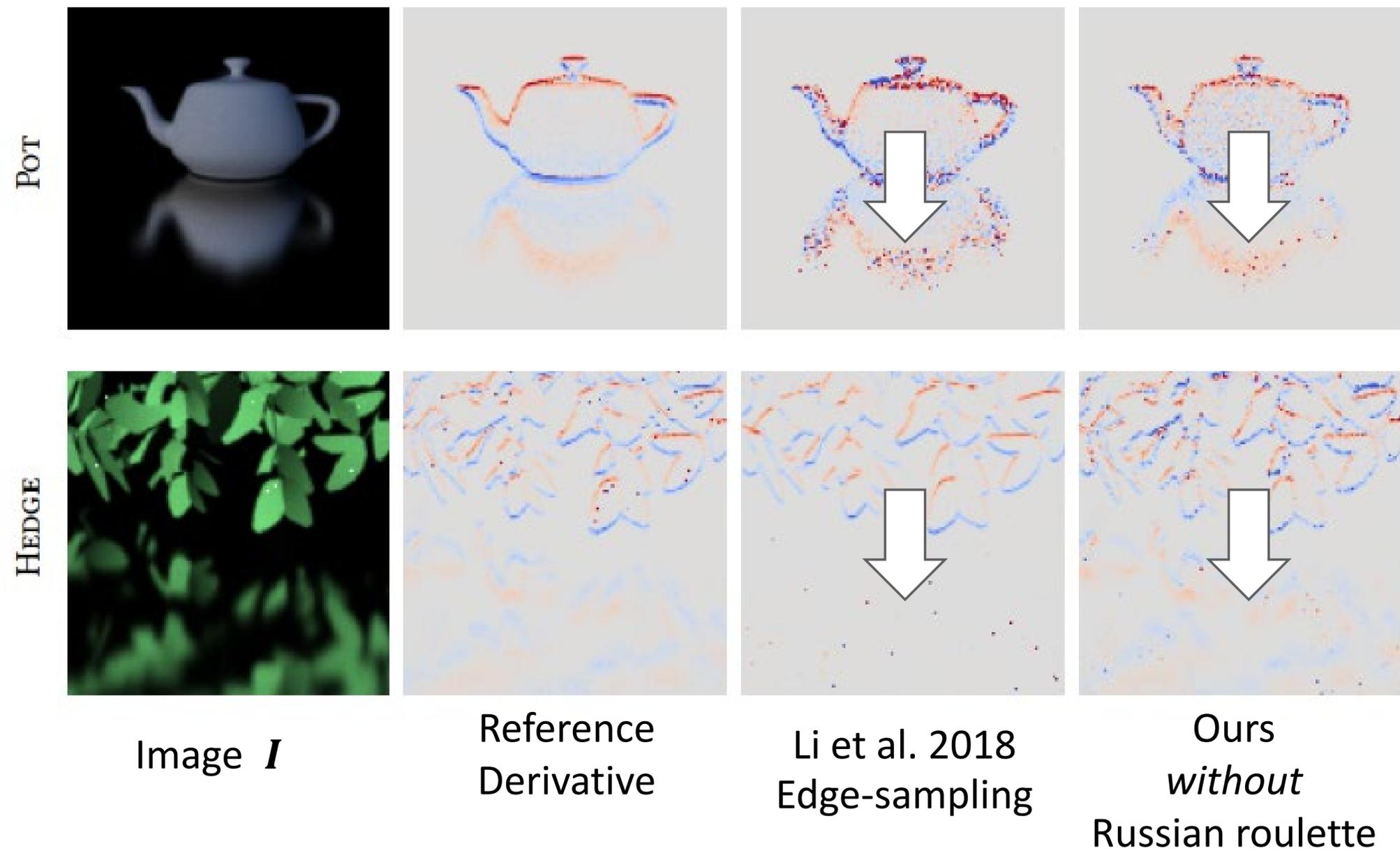
3. Compute boundary term **B()** locally

4. Compute weight **k(...)** and $\partial_{\theta}\omega$

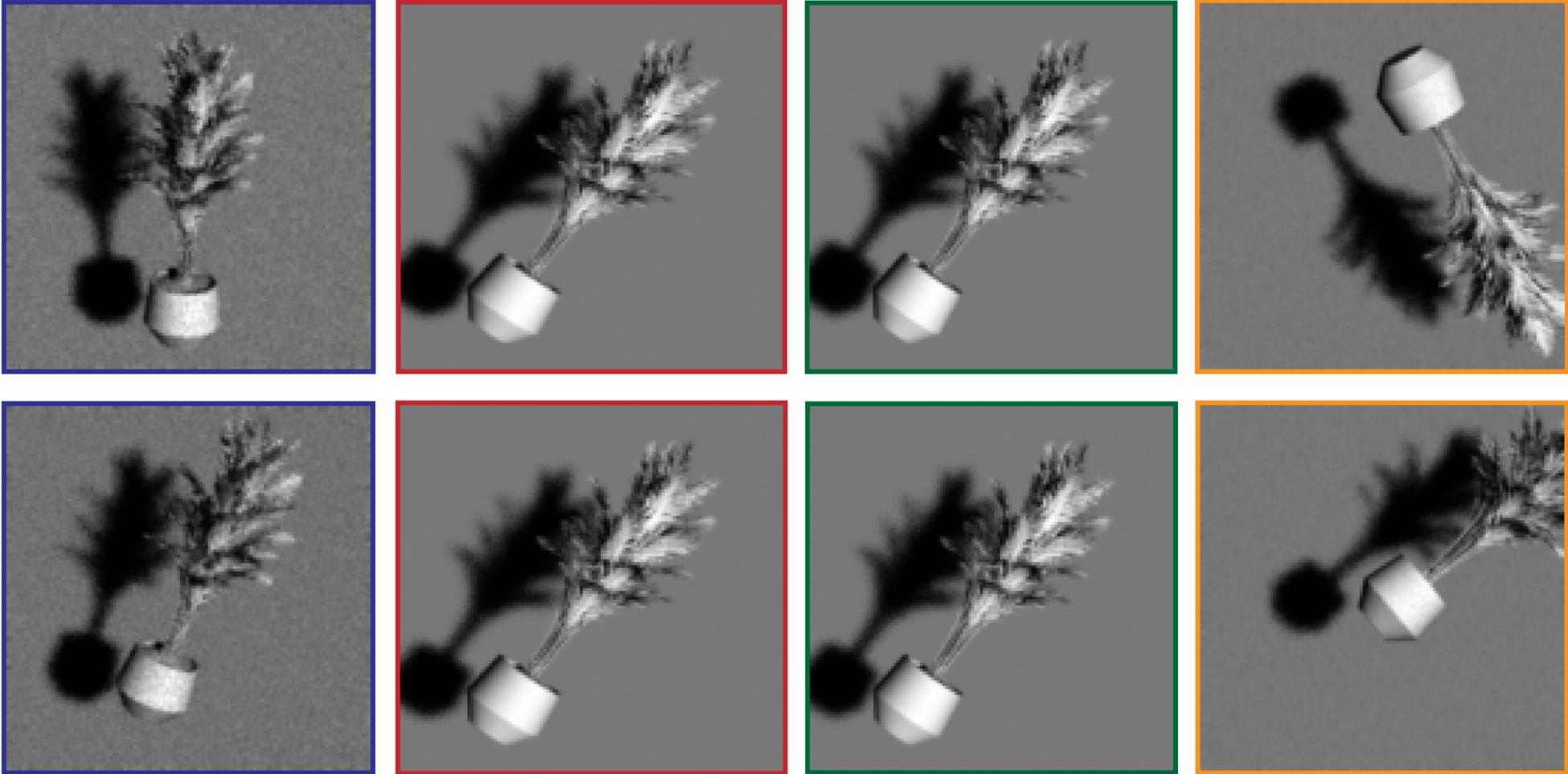
5. Find weighted mean

RESULTS

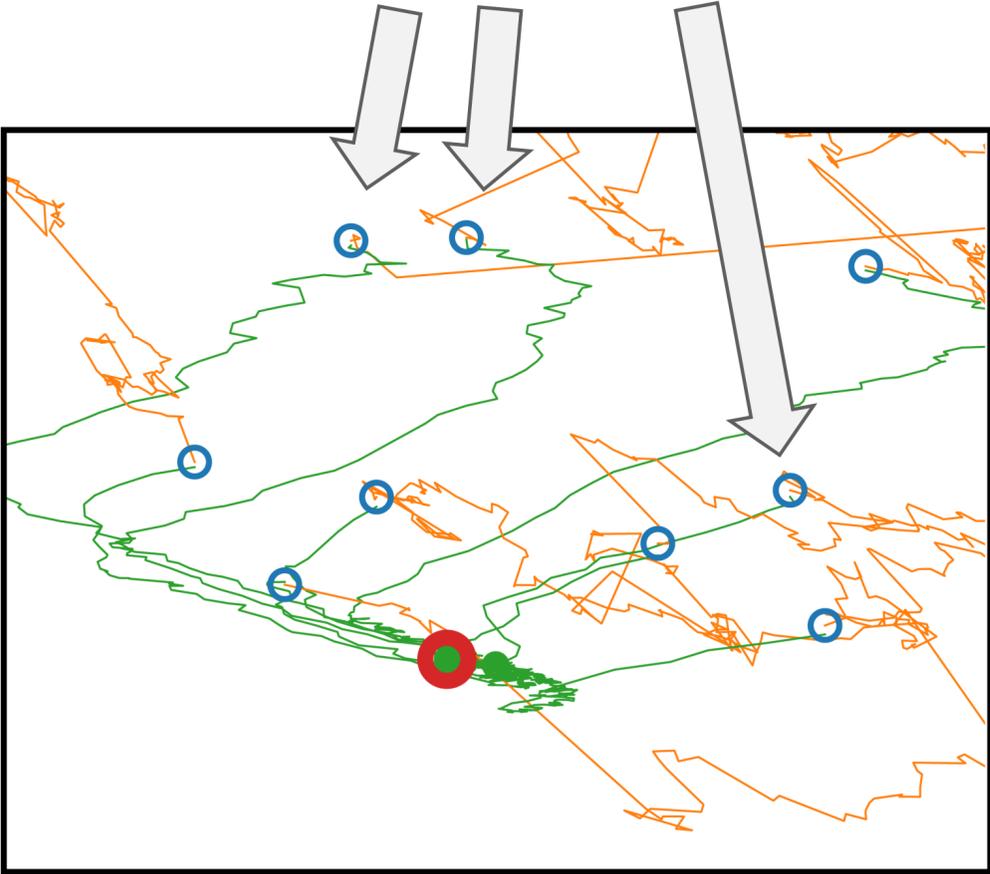
Area-sampling handles higher-order effects better



Pose estimation can fail with biased gradients



Multiple Initializations



Optimization trajectories

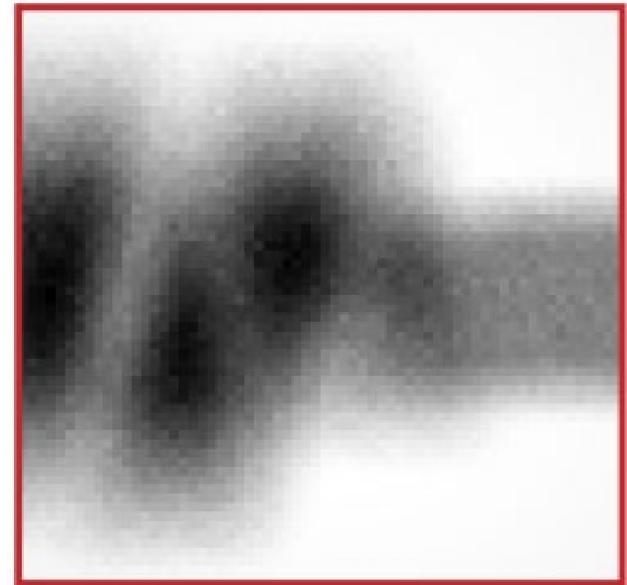
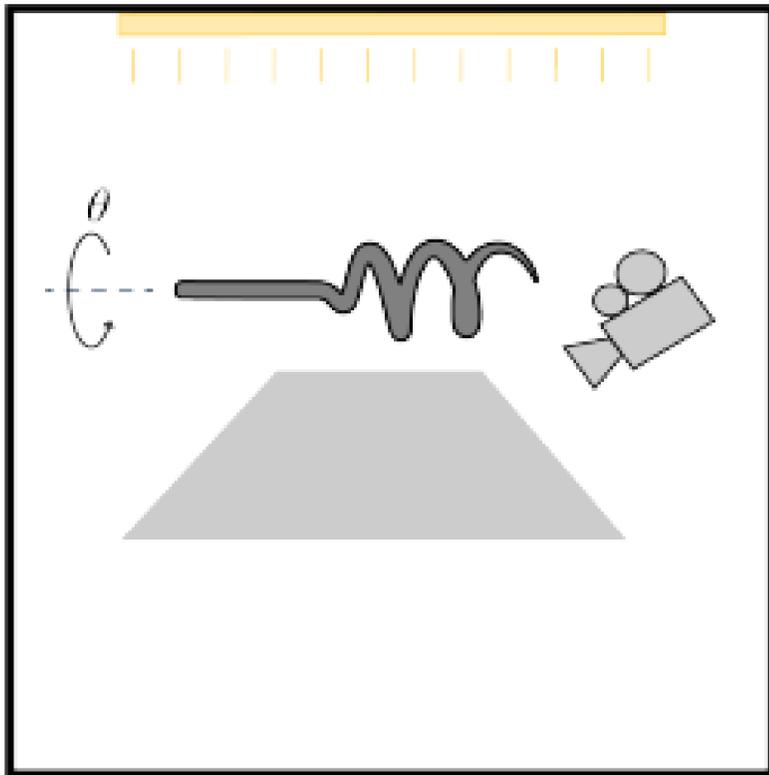
○ Initialization

○ Target

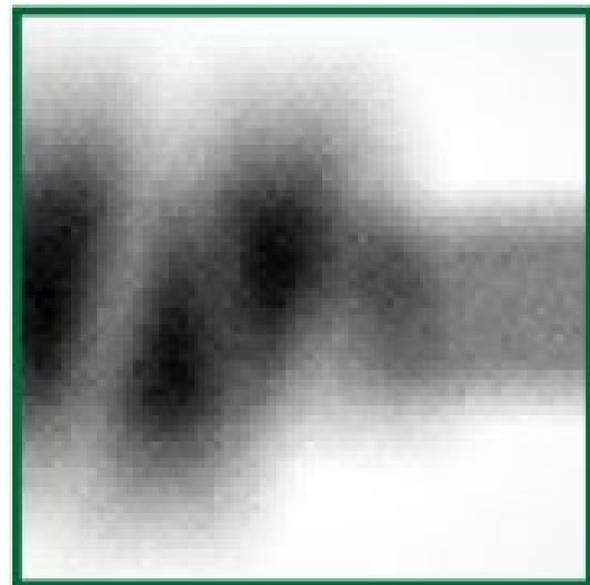
● Ours

● Reparameterization
(Biased gradients)

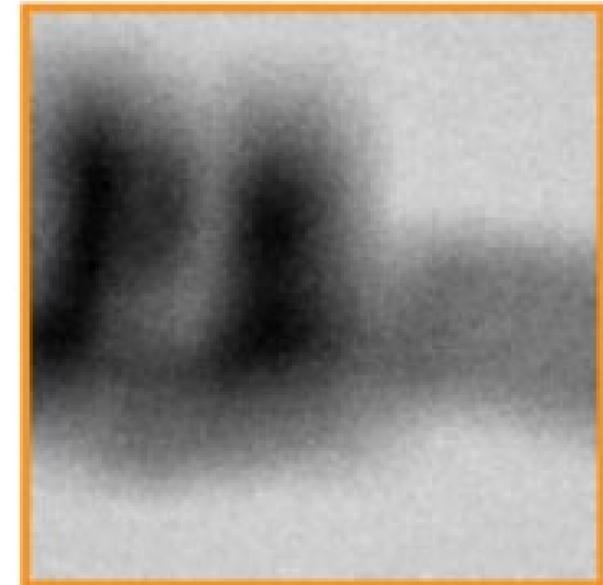
Pose estimation can fail with biased gradients



○ Target



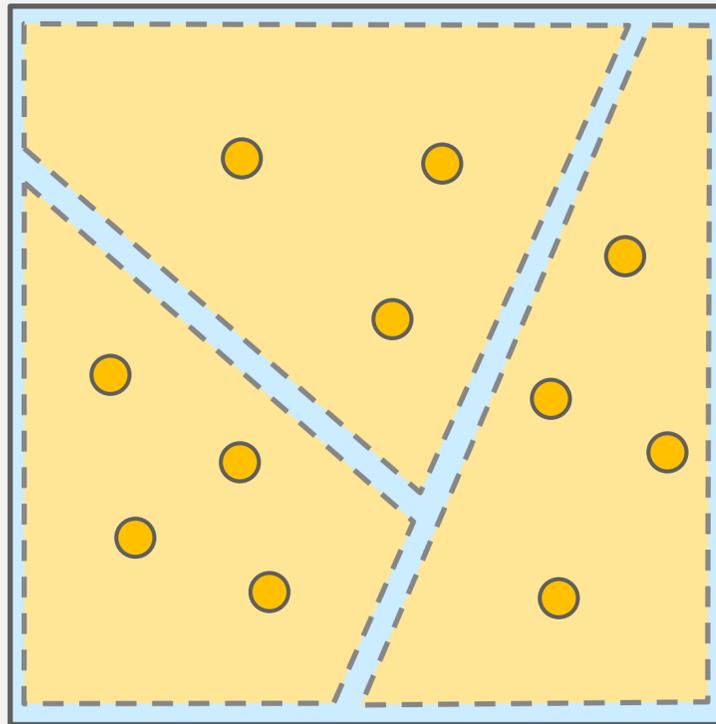
● Ours



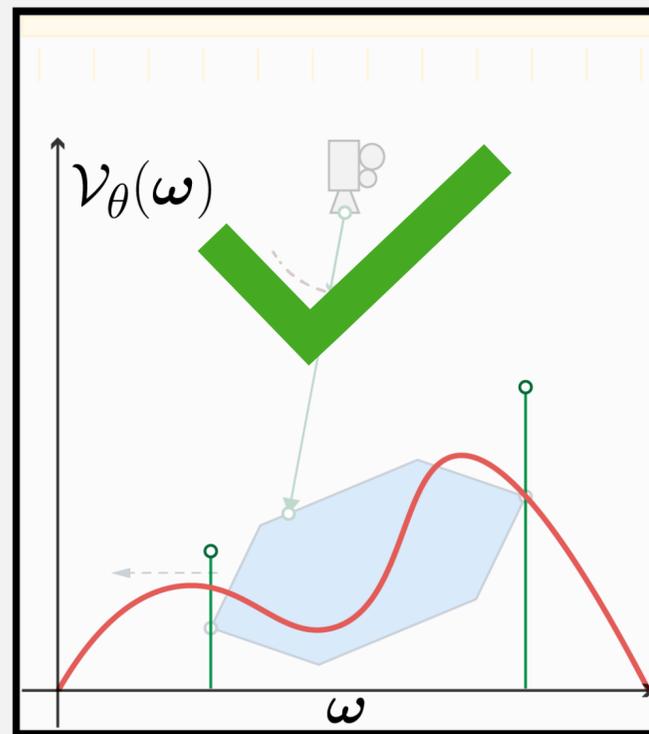
● Reparameterization
(Biased gradients)

Summary of Warped-area sampling

Edge-integral to Area-integral

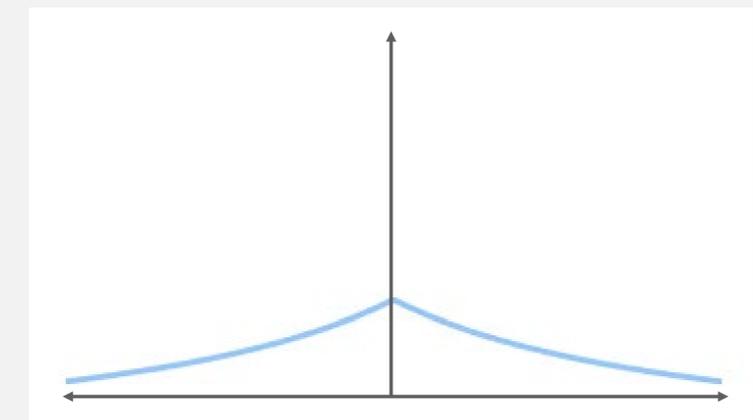


Warp field conditions

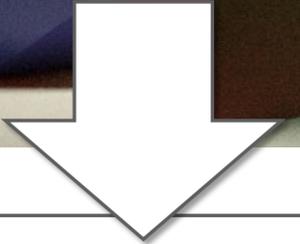
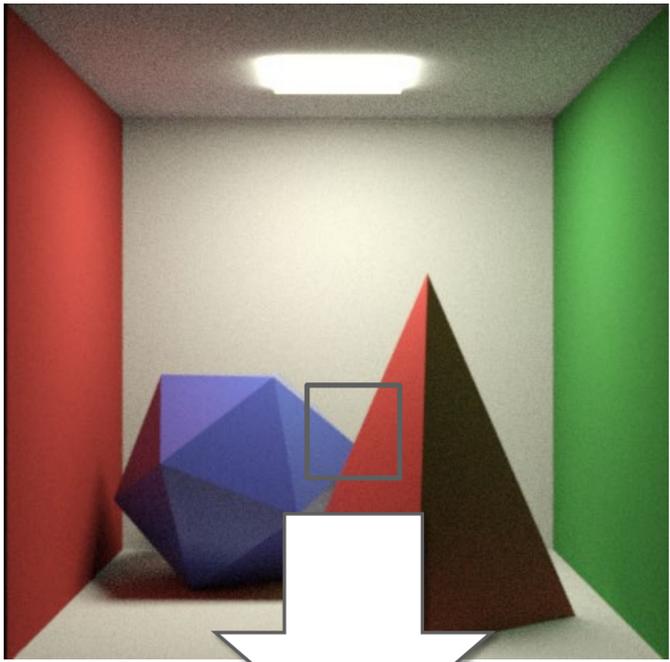


Harmonic interpolation

$$k(\omega, \omega') = \frac{1}{\mathcal{D}(\omega, \omega') + \mathcal{B}(\omega')}$$

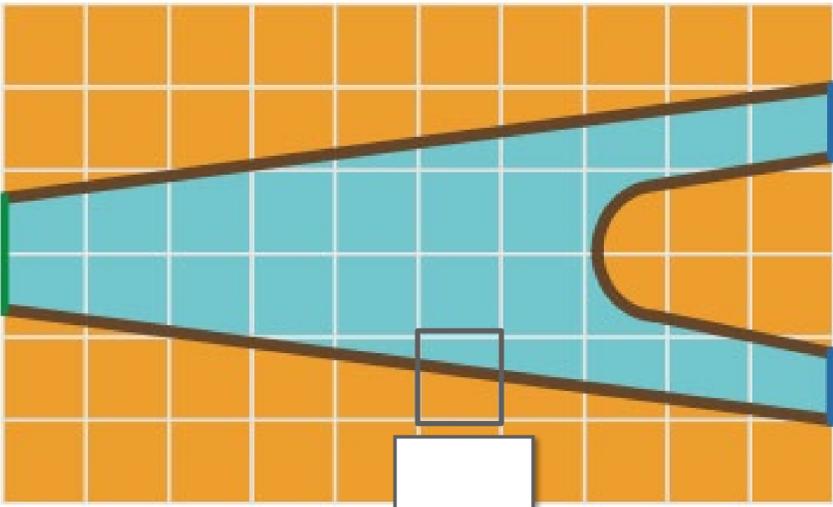


Many programs in graphics have this problem



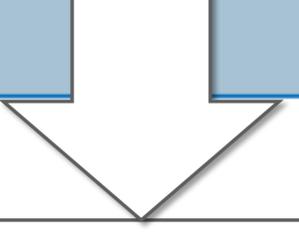
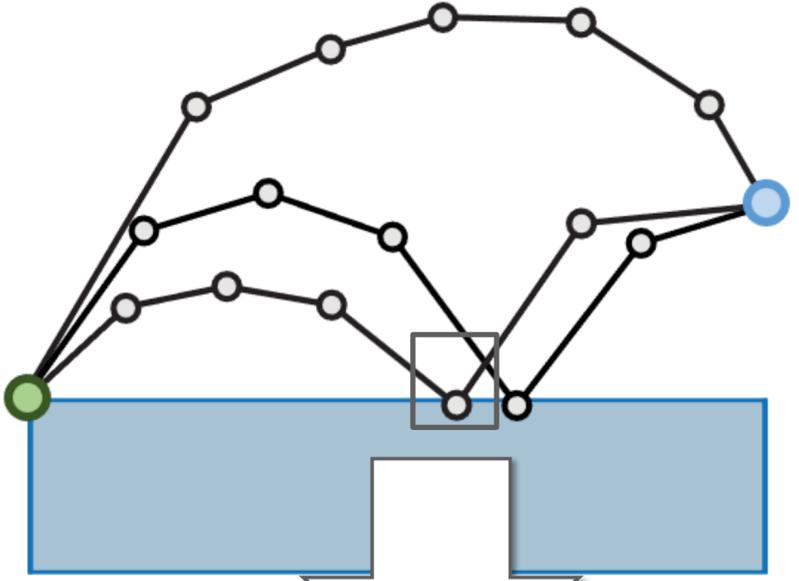
\int

(Bangaru 2020)



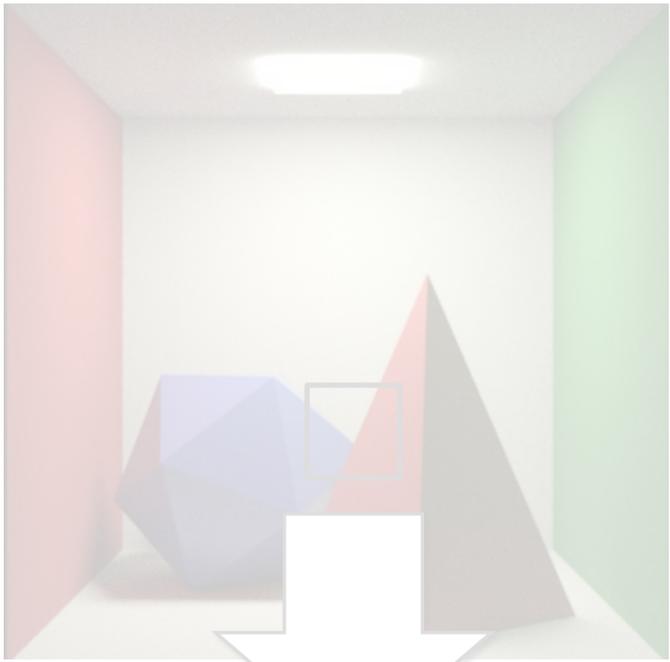
\int

(Du 2020)



\int_t

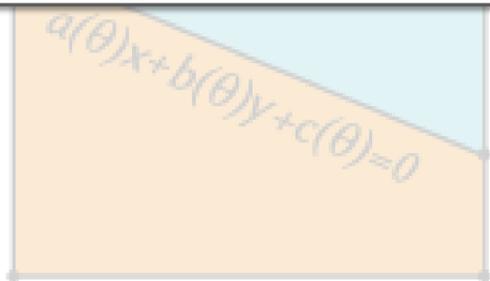
Many programs in graphics have this problem

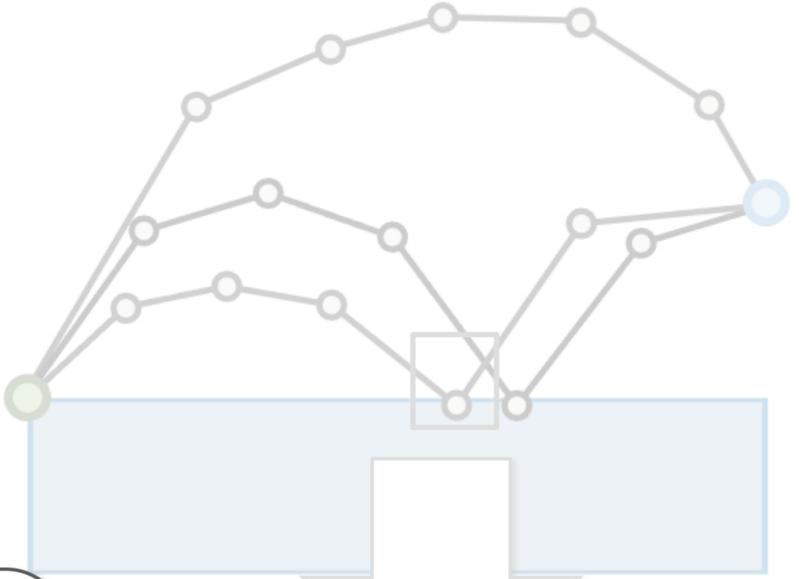


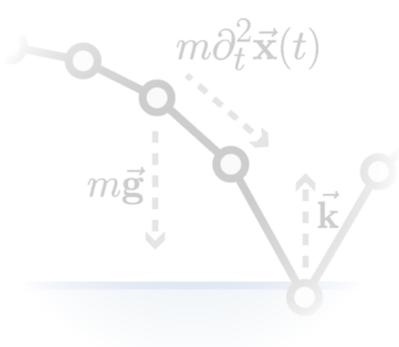
\int 
(Bangaru 2020)



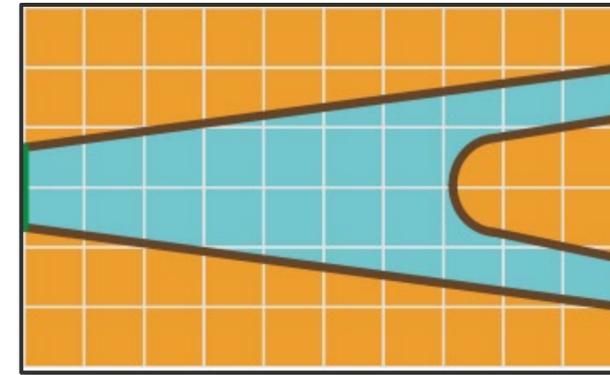
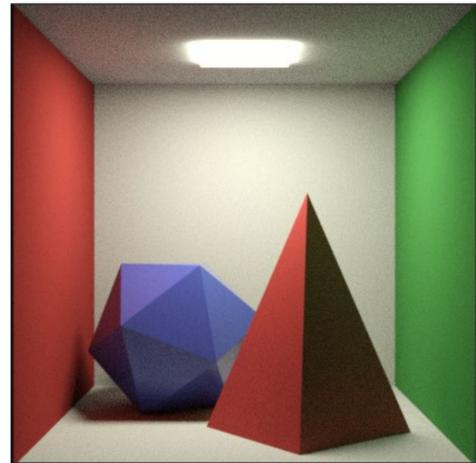
We have seen why it's difficult to differentiate such integrals

\int 
(Du 2020)



\int_t 

Existing solutions to this specific problem



Differentiable Monte Carlo Ray Tracing through Edge Sampling

Path-Space Differentiable Rendering

CHENG ZHANG, University of California, Irvine
 BAILEY MILLER, Carnegie Mellon University
 KAI YAN, University of California, Irvine
 IOANNIS GKIOULEKAS, Carnegie Mellon University
 SHUANG ZHAO, University of California, Irvine

Fig. 1. with compute shows As one (a) will optimize method. Gradient rasterization parameterize different do not they do coordinate. We know the direct the dis method times is desired.

Fig. 1. We introduce **path-space differentiable rendering**, a new theoretical framework to estimate derivatives of radiometric measurements with respect to arbitrary scene parameters (e.g., material properties and object geometries). By directly differentiating full path integrals, we derive the **differential path integral** framework, enabling the design of new unbiased Monte Carlo methods capable of efficiently estimating derivatives in virtual scenes with complex geometry and light transport effects. This example shows a dining room scene lit by the sun from outside the window. On the right, we show the corresponding derivative image with respect to the vertical location of the sun. (Please use Adobe Acrobat to view the teaser images to see them animated.)

Physics-based differentiable rendering, the estimation of derivatives of radiometric measures with respect to arbitrary scene parameters, has a diverse array of applications from solving analysis-by-synthesis problems to training machine learning pipelines incorporating forward rendering processes. Unfortunately, general-purpose differentiable rendering remains challenging due to the lack of efficient estimators as well as the need to identify and handle complex discontinuities such as visibility boundaries.

In this paper, we show how path integrals can be differentiated with respect to arbitrary differentiable changes of a scene. We provide a detailed theoretical analysis of this process and establish new differentiable rendering formulations based on the resulting differential path integrals. Our path-space differentiable rendering formulation allows the design of new Monte Carlo estimators that offer significantly better efficiency than state-of-the-art methods in handling complex geometric discontinuities and light transport phenomena such as caustics.

We validate our method by comparing our derivative estimates to those generated using the finite-difference method. To demonstrate the effectiveness of our technique, we compare inverse-rendering performance with a few state-of-the-art differentiable rendering methods.

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

Additional Key Words and Phrases: Differentiable rendering, path integral, Monte Carlo rendering

ACM Reference Format:
 Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-Space Differentiable Rendering. *ACM Trans. Graph.* 39, 4, Article 143 (July 2020), 19 pages. <https://doi.org/10.1145/3386569.3392383>

1 INTRODUCTION

Physics-based light transport simulation, a core research topic in computer graphics since the field's inception, focus on numerically estimating radiometric sensor responses in fully specified virtual scenes. Previous research efforts have led to mature **forward rendering** algorithms that can efficiently and accurately simulate light transport in virtual environments with high complexities.

Differentiable rendering computes the derivatives of radiometric measurements with respect to differential changes of such environments. These techniques can enable, for example, (i) **gradient-based optimization** when solving *inverse-rendering* problems; and (ii) efficient integration of physics-based light transport simulation in *machine learning* and *probabilistic inference* pipelines.

ACM Trans. Graph., Vol. 39, No. 4, Article 143. Publication date: July 2020.

Differentiable Vector Graphics Rasterization for Editing and Learning

TZU-MAO LI, MIT CSAIL

Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning

Shichen Liu^{1,2}, Tianye Li^{1,2}, Weikai Chen¹, and Hao Li^{1,2,3}

¹USC Institute for Creative Technologies
²University of Southern California
³Pinscreen

{shichen, tli, weichen}@ict.usc.edu hao@hao-li.com

Abstract

Rendering bridges the gap between 2D vision and 3D scenes by simulating the physical process of image formation. By inverting such renderer, one can think of a learning approach to infer 3D information from 2D images. However, standard graphics renderers involve a fundamental discretization step called rasterization, which prevents the rendering process to be differentiable, hence able to be learned. Unlike the state-of-the-art differentiable renderers [29, 19], which only approximate the rendering gradient in the back propagation, we propose a truly differentiable rendering framework that is able to (1) directly render colorized mesh using differentiable functions and (2) back-propagate efficient supervision signals to mesh vertices and their attributes from various forms of image representations, including silhouette, shading and color images. The key to our framework is a novel formulation that views rendering as an aggregation function that fuses the probabilistic contributions of all mesh triangles with respect to the rendered pixels. Such formulation enables our framework to flow gradients to the occluded and far-range vertices, which cannot be achieved by the previous state-of-the-art. We show that by using the proposed renderer, one can achieve significant improvement in 3D unsupervised single-view reconstruction both qualitatively and quantitatively. Experiments also demonstrate that our approach is able to handle the challenging tasks in image-based shape fitting, which remain nontrivial to existing differentiable renderers. Code is available at <https://github.com/ShichenLiu/SoftRas>.

1. Introduction

Understanding and reconstructing 3D scenes and structures from 2D images has been one of the fundamental goals in computer vision. The key to image-based 3D reasoning is to find sufficient supervisions flowing from the pixels to the 3D properties. To obtain image-to-3D correlations, prior approaches mainly rely on the matching losses based on 2D

Fig. 1. We rasterize geometric result, (d) image rasterization as a vector pixel tone as a technique, as conflict high-quality respect to. We design a vector perceptual well-known generative under a V.

Fig. 1. We propose Soft Rasterizer \mathcal{R} (upper), a truly differentiable renderer, which formulates rendering as a differentiable aggregating process $A(\cdot)$ that fuses per-triangle contributions $\{D_i\}$ in a “soft” probabilistic manner. Our approach attacks the core problem of differentiating the standard rasterizer, which cannot flow gradients from pixels to geometry due to the discrete sampling operation (below).

key points/contours [3, 35, 26, 32] or shape/appearance priors [1, 28, 6, 23, 48]. However, the above approaches are either limited to task-specific domains or can only provide weak supervision due to the sparsity of the 2D features. In contrast, as the process of producing 2D images from 3D assets, rendering relates each pixel with the 3D parameters by simulating the physical mechanism of image formation. Hence, by inverting a renderer, one can obtain dense pixel-level supervision for *general-purpose* 3D reasoning tasks, which cannot be achieved by conventional approaches.

However, the rendering process is not differentiable in conventional graphics pipelines. In particular, standard mesh renderers involves a discrete sampling operation, called *rasterization*, which prevents the gradient to be flowed into the mesh vertices. Since the forward rendering

Functional Optimization of Fluidic Devices with Differentiable Stokes Flow

TAO DU, MIT CSAIL
 KUI WU, MIT CSAIL
 ANDREW SPIELBERG, MIT CSAIL
 WOJCIECH MATUSIK, MIT CSAIL
 BO ZHU, Dartmouth College
 EFTYCHIOS SIFAKIS, University of Wisconsin-Madison

Design space:
 - NURBS control points
 - Rotation angle ϕ about z-axis

Differentiable Stokes flow simulation

Gradient-based optimization

Optimized shape

Task 1 rotate ϕ

Task 2 rotate ϕ

Fig. 1. Our system automates the design of fluidic devices with differentiable Stokes flow. Given a parameterized design in the form of NURBS surfaces or curves (leftmost) that separate rigid boundaries from fluid flow, we employ a Stokes flow (second from left) that evaluates the performance of this design. The flow is differentiable and gradients can be quickly evaluated, enabling gradient-based optimization (center) of the control points, and thus, the boundary. The optimized design (rightmost) can be specified to operate in one configuration or several. This example features an optimized fluidic rotational switch that shifts flow from the top outlet path to the bottom outlet path when turned.

We present a method for performance-driven optimization of fluidic devices. In our approach, engineers provide a high-level specification of a device using parametric surfaces for the fluid-solid boundaries. They also specify desired flow properties for inlets and outlets of the device. Our computational approach optimizes the boundary of the fluidic device such that its steady-state flow matches desired flow at outlets. In order to deal with computational challenges of this task, we propose an efficient, differentiable Stokes flow solver. Our solver provides explicit access to gradients of performance metrics with respect to the parametric boundary representation. This key feature allows us to couple the solver with efficient gradient-based optimization methods. We demonstrate the efficacy of this approach on designs of five complex 3D fluidic systems. Our approach makes an important step towards practical computational design tools for high-performance fluidic devices.

CCS Concepts • **Computing methodologies** → **Physical simulation**.

Additional Key Words and Phrases: Physically-based simulation, fluid simulation, computational design optimization

Authors' addresses: Tao Du, MIT CSAIL, taodu@csail.mit.edu; Kui Wu, MIT CSAIL, kuwu@csail.mit.edu; Andrew Spielberg, MIT CSAIL, wojspe@csail.mit.edu; Wojciech Matusik, MIT CSAIL, wojciech@csail.mit.edu; Bo Zhu, Dartmouth College, bozhu@dartmouth.edu; Eftychios Sifakis, University of Wisconsin-Madison, sifakis@cs.wisc.edu.

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s).

© 2020 Copyright held by the owner/author(s).
 0730-0301/2020/12-ART197
<https://doi.org/10.1145/3414685.3417795>

ACM Reference Format:
 Tao Du, Kui Wu, Andrew Spielberg, Wojciech Matusik, Bo Zhu, and Eftychios Sifakis. 2020. Functional Optimization of Fluidic Devices with Differentiable Stokes Flow. *ACM Trans. Graph.* 39, 6, Article 197 (December 2020), 15 pages. <https://doi.org/10.1145/3414685.3417795>

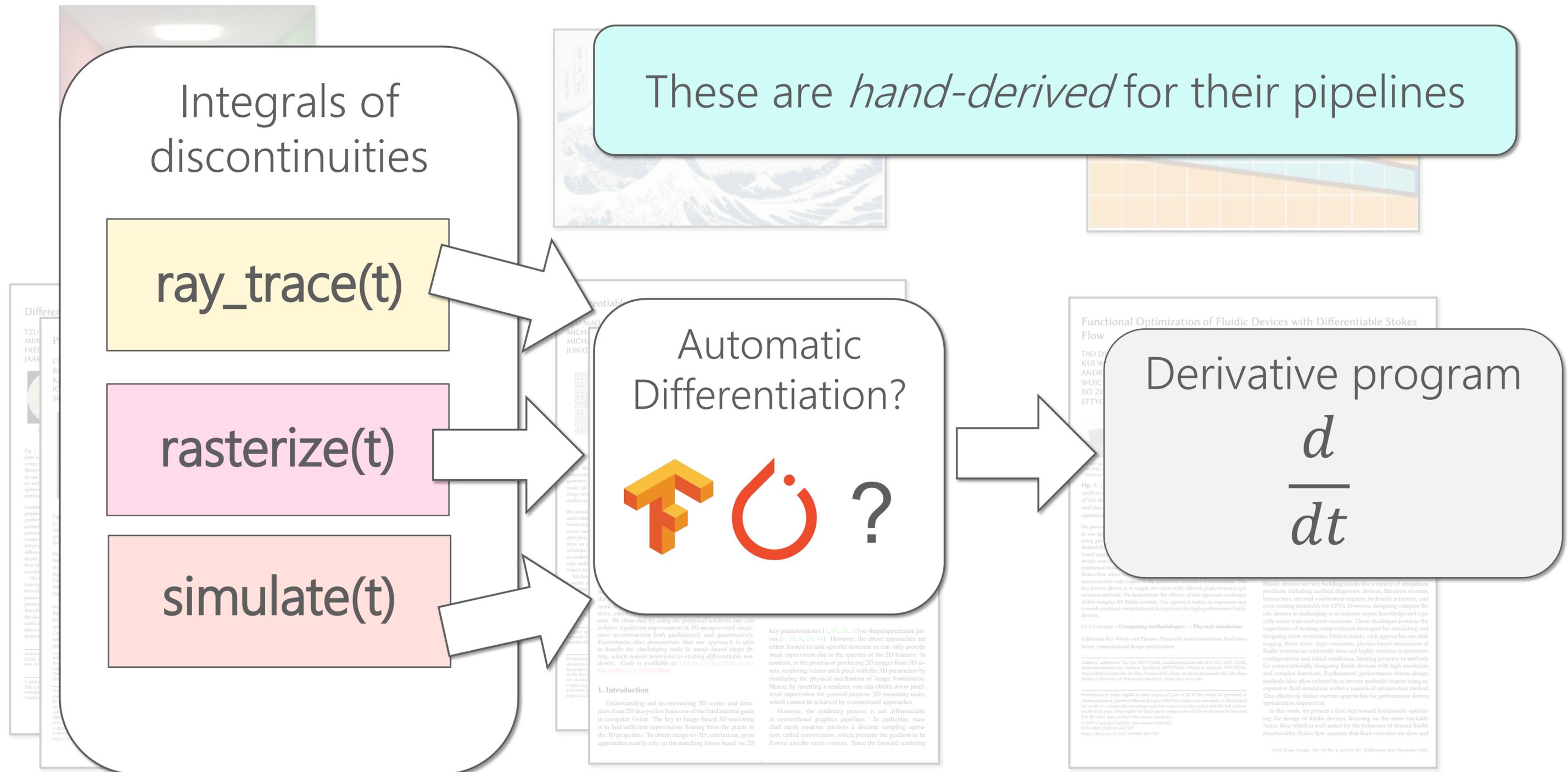
1 INTRODUCTION

Fluidic devices are key building blocks for a variety of ubiquitous products, including medical diagnostic devices, filtration systems, bioreactors, internal combustion engines, hydraulic actuators, and even cooling manifolds for CPUs. However, designing complex fluidic devices is challenging as it requires expert knowledge and typically many trial-and-error iterations. These challenges promote the importance of finding computational strategies for simulating and designing these structures. Unfortunately, such approaches are challenging. Brute-force, high-resolution, physics-based simulations of fluidic systems are inherently slow and highly sensitive to geometric configurations and initial conditions, limiting progress in methods for computationally designing fluidic devices with high resolution and complex functions. Furthermore, performance-driven design methods (also often referred to as inverse methods) require using an expensive fluid simulation within a numerical optimization method. This effectively makes current approaches for performance-driven optimization impractical.

In this work, we present a first step toward functionally optimizing the design of fluidic devices, focusing on the more tractable Stokes flow, which is well-suited for the behaviors of desired fluid functionality. Stokes flow assumes that fluid velocities are slow and

ACM Trans. Graph., Vol. 39, No. 6, Article 197. Publication date: December 2020.

Existing solutions to this specific problem



Systematically Differentiating Parametric Discontinuities

SAI PRAVEEN BANGARU*, MIT CSAIL
 JESSE MICHEL*, MIT CSAIL
 KEVIN MU, MIT CSAIL
 GILBERT BERNSTEIN, UC Berkeley and MIT CSAIL
 TZU-MAO LI, MIT CSAIL
 JONATHAN RAGAN-KELLEY, MIT CSAIL

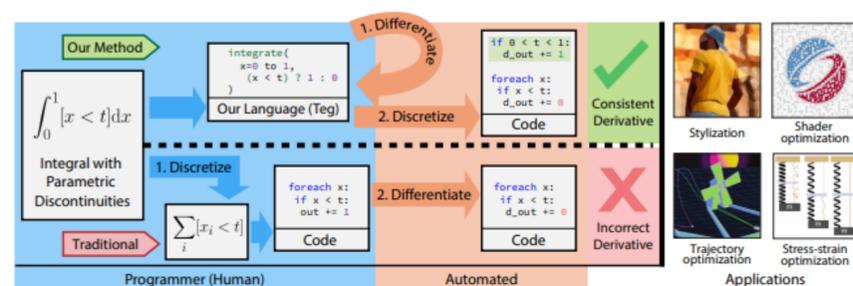


Fig. 1. We propose a language for the automatic differentiation of integrals with discontinuities. Existing auto-diff frameworks require integrals to be discretized into summations prior to differentiation, and therefore lose the derivative contribution from discontinuities. Our method produces a statistically consistent derivative program by introducing integration as a language primitive, which allows us to differentiate discontinuities in continuous space, before discretizing them into summations over discrete samples.

Emerging research in computer graphics, inverse problems, and machine learning requires us to differentiate and optimize parametric discontinuities. These discontinuities appear in object boundaries, occlusion, contact, and sudden change over time. In many domains, such as rendering and physics simulation, we differentiate the parameters of models that are expressed as integrals over discontinuous functions. Ignoring the discontinuities during differentiation often has a significant impact on the optimization process. Previous approaches either apply specialized hand-derived solutions, smooth out the discontinuities, or rely on incorrect automatic differentiation.

We propose a systematic approach to differentiating integrals with discontinuous integrands, by developing a new differentiable programming

*Both authors contributed equally to this research.

Authors' addresses: Sai Praveen Bangaru, MIT CSAIL, Cambridge, MA, sbangaru@mit.edu; Jesse Michel, MIT CSAIL, Cambridge, MA, jmmichel@mit.edu; Kevin Mu, MIT CSAIL, Cambridge, MA, kmu@csail.mit.edu; Gilbert Bernstein, UC Berkeley, Berkeley, CA, MIT CSAIL, Cambridge, MA, gilbo@berkeley.edu; Tzu-Mao Li, MIT CSAIL, Cambridge, MA, tzumao@mit.edu; Jonathan Ragan-Kelley, MIT CSAIL, Cambridge, MA, jrk@csail.mit.edu.

Permission to make digital or hard copies of part or all 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 third-party components of this work must be honored. For all other uses, contact the owner/author(s).
 © 2021 Copyright held by the owner/author(s).
 0730-0301/2021/8-ART107
<https://doi.org/10.1145/3450626.3459775>

language. We introduce integration as a language primitive and account for the Dirac delta contribution from differentiating parametric discontinuities in the integrand. We formally define the language semantics and prove the correctness and closure under the differentiation, allowing the generation of gradients and higher-order derivatives. We also build a system, Teg, implementing these semantics. Our approach is widely applicable to a variety of tasks, including image stylization, fitting shader parameters, trajectory optimization, and optimizing physical designs.

CCS Concepts: • **Theory of computation** → Denotational semantics; • **Mathematics of computing** → **Differential calculus**; Stochastic control and optimization; *Probabilistic inference problems*; • **Computing methodologies** → **Computer graphics**; **Visibility**; **Animation**; Computer vision; **Modeling and simulation**.

Additional Key Words and Phrases: Automatic differentiation, differentiable programming, differentiable graphics, differentiable rendering, differentiable physics, domain-specific language.

ACM Reference Format:

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically Differentiating Parametric Discontinuities. *ACM Trans. Graph.* 40, 4, Article 107 (August 2021), 17 pages. <https://doi.org/10.1145/3450626.3459775>

ACM Trans. Graph., Vol. 40, No. 4, Article 107. Publication date: August 2021.

Systematically Differentiating Parametric Discontinuities

Sai Bangaru*, Jesse Michel*, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, Jonathan Ragan-Kelley
 (MIT CSAIL & UC Berkeley)

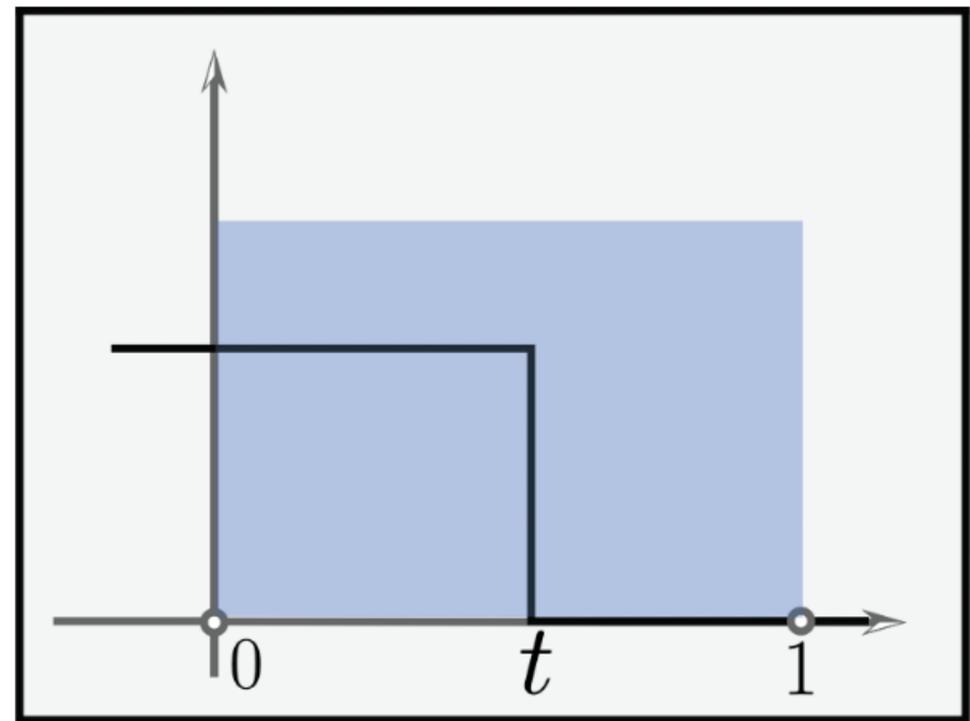
SIGGRAPH 2021 (to appear)

A simple demonstration

$$\frac{d}{dt} \int_0^1 [x < t] dx$$

Derivative of the analytical integral

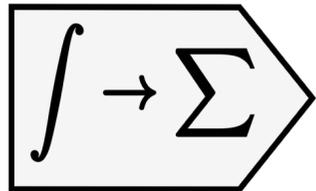
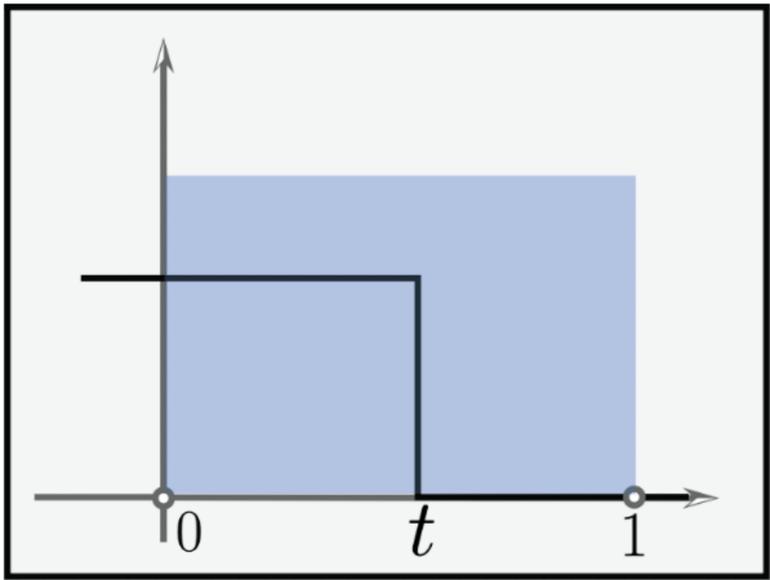
$$\int_0^1 [x < t] dx = \begin{cases} 0 & \text{if } t \leq 0 \\ t & \text{if } 0 < t < 1 \\ 1 & \text{if } 1 \geq t \end{cases}$$



$$\frac{d}{dt} \int_0^1 [x < t] dx = \begin{cases} 0 & \text{if } t \leq 0 \\ 1 & \text{if } 0 < t < 1 \\ 0 & \text{if } 1 \geq t \end{cases}$$

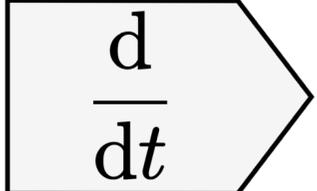
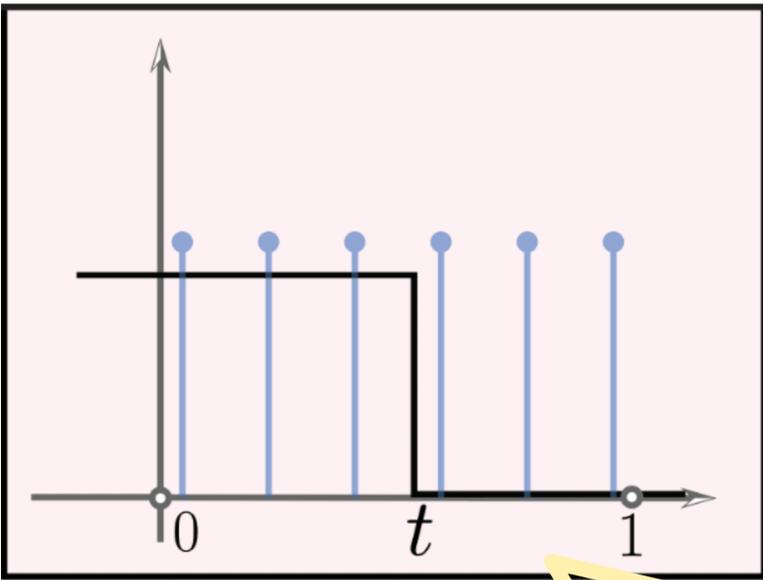
Naïve autodiff of integrals with derivatives

$$\int_0^1 [x < t] dx$$



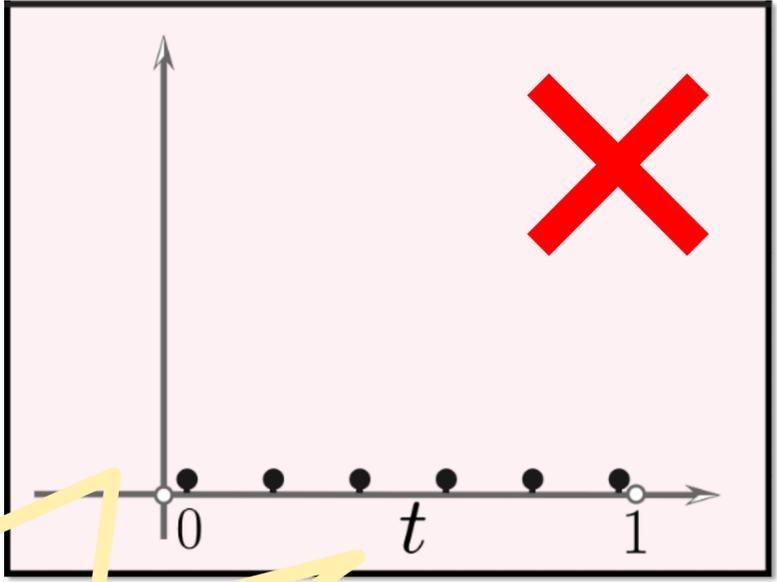
DISCRETIZE

$$\sum_{i=0}^N \frac{1}{N} \left[\frac{i}{N} < t \right]$$



DIFFERENTIATE

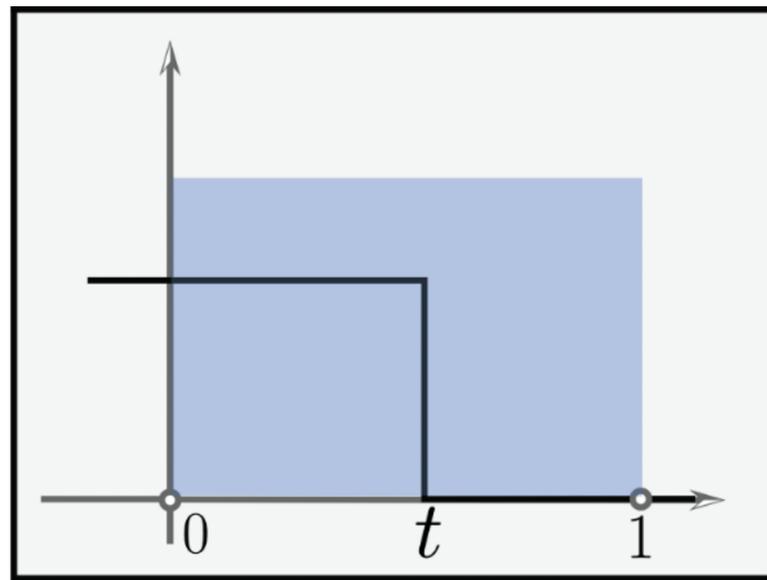
$$\sum_{i=0}^N 0$$



$$[0 < t < 1]$$

Correct derivatives of integrals with discontinuities

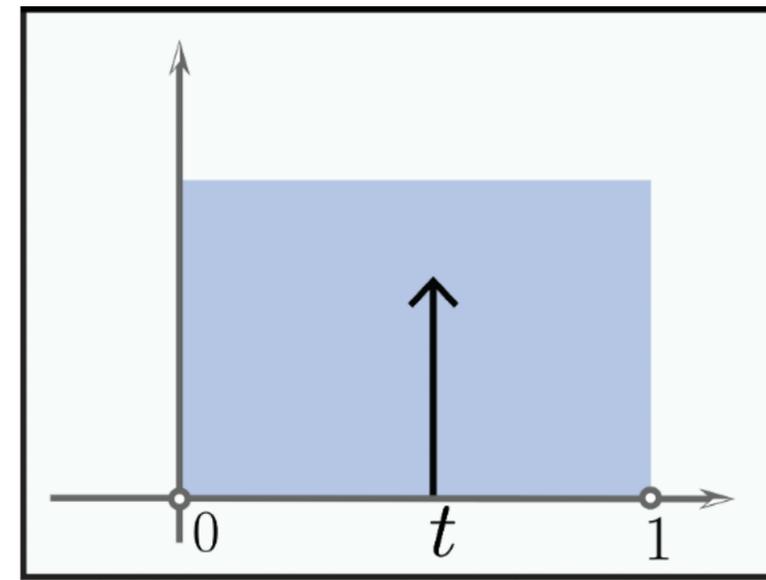
$$\int_0^1 [x < t] dx$$



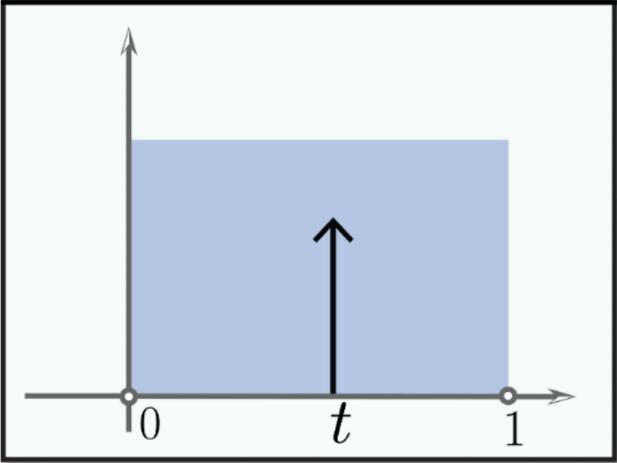
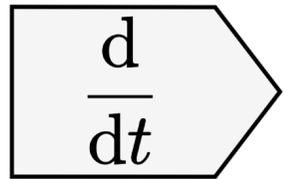
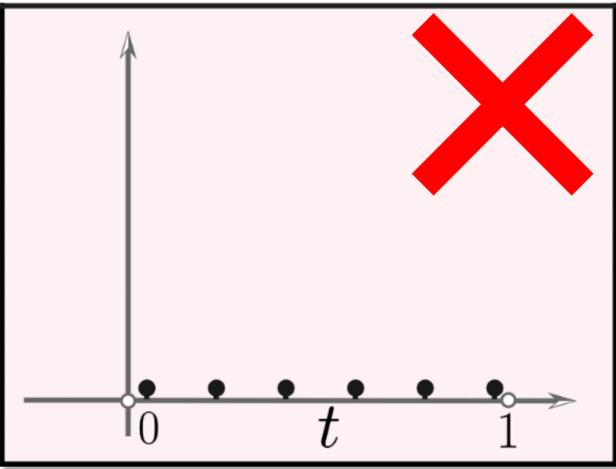
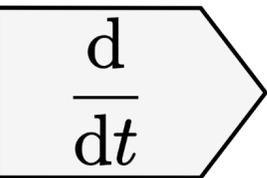
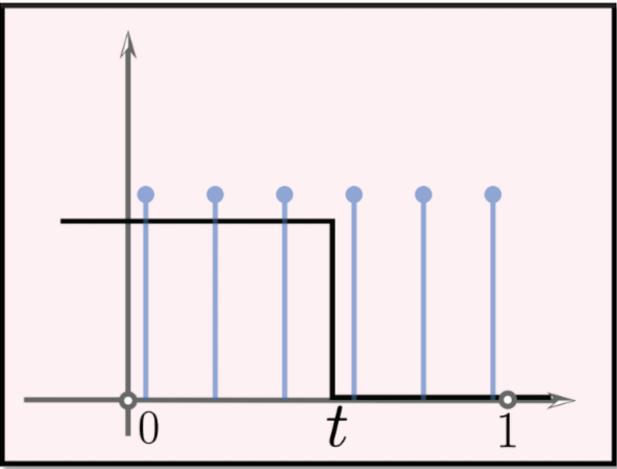
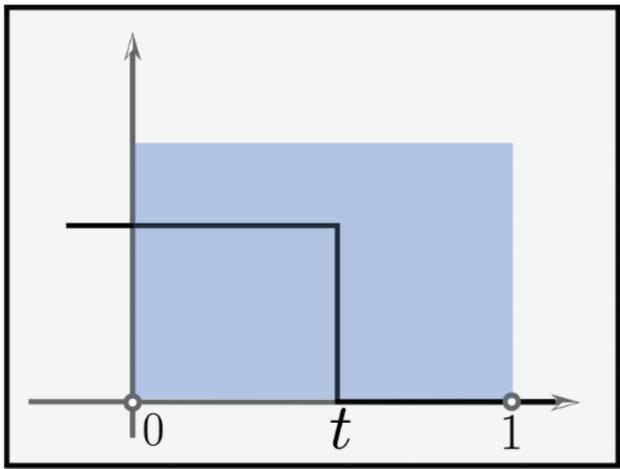
$$\frac{d}{dt}$$

DIFFERENTIATE

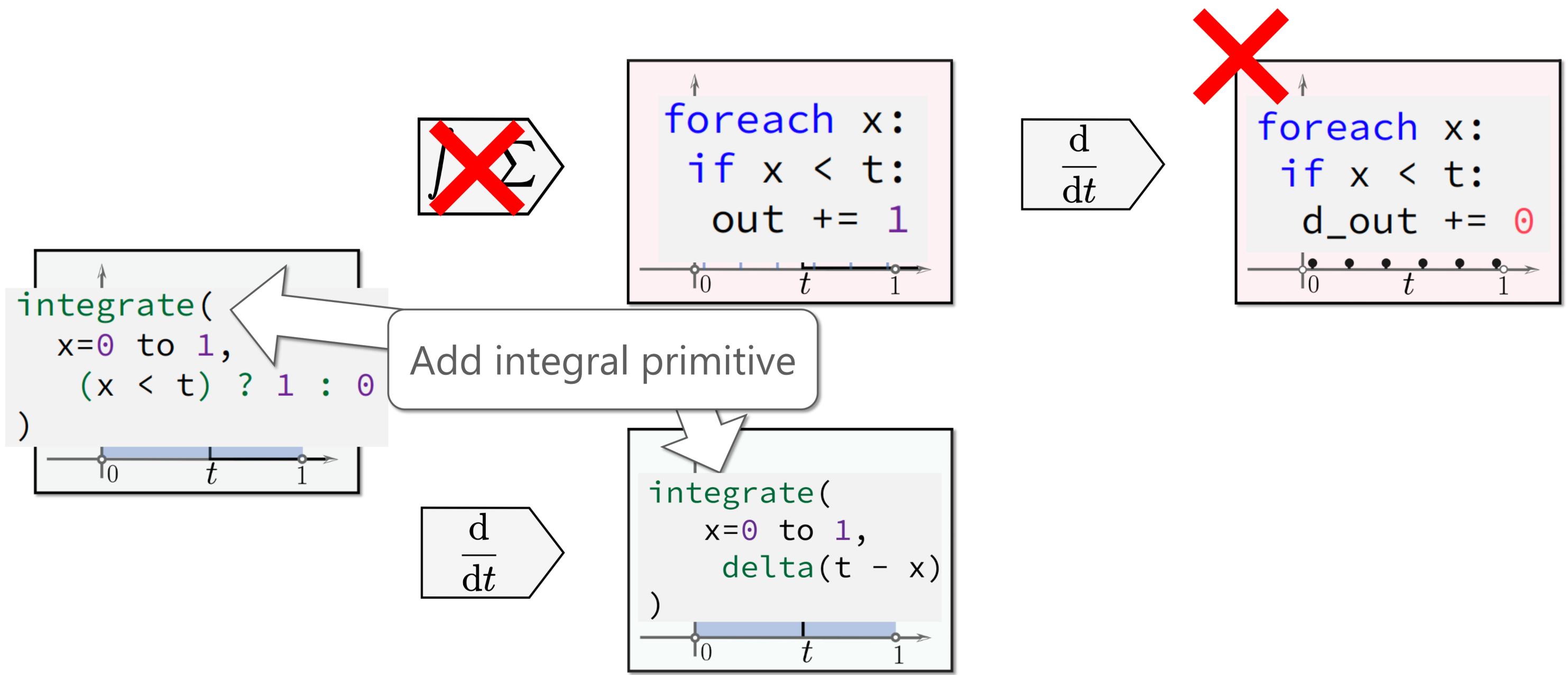
$$\int_0^1 \delta(t - x) dx$$



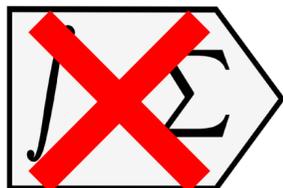
Integrals with discontinuities break auto-diff



The need for an integral primitive



Discontinuities now need a delta operation



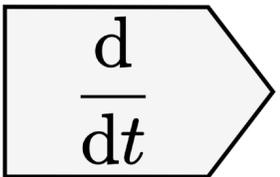
Not a well-defined operation



```
foreach x:  
  if x < t:  
    d_out += 0
```

```
integrate(  
  x=0 to 1,  
  (x < t) ? 1 : 0  
)
```

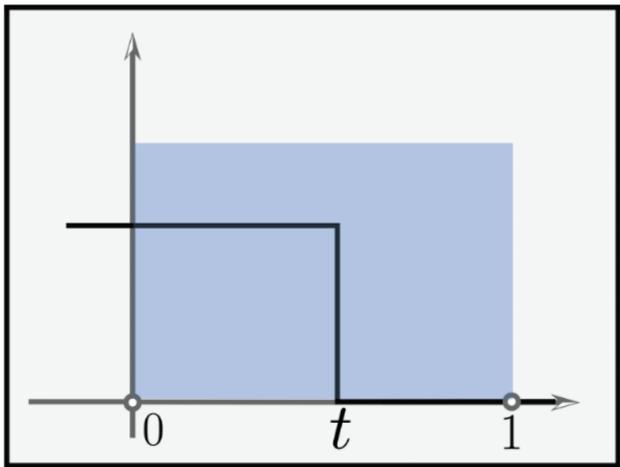
Add a delta operator



```
integrate(  
  x=0 to 1,  
  delta(t-x)  
)
```

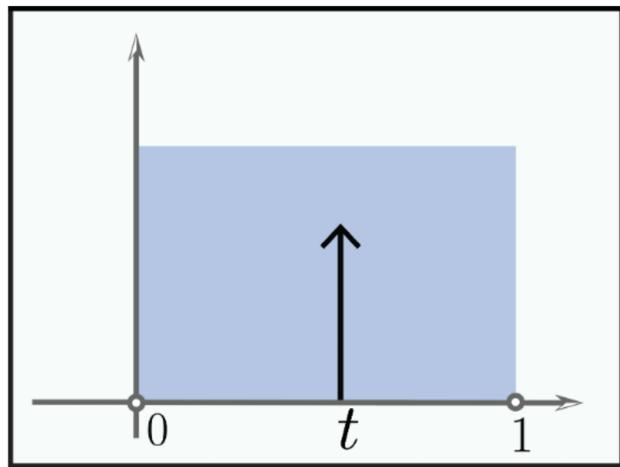
Recap: Differentiate first, then discretize

$$\int_0^1 [x < t] dx$$



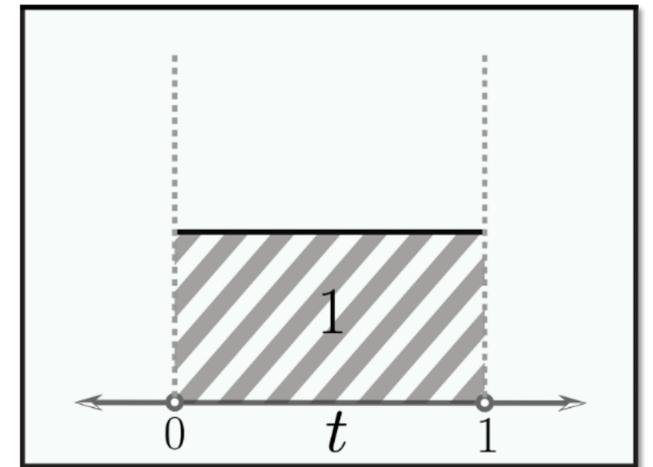
$\frac{d}{dt}$

$$\int_0^1 \delta(t - x) dx$$



δ

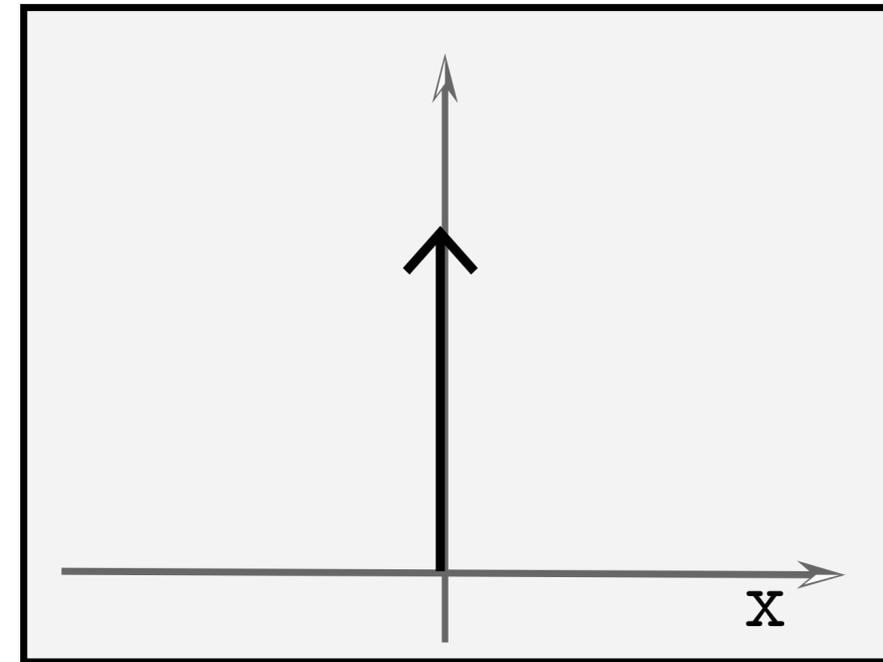
$[0 < t < 1]$



Eliminating deltas with the Sifting property

$$\int_{x=a}^b \delta(\mathbf{x}) \mathbf{f}(\mathbf{x}) = [a < 0 < b] \cdot \mathbf{f}(0)$$

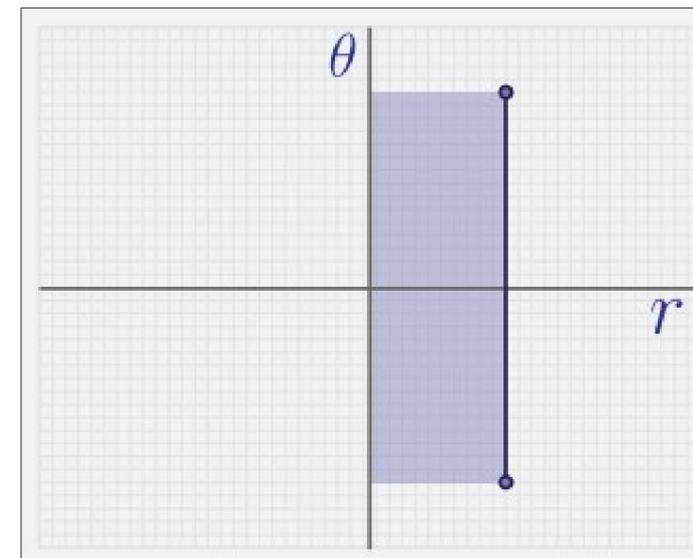
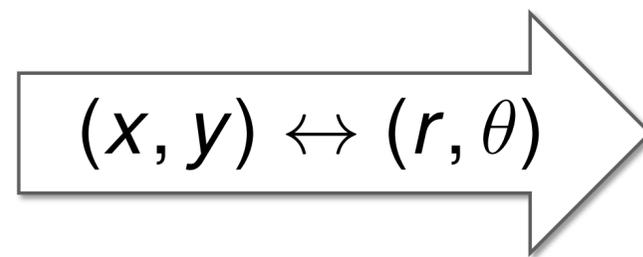
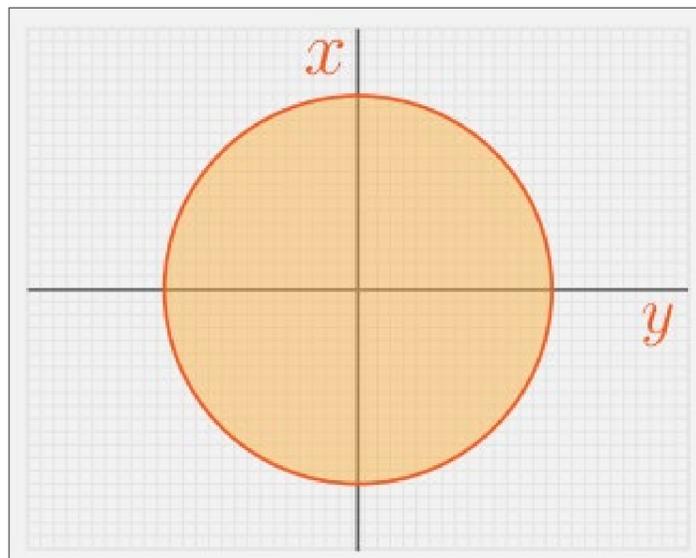
Single variable of integration



Pass 2: Delta Normalization

$$\int_{-1}^1 \int_{-1}^1 \delta(2x^2 + 2y^2 - t) dy dx$$

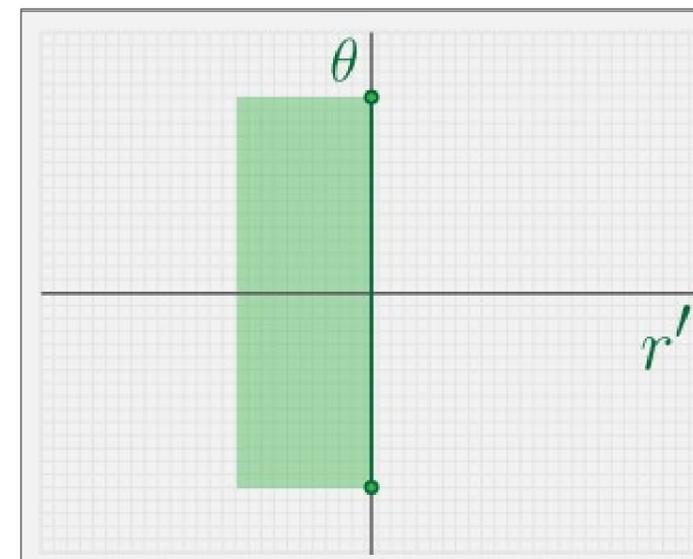
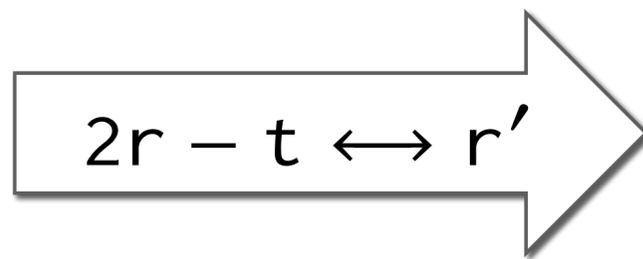
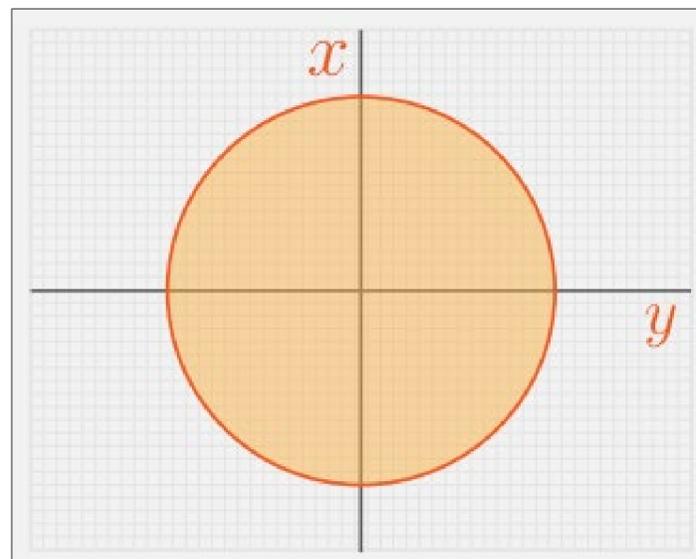
Not in normal form



Pass 2: Delta Normalization

$$\int_{-1}^1 \int_{-1}^1 \delta(2r - t) dy dx$$

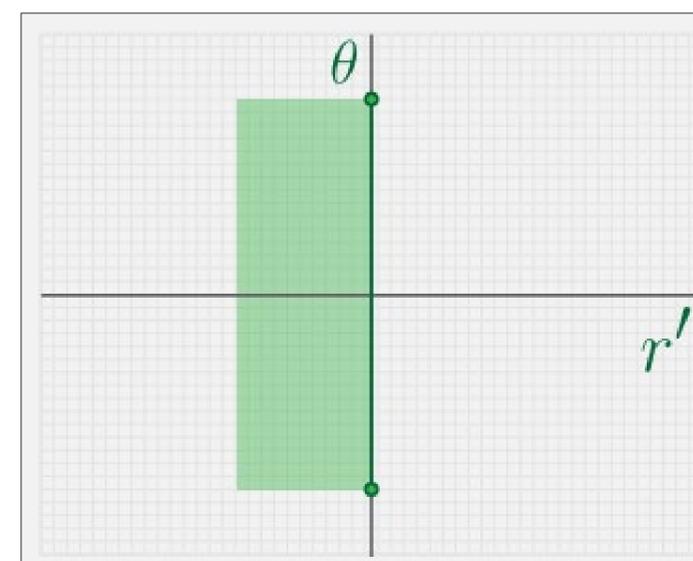
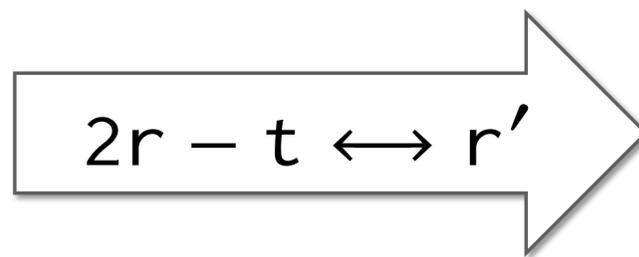
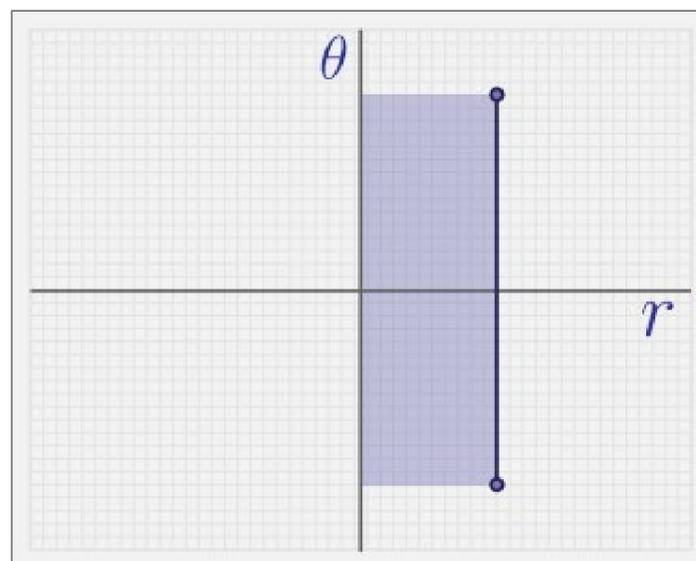
Still not in normal form



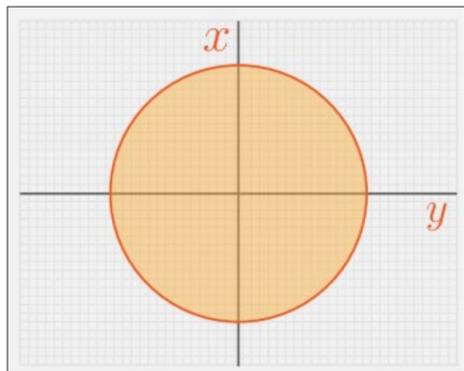
Pass 2: Delta Normalization

$$\int_{-1}^1 \int_{-1}^1 \delta(r') \, dydx$$

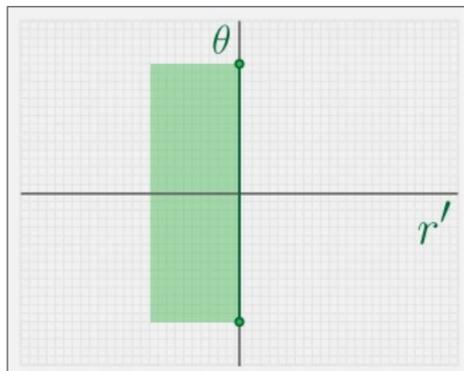
Final coordinates are in normal form!



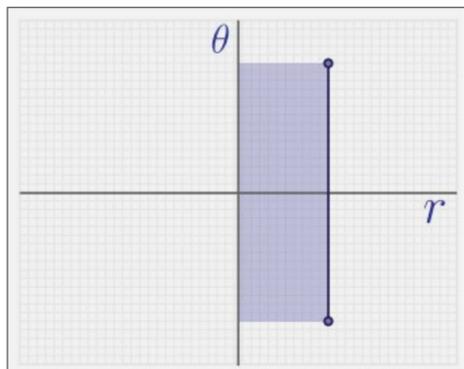
Pass 2: Delta Normalization



$$\int_{-1}^1 \int_{-1}^1 \delta(2x^2 + 2y^2 - t) dy dx$$



$$\int_0^{2\pi} \int_0^1 \delta(2r - t) r dr d\theta$$



$$\int_0^{2\pi} \int_{-t}^{2-t} \delta(r') (r' + t)/2 (dr'/2) d\theta$$

Pass 2: Delta Normalization

$$\int_0^{2\pi} \int_0^1 \delta(2r - t) r dr d\theta$$

$2r - t \longrightarrow r'$

$$\int_0^{2\pi} \int_{-t}^{2-t} \delta(r') (r' + t)/2 (dr'/2) d\theta$$

Pass 2: Delta Normalization

$$\int_0^{2\pi} \int_0^1 \delta(2r - t) r dr d\theta$$

$$r \longrightarrow (r' + t)/2$$

Inverse

$$\int_0^{2\pi} \int_{-t}^{2-t} \delta(r') (r' + t)/2 (dr'/2) d\theta$$

Pass 2: Delta Normalization

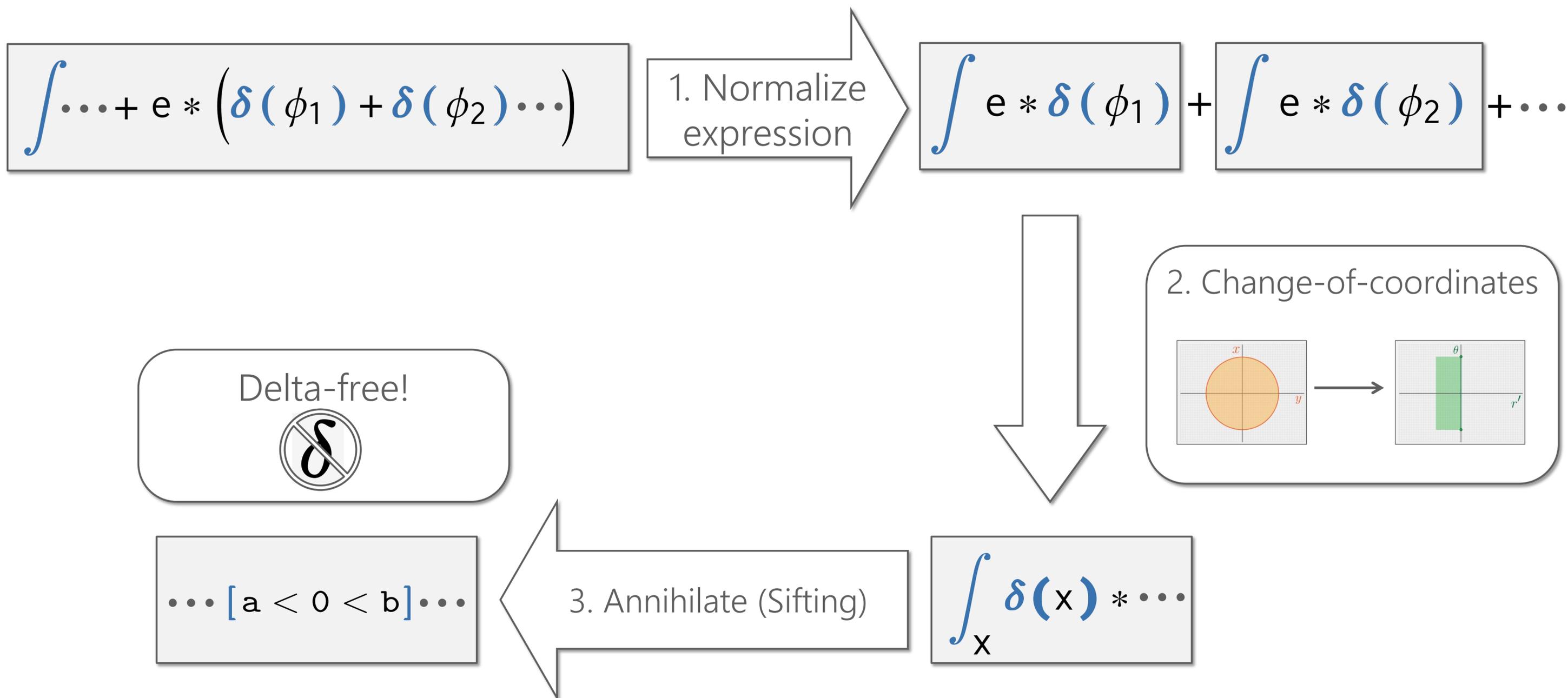
$$\int_0^{2\pi} \int_0^1 \delta(2r - t) r dr d\theta$$

$$dr \longrightarrow dr'/2$$

Derivative

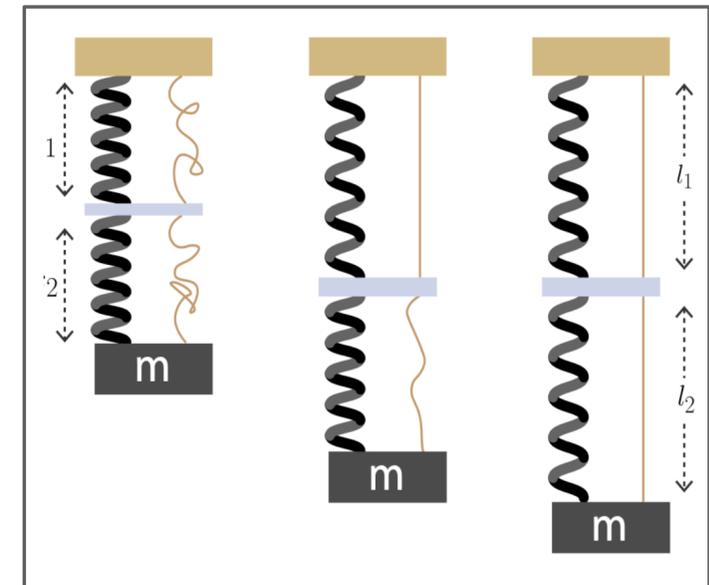
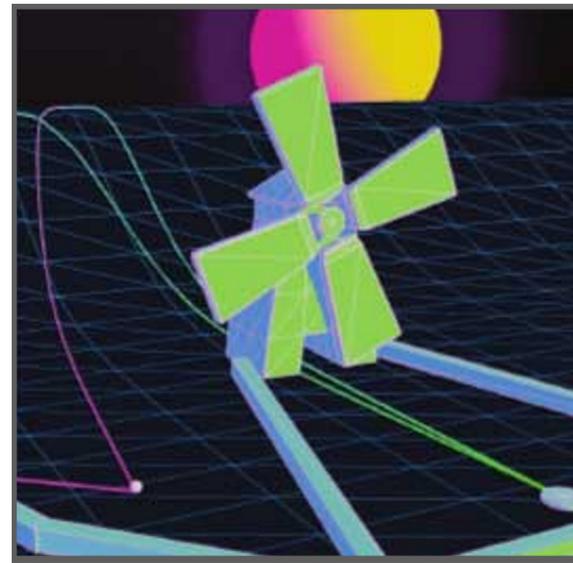
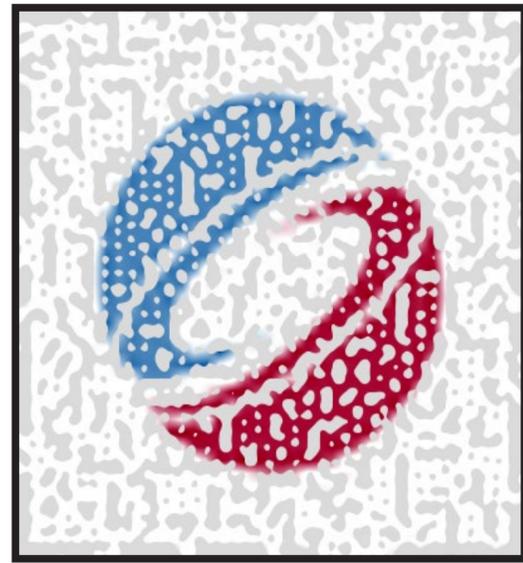
$$\int_0^{2\pi} \int_{-t}^{2-t} \delta(r') (r' + t)/2 (dr'/2) d\theta$$

All passes together



Systems and Applications

Applications

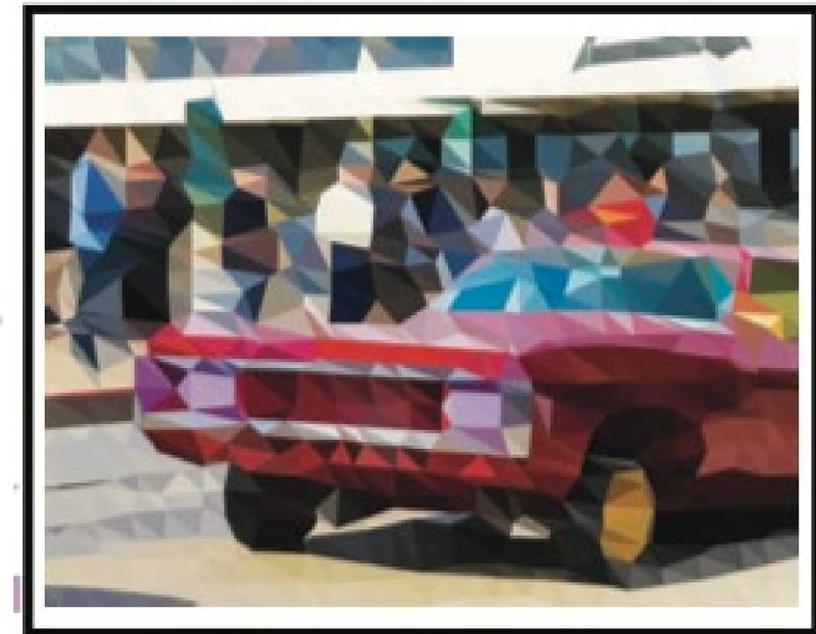
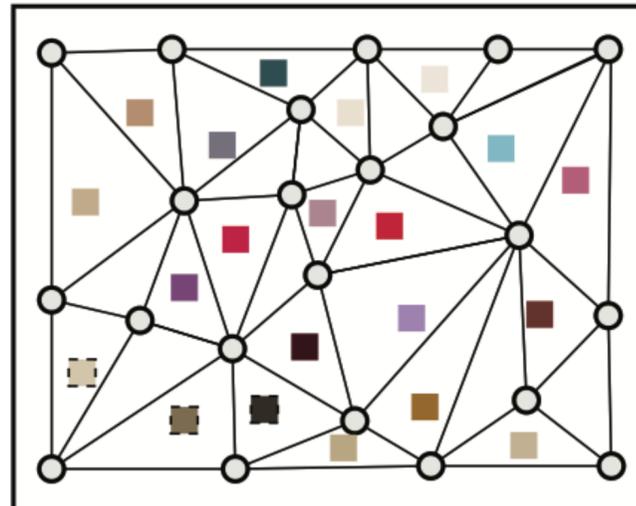


Application: Image Style Filters



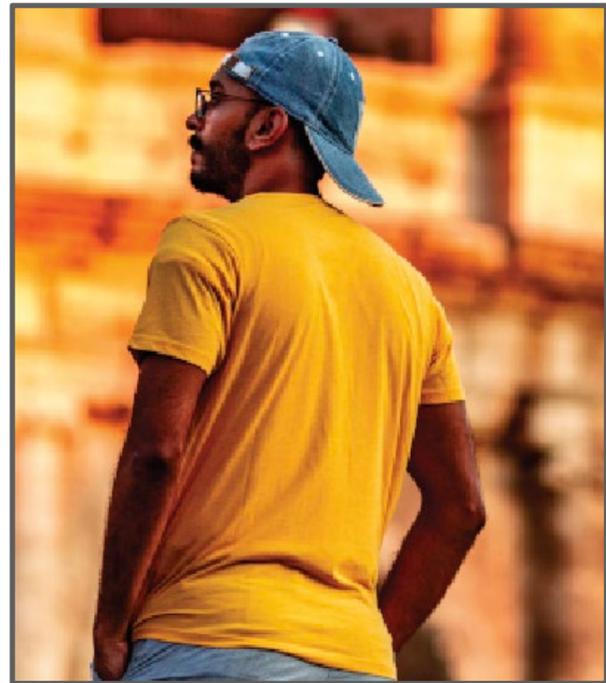
Target image

rasterize(t)



Triangulated image

Ignoring δ -terms produces 0 gradient



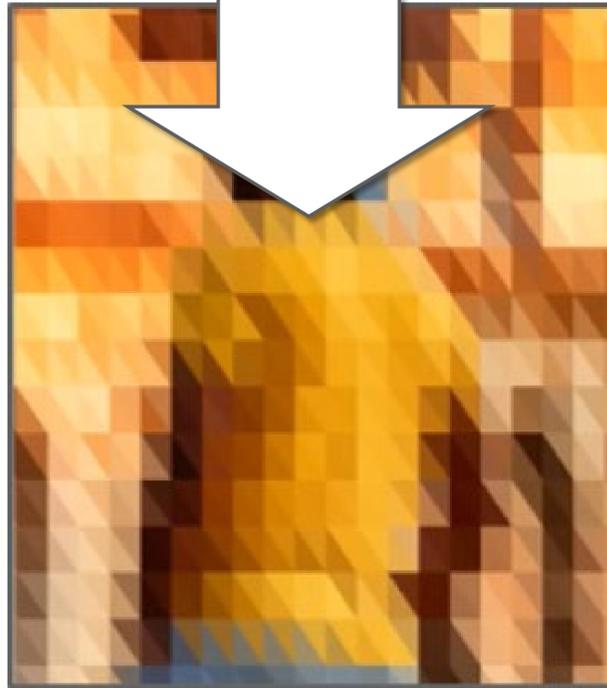
Target Image

δ s account for discontinuities



Optimize with Ours

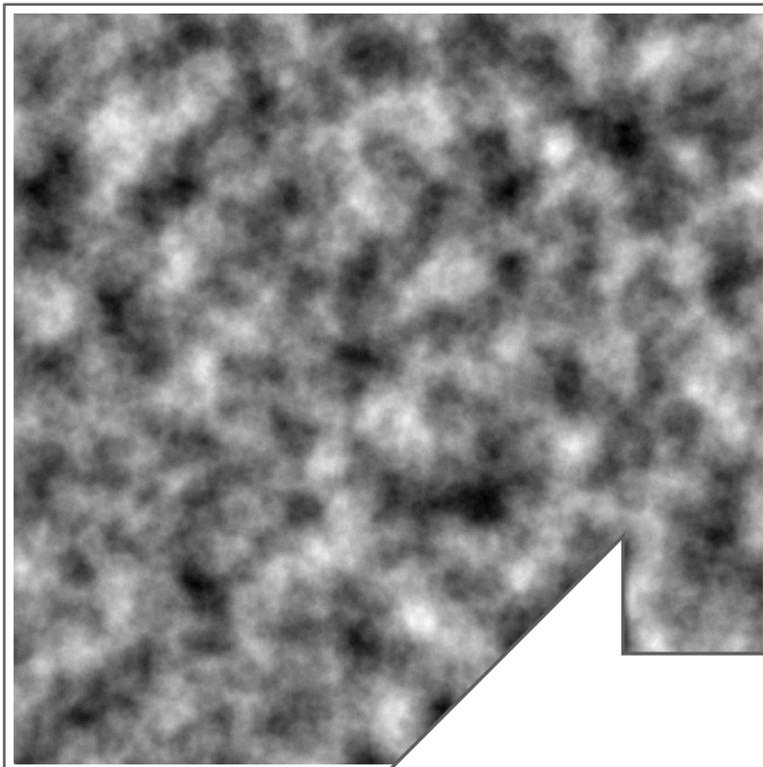
No gradient



Optimize with Traditional Auto-diff

Thresholded noise shaders

Perlin noise map



Thresholded noise
(Discontinuous)

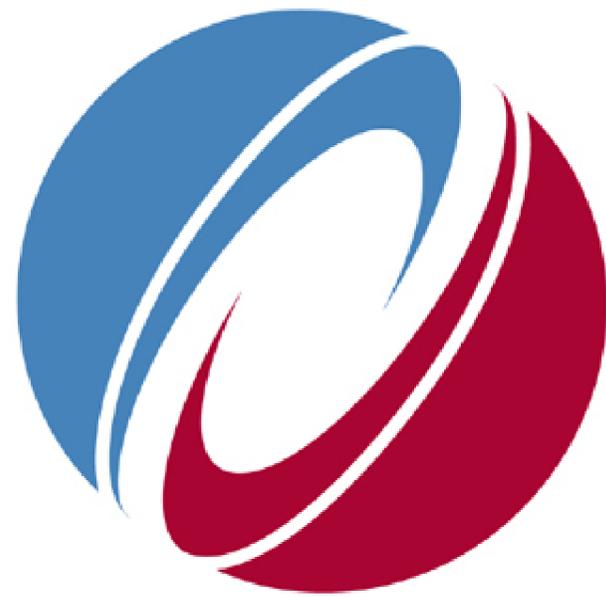


Threshold
... * $[noise > t]$ * ...

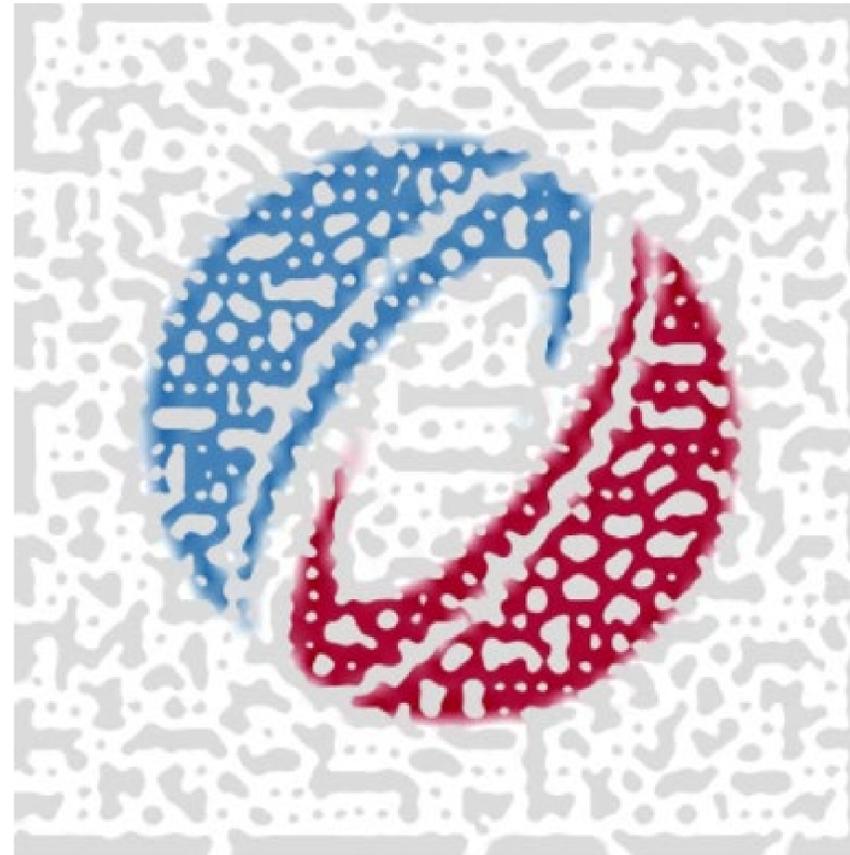
Inverse problem:
Optimize *noise* to fit a *target pattern*

Bozo's Donut
(Perlin 1985)

Inverse shader design using our approach

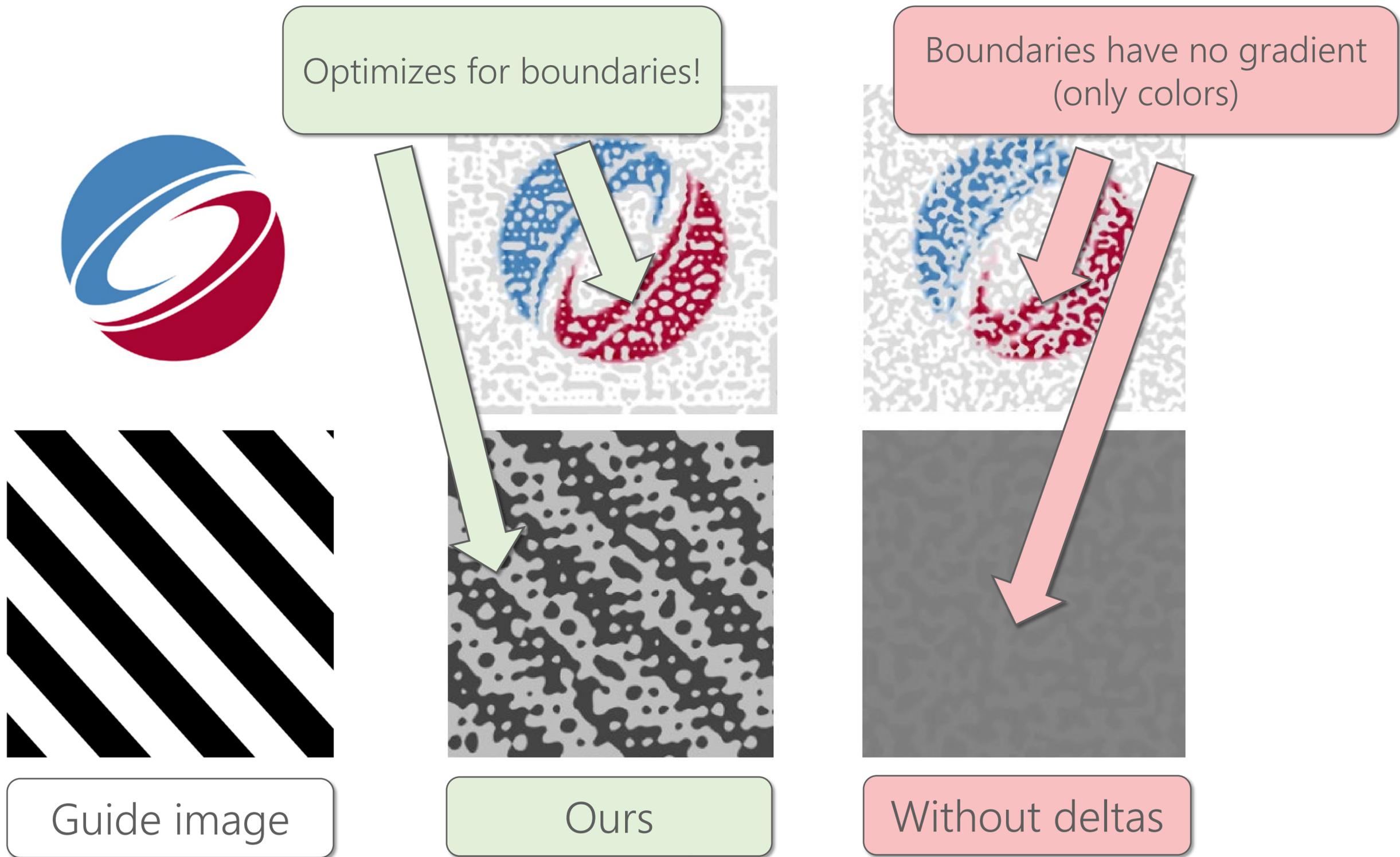


Guide image

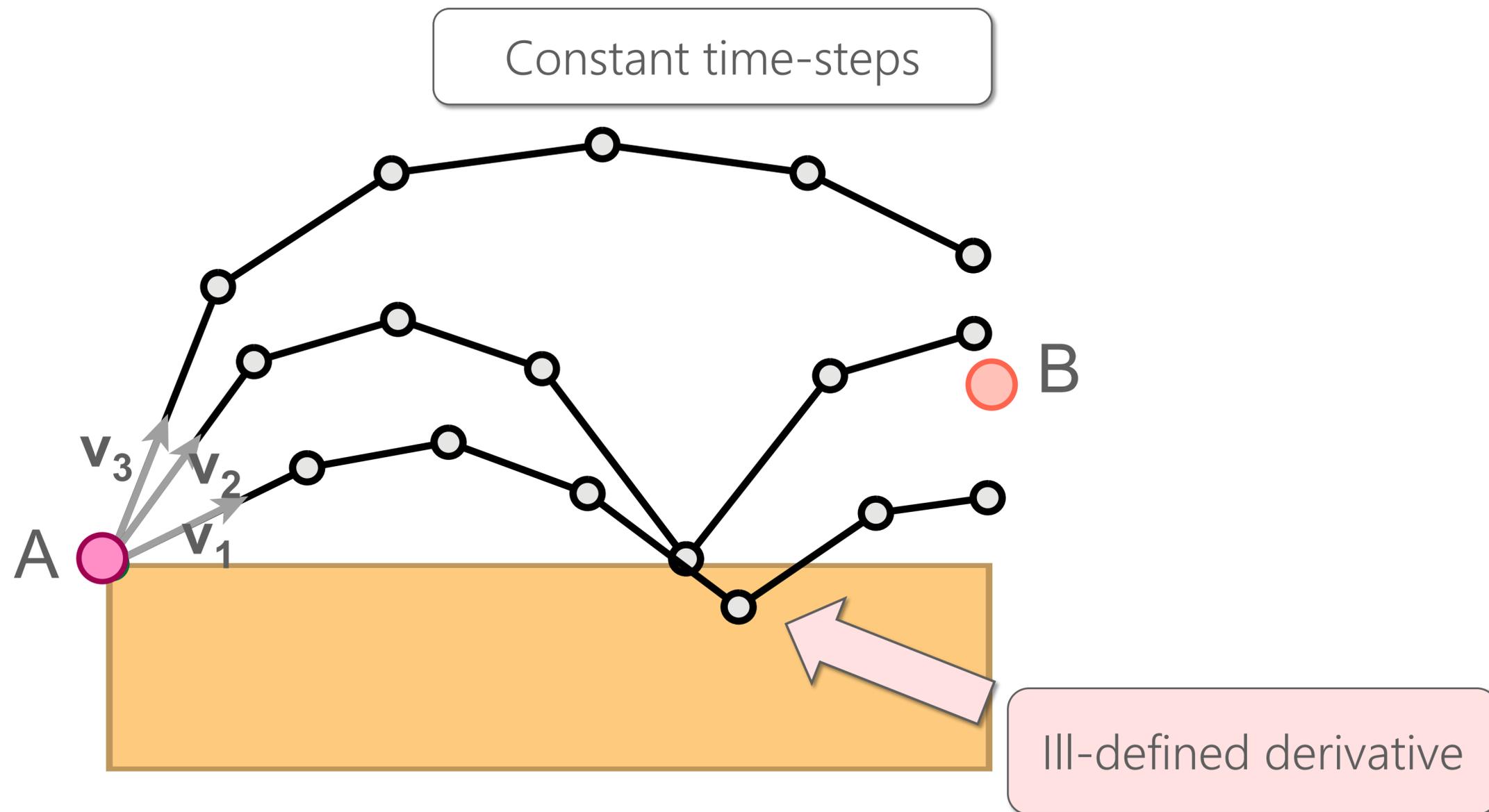


Optimized with
Ours

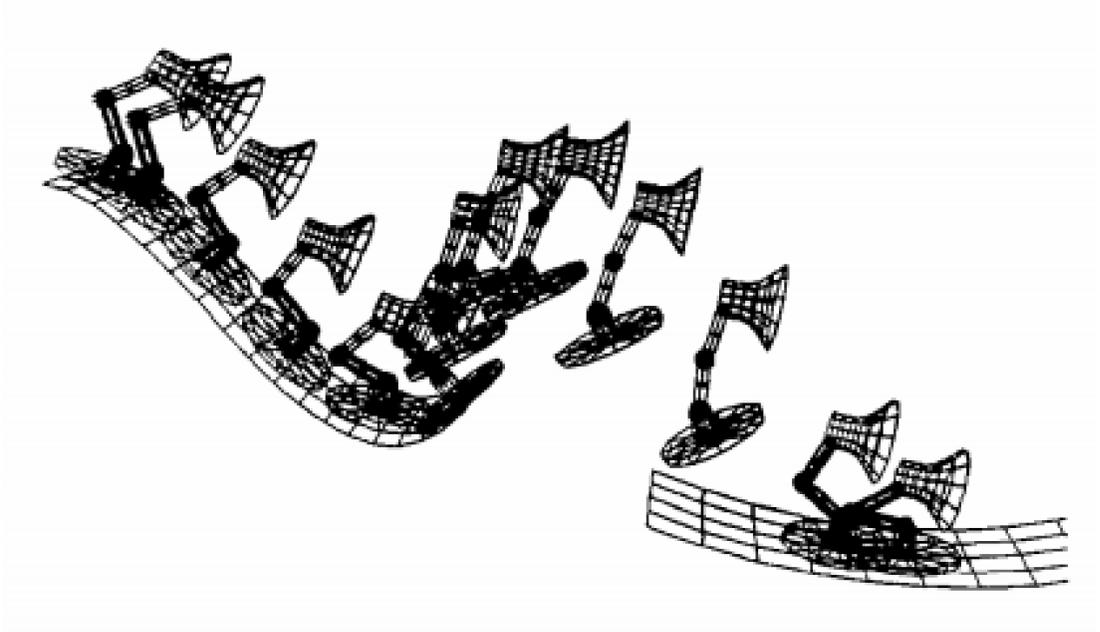
Ignoring delta terms produces incorrect results!



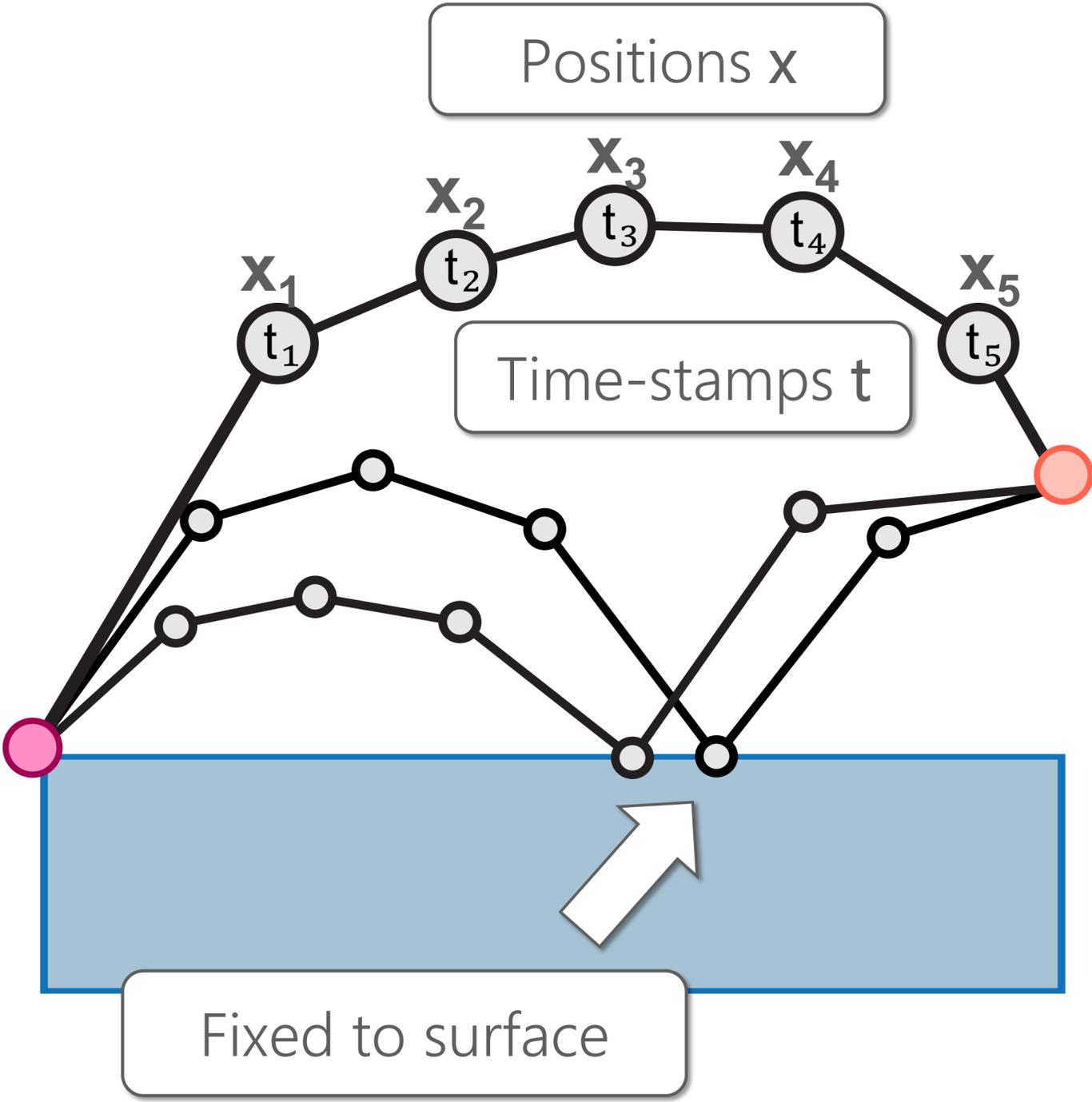
Application: Animation with hard contact



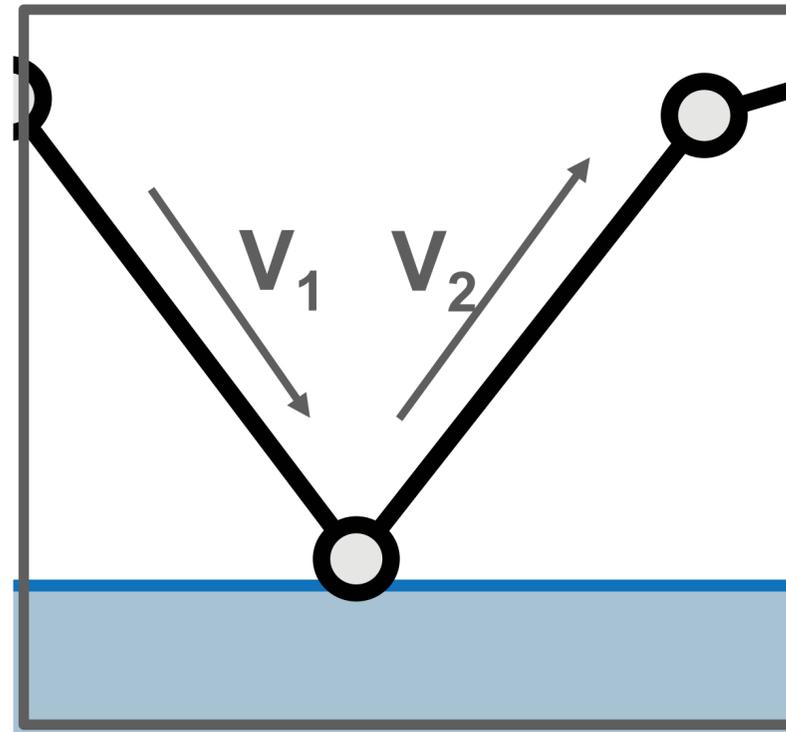
Inverse physical simulation with *space-time constraints*



Spacetime constraints (1988)

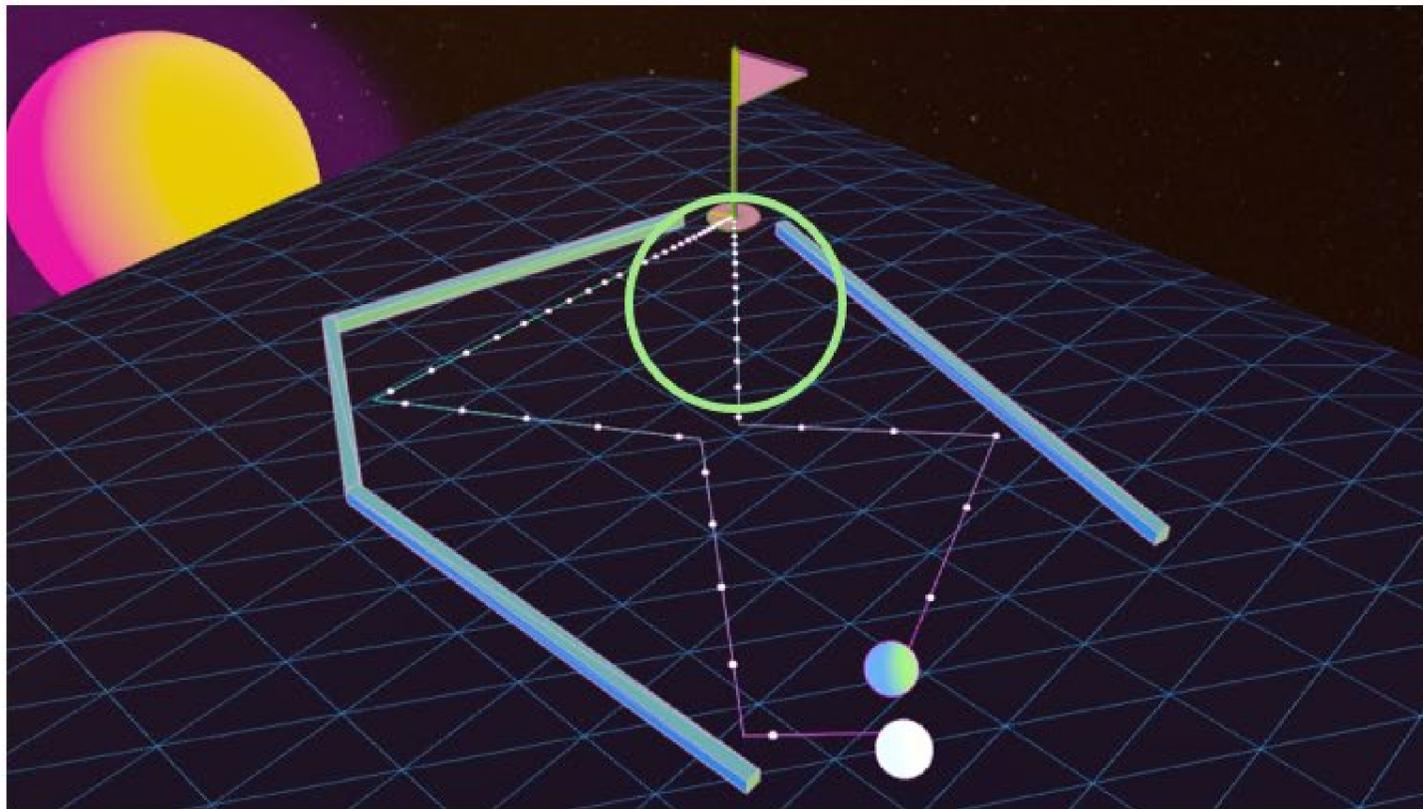


Physical simulation with space-time constraints

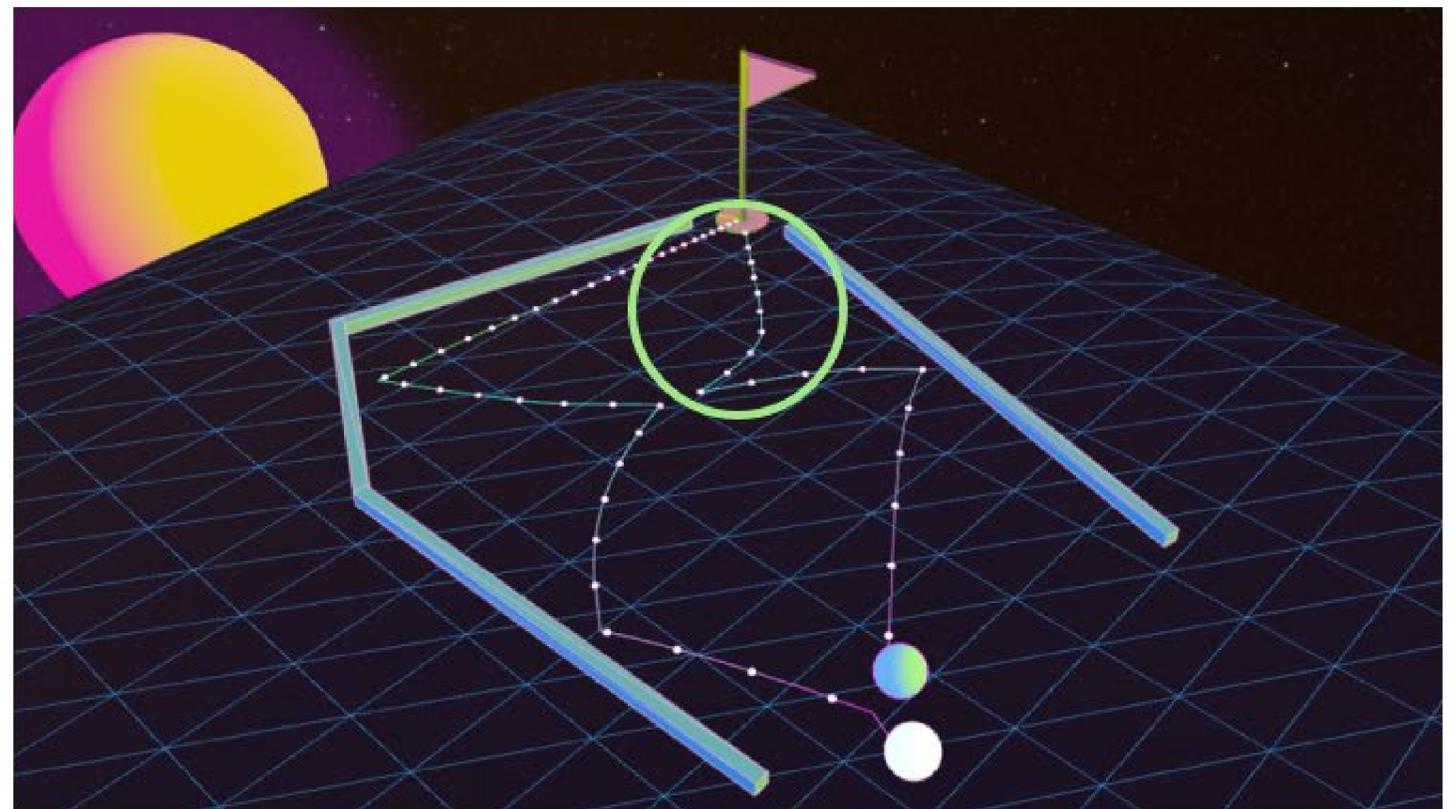


Discontinuous
velocity

Comparison with ignoring boundaries



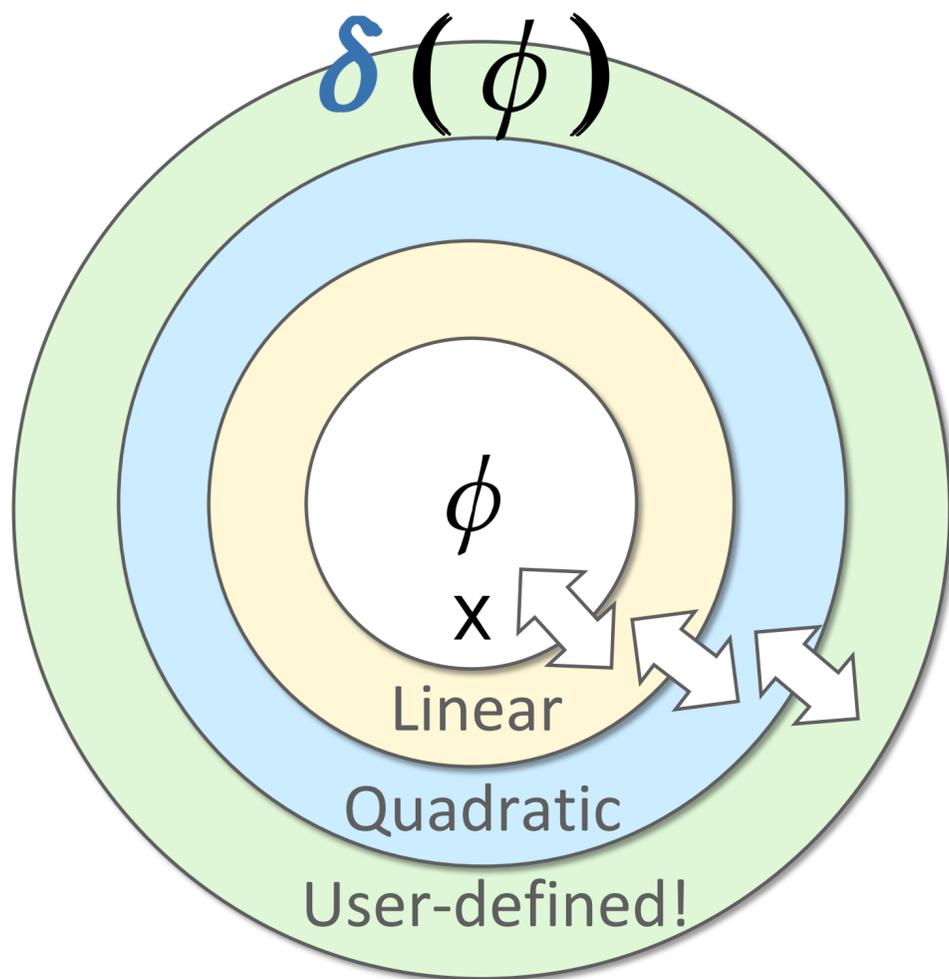
Optimized with Ours
(Physically *correct*)



Optimized with Traditional Auto-diff
(Physically *incorrect*)

Limitations & Future Directions

Diffeomorphisms

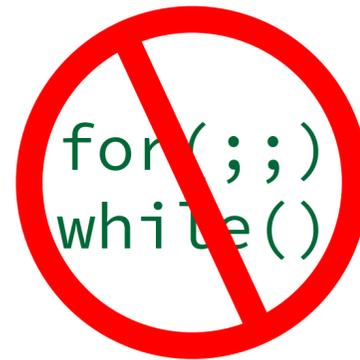


Scalability

Scalar variables
(no indexing)



Expression lang
(no looping)



Modularity

$$\int_{x=a}^b \dots \delta(\phi)$$

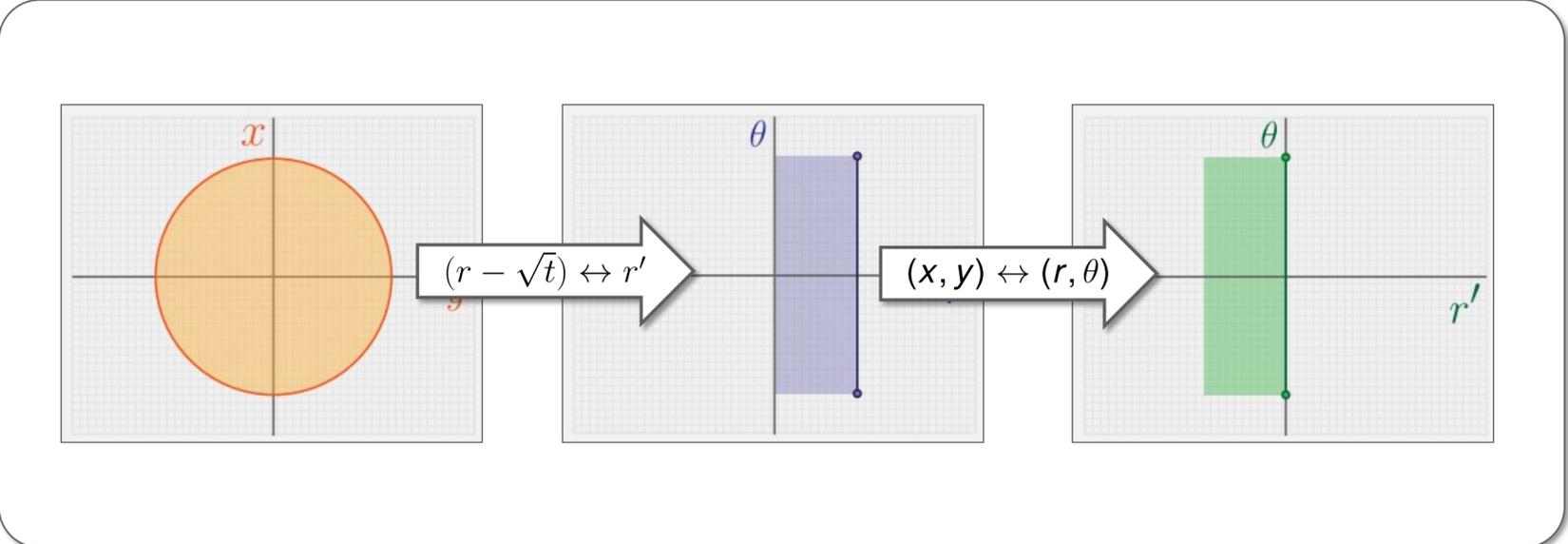
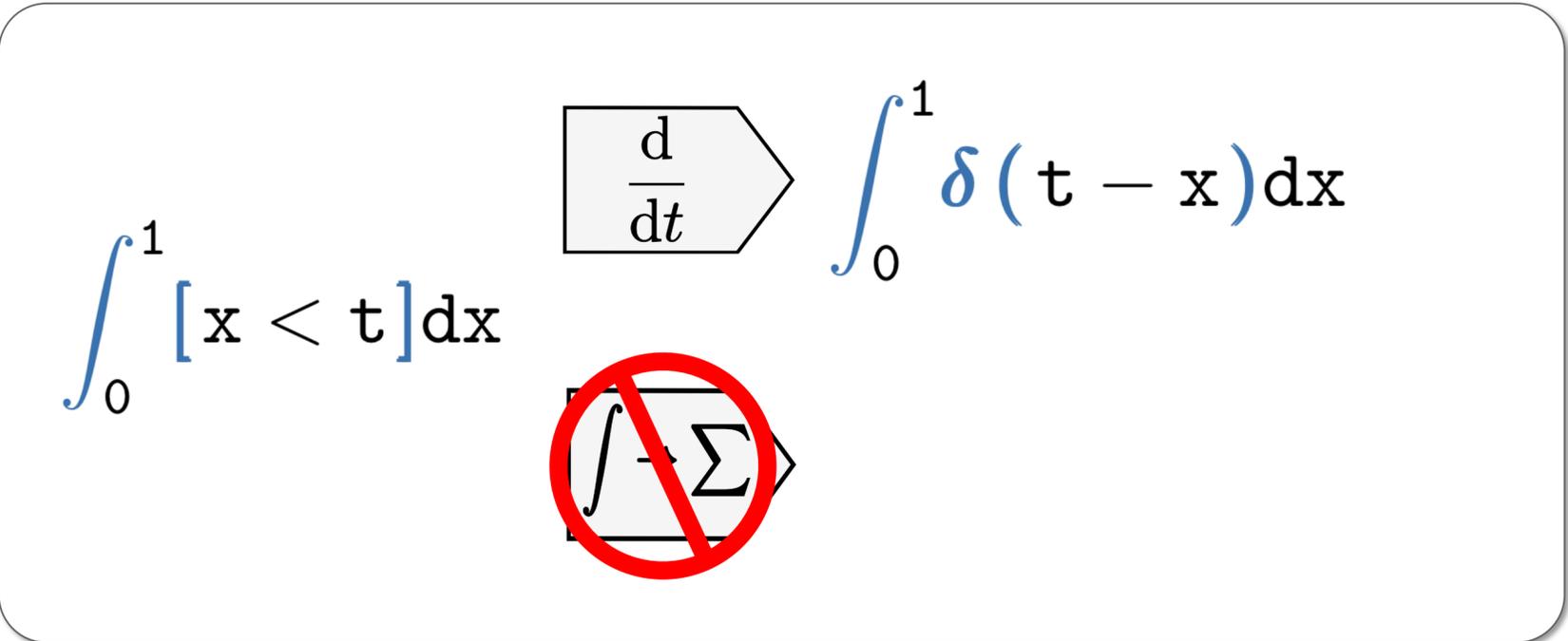
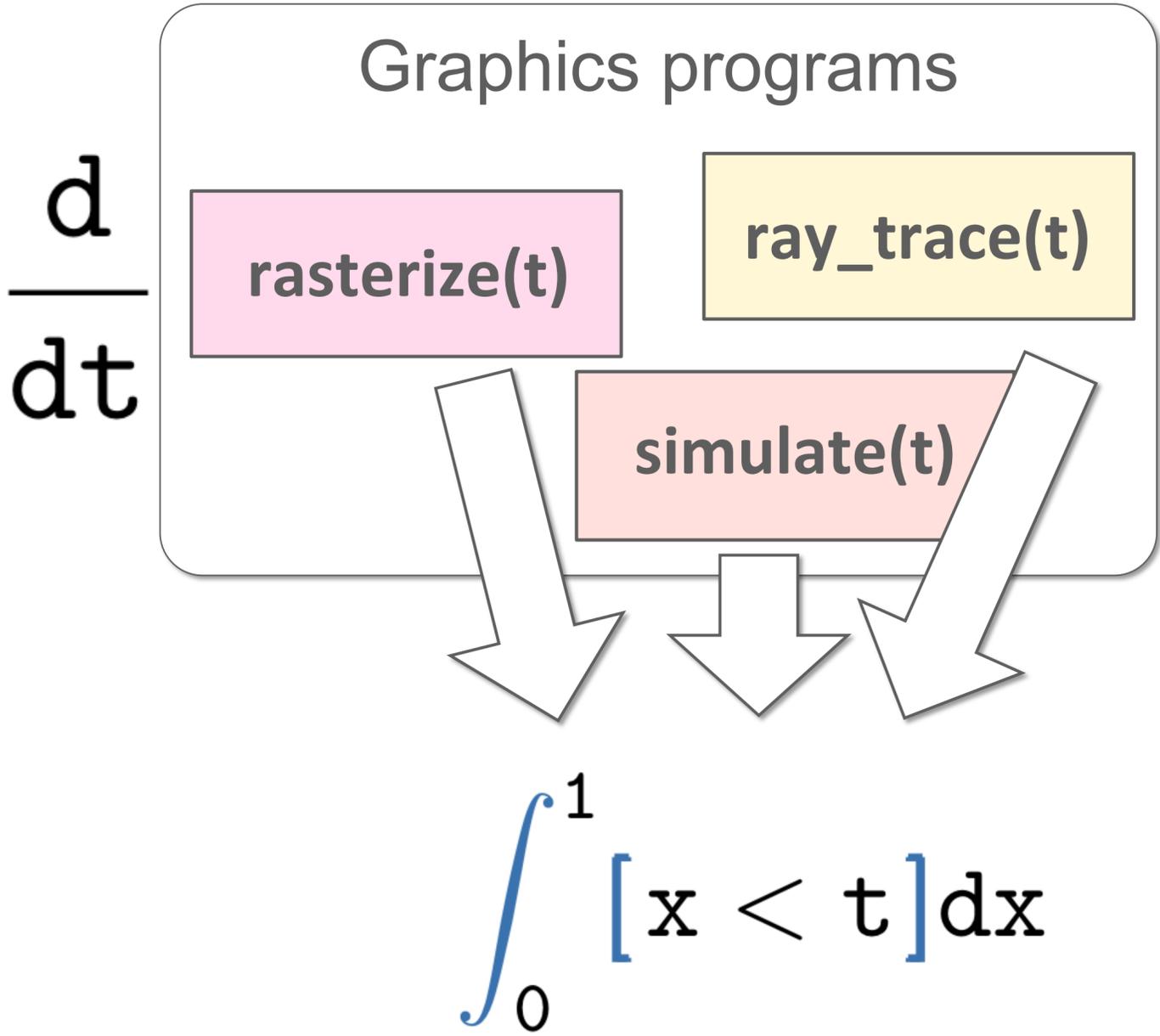
Requires global
transforms

$$\int e * (\delta(\phi_1) + \delta(\phi_2))$$

Code duplication

$$\int e * \delta(\phi_1) + e * \delta(\phi_2)$$

Systematically handling discontinuities: A Summary



Mitsuba 2

- A **general-purpose differentiable** renderer developed by Jakob et al.
- **Strengths**
 - Feature-rich (e.g., supports hyper-spectral and polarized rendering)
 - Efficient at handling many (e.g., millions) of parameters
- **Weaknesses**
 - Currently offers limited support for differentiation with respect to geometry



PSDR-CUDA

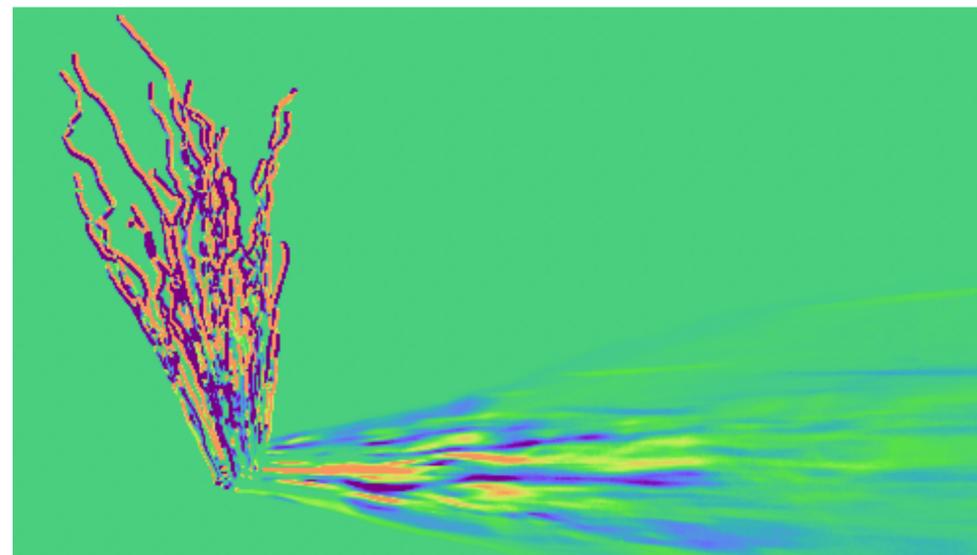
- A GPU-based **general-purpose differentiable** renderer
- Built upon the same numerical backend (i.e., Enoki) as Mitsuba 2
 - Much lighter weighted
 - Python bindings via pybind11
- Implements **path-space differentiable** path tracing [Zhang et al. 2020, 2021]
 - Fast and unbiased geometric gradients



Try PSDR-CUDA!



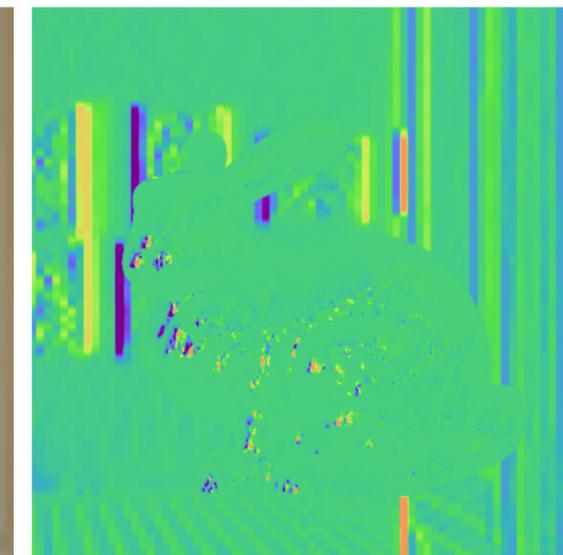
Original image



Derivative image
(w.r.t. rotation of the object)



Original image



Derivative image
(w.r.t. rotation of the env. map)

Differentiating Image RMSE using PSDR-CUDA

```
import enoki as ek
from enoki.cuda_autodiff import Float32 as FloatD, Vector3f as Vector3fD, Matrix4f as Matrix4fD
import psdr_cuda

# Load the scene without configuring it
scene = psdr_cuda.Scene()
scene.load_file('scene.xml', auto_configure=False)

# Compute gradient with respect to mesh vertex positions
ek.set_requires_gradient(scene.param_map["Mesh[0]"].vertex_positions)

# Configure the scene
scene.configure()

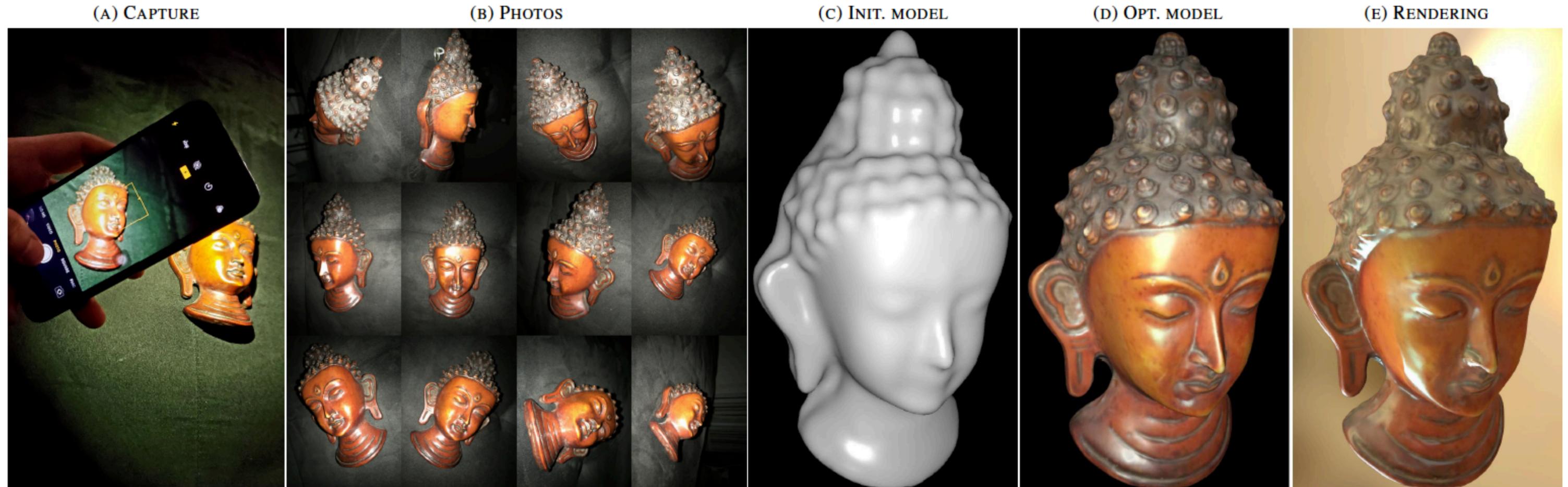
# Start rendering!
image = psdr_cuda.DirectIntegrator().renderD(scene, sensor_id=0)

# Compute the RMSE image loss
loss = ek.sqrt(ek.hmean(ek.squared_norm(target_image - image)))

# Reverse-mode autodiff
ek.backward(loss)

# Obtain the gradient of the loss
grad = ek.gradient(scene.param_map["Mesh[0]"].vertex_positions)
```

Application: Shape and Material Reconstruction



To appear at *Eurographics Symposium on Rendering (EGSR) 2021*

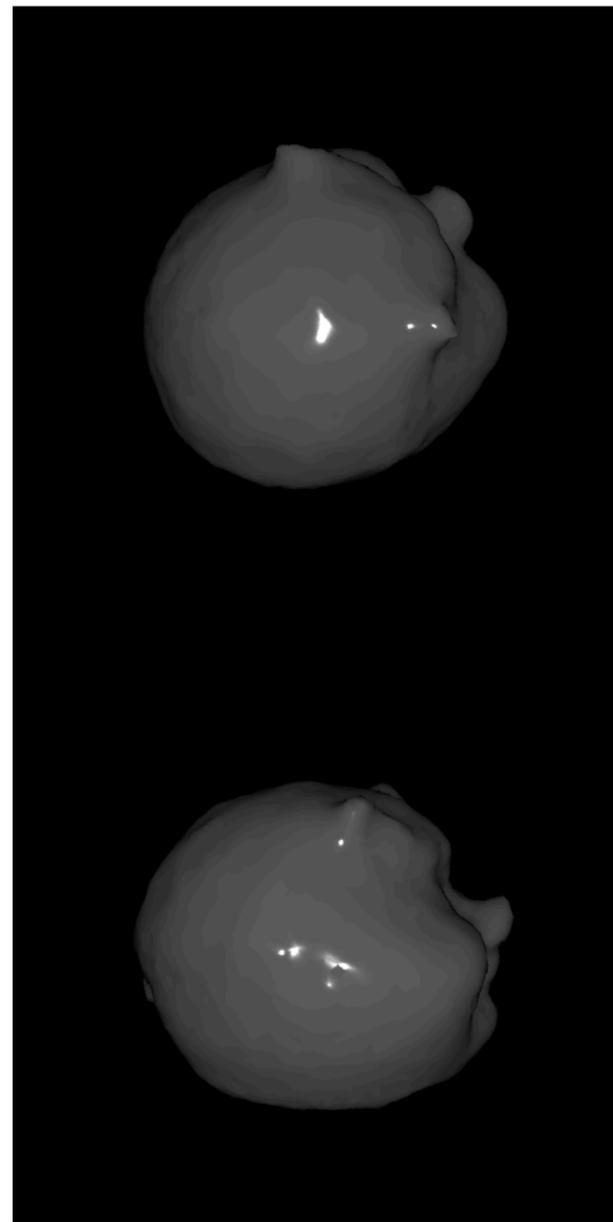
Joint work with Fujun Luan, Kavita Bala, and Zhao Dong

Application: Shape and Material Reconstruction

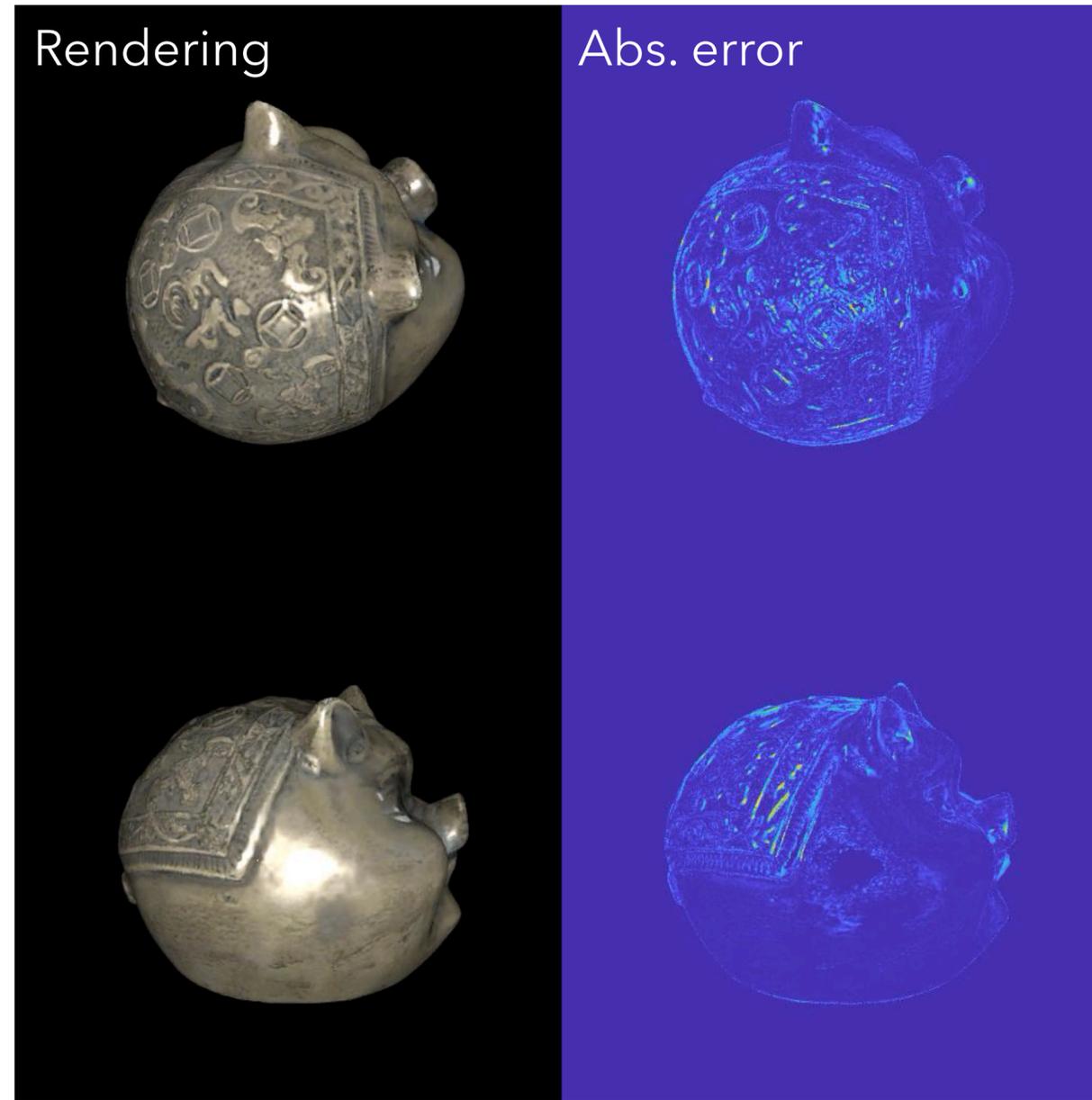
- We solve an **analysis-by-synthesis** problem by jointly optimizing:
 - Object shape (i.e., positions of all mesh vertices)
 - Object reflectance (as diffuse/specular albedo and roughness maps)
- > **1M** parameters!
- Losses:
 - **Rendering** loss (computed & differentiated using PSDR-CUDA)
 - Regularization losses (e.g., mesh Laplacian, map smoothness)
 - For improved **robustness**:
 - Coarse-to-fine scheme
 - When updating vertex positions, use elTopo [Brochu et al. 2009] to avoid self-intersections

Application: Shape and Material Reconstruction

Joint optimization of *object shape* and *spatially varying reflectance* (100 views used, 2 shown)



Initial (Kinect Fusion)



Optimized (using gradients generated by PSDR-CUDA)

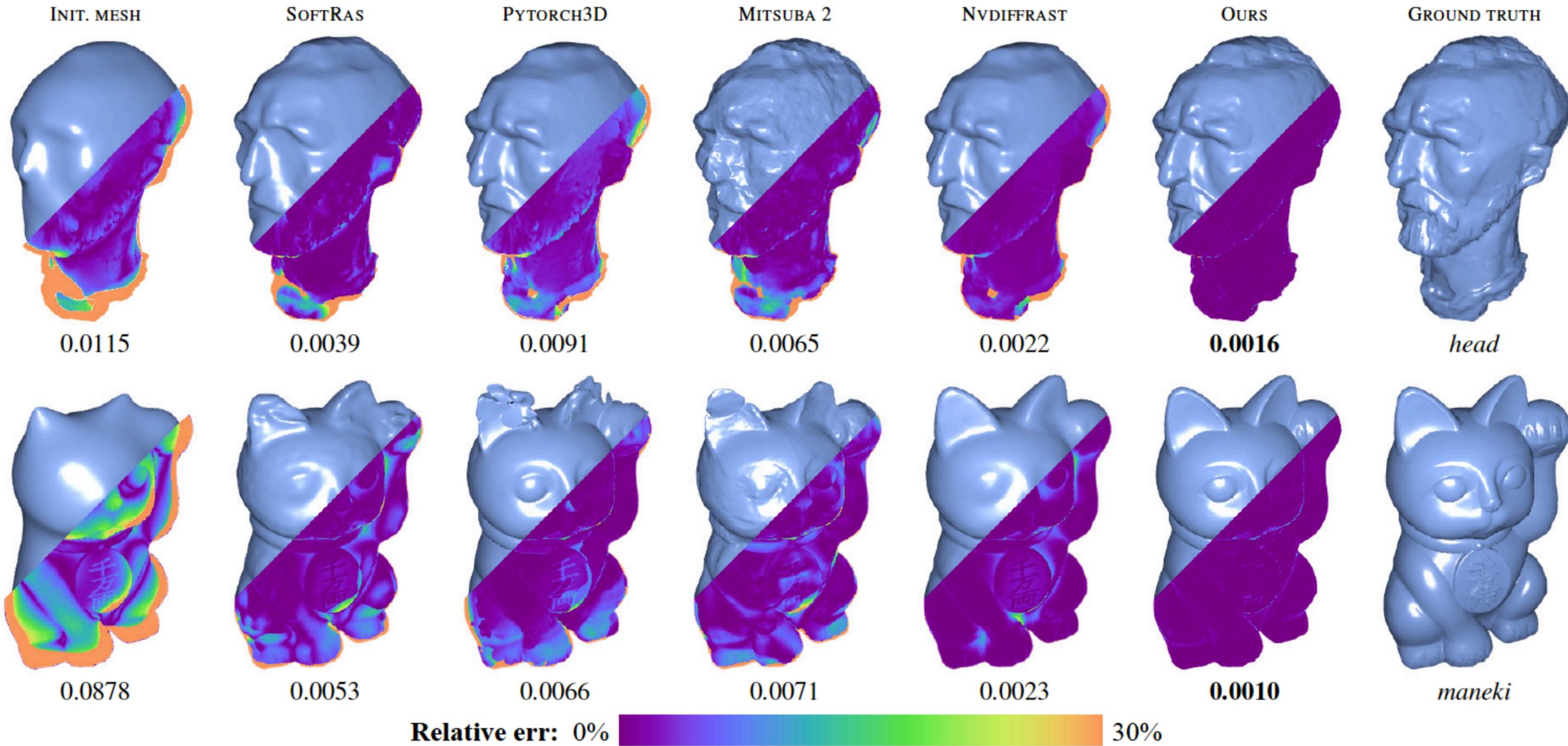


Target

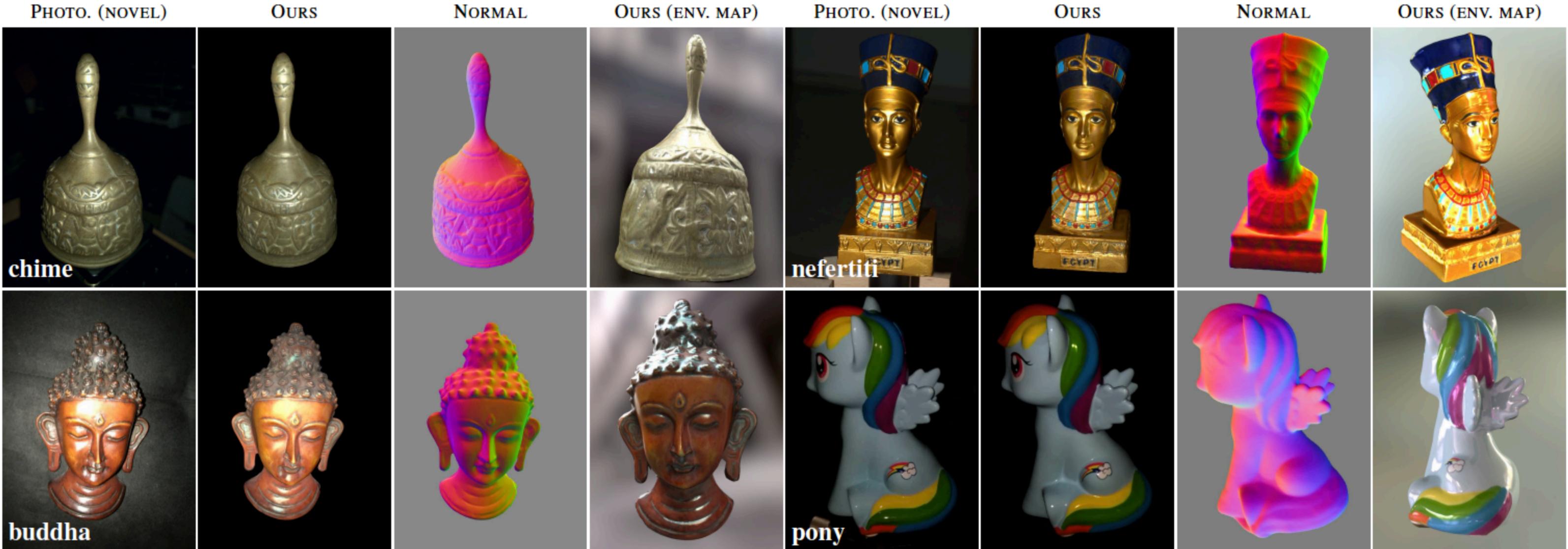


Gradient Accuracy Matters!

Inverse-rendering results with *identical* optimization settings



Reconstruction Results of Real Objects



(No normal mapping is used, all geometric details emerge from actual mesh geometries.)

Reconstruction Results of Real Objects

Re-rendering in
novel 3D scene



Object insertion in
augmented reality (AR)



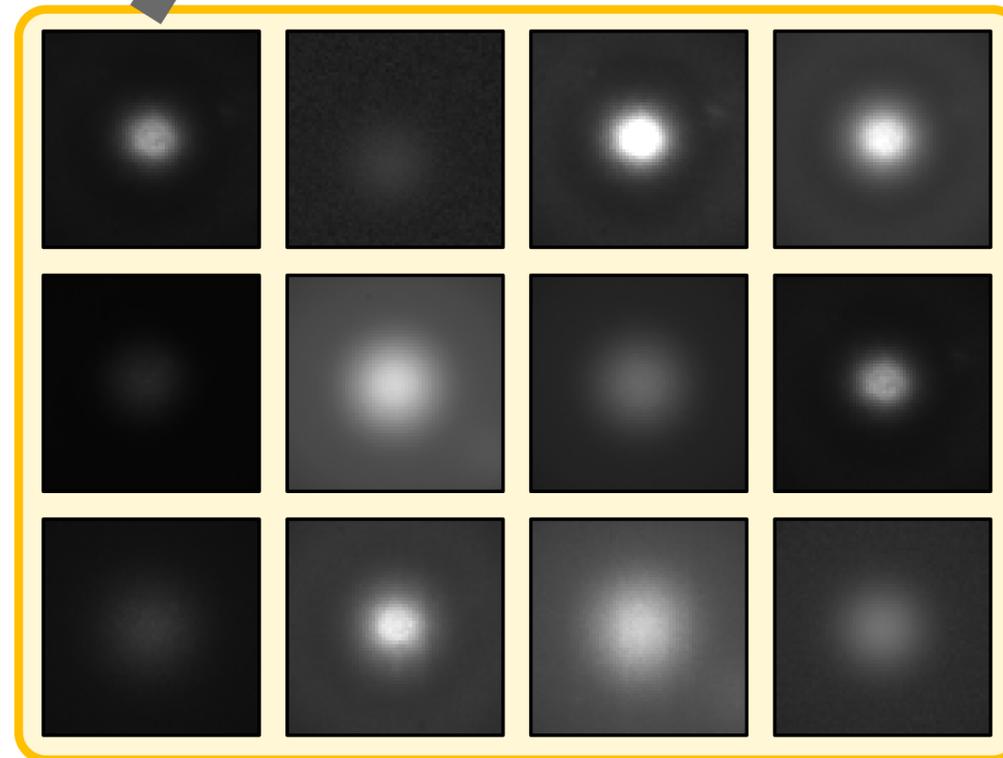
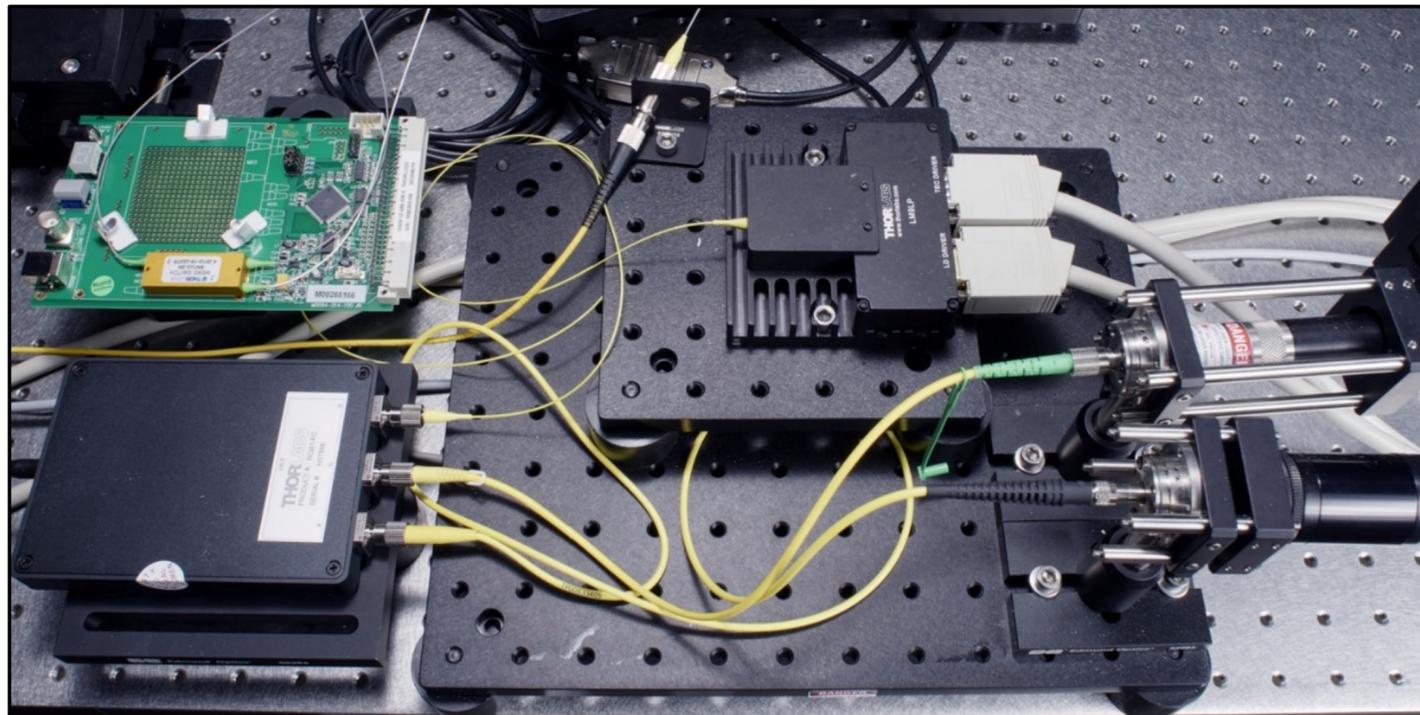
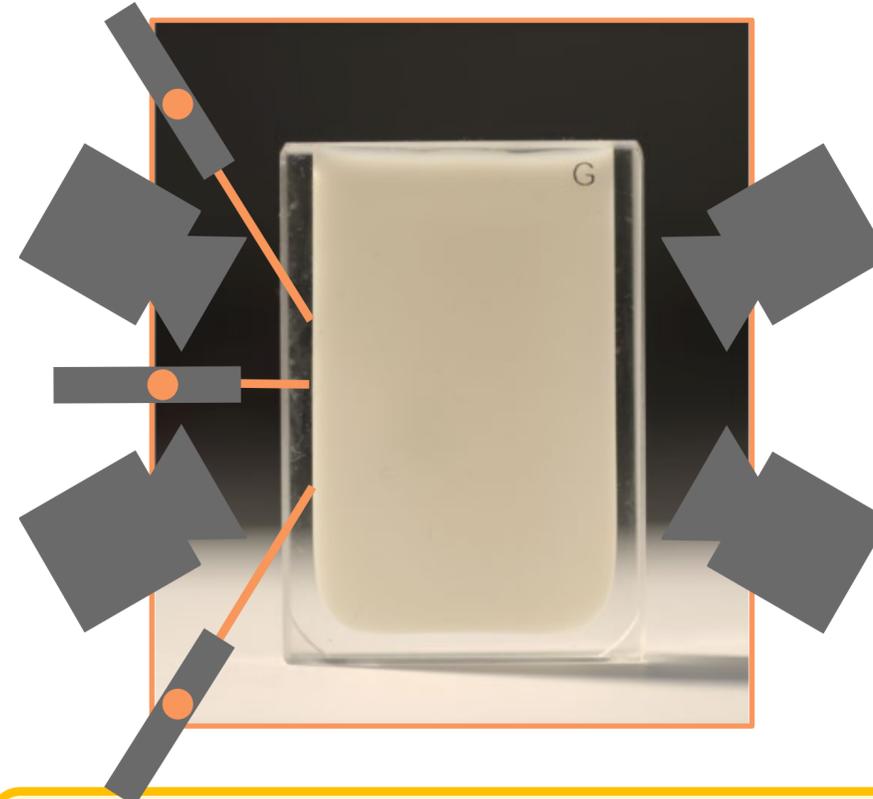
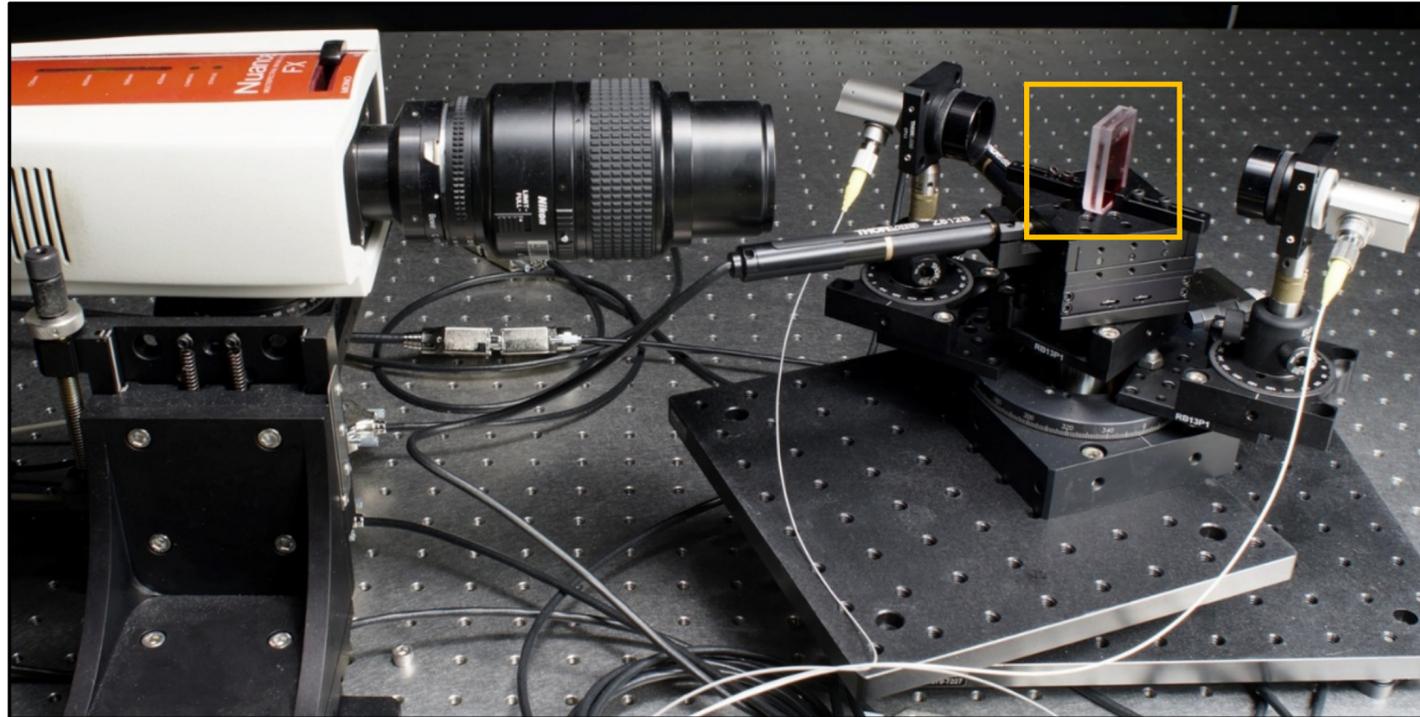
Inverse-Rendering Performance

- Each inverse-rendering optimization takes 15–100 minutes
- **Inverse**-rendering performance \neq **differentiable**-rendering performance
 - *Differentiable rendering* only accounts for **<4%** of total optimization time
 - *Geometric processing* (e.g., collision detection) takes up to **70%**
- We need much better geometric processing systems!
 - e.g., elTopo [Brochu et al. 2009] is CPU-based and single-threaded

Inverse scattering [Gkioulekas et al. 2013]

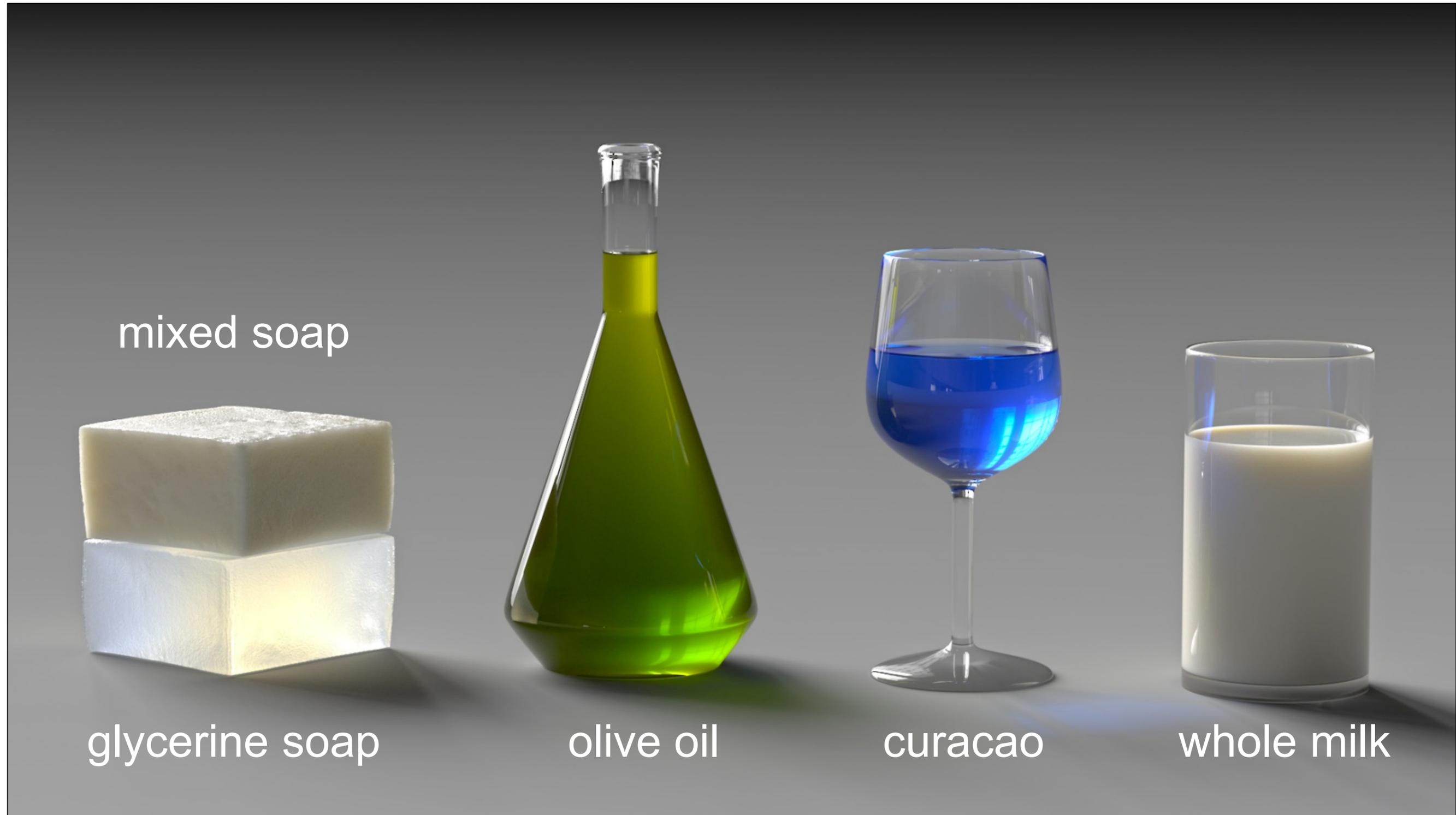


Acquisition setup



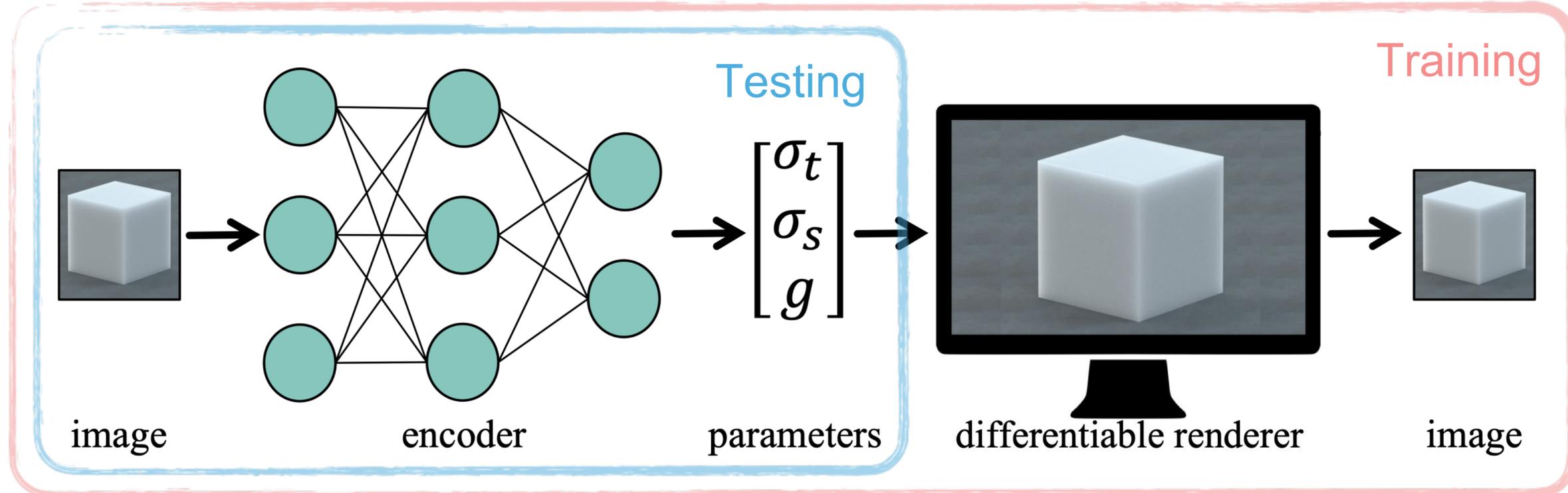
Invert using
differentiable
rendering

Synthetic renderings



Inverse transport networks [Che et al. 2020]

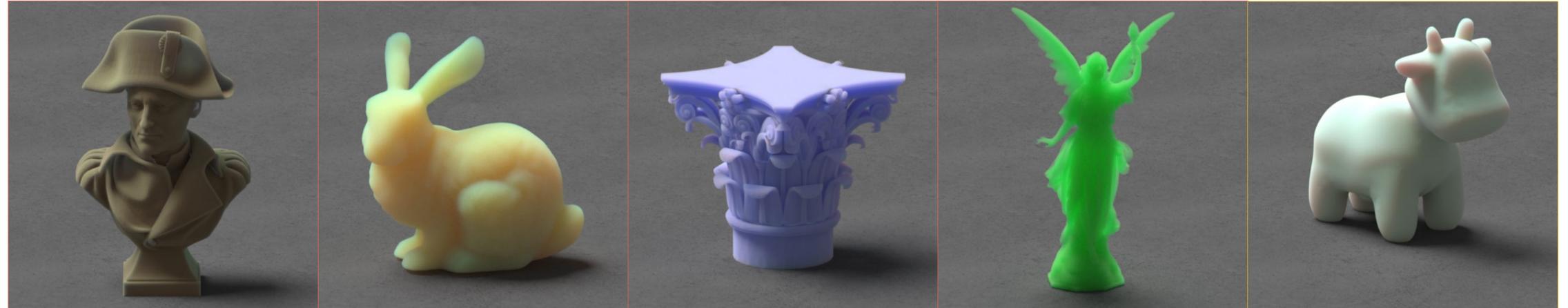
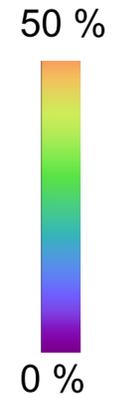
- Integrate physics-based rendering into **machine learning** pipeline
- Predict scattering parameters from images



- Utilize *image loss* provided by a volume path tracer to regularize training
- Use the trained encoder to perform inverse scattering during testing

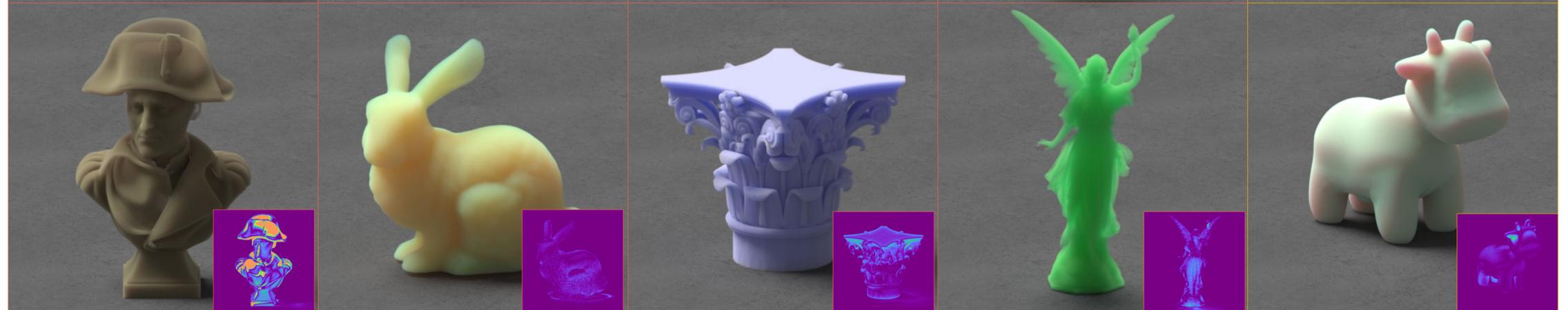
Examples

Groundtruth



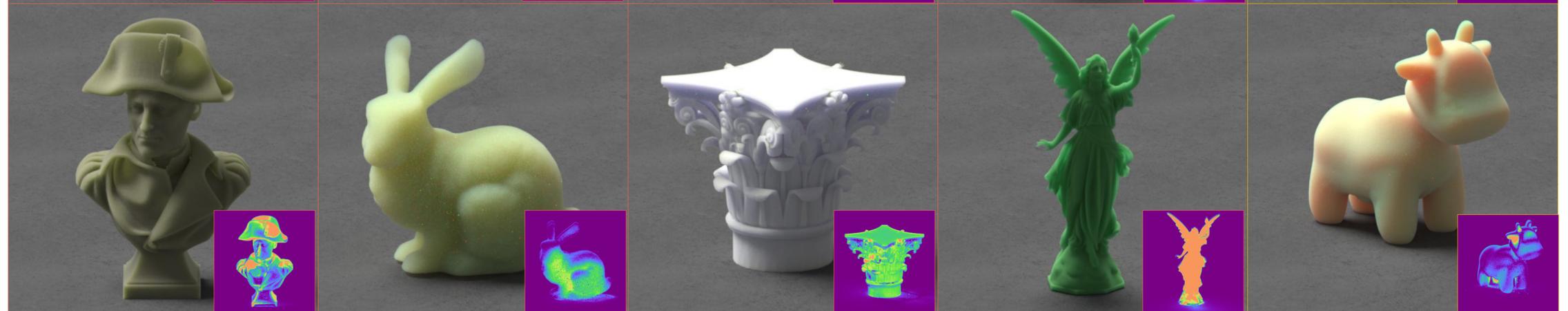
Inverse transport network

parameter loss: 0.60x
appearance loss: 0.40x
novel appearance loss: 0.42x

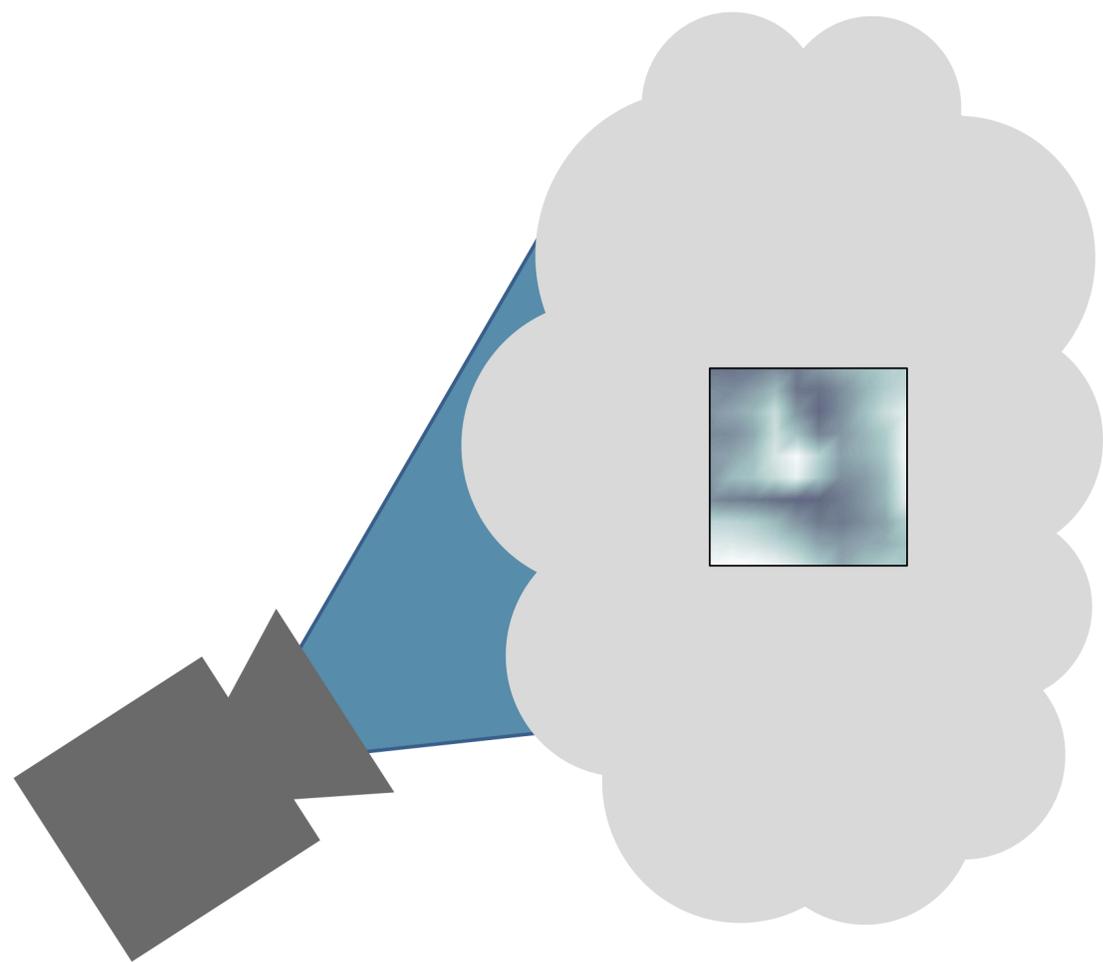


Baseline

parameter loss: 1x
appearance loss: 1x
novel appearance loss: 1x

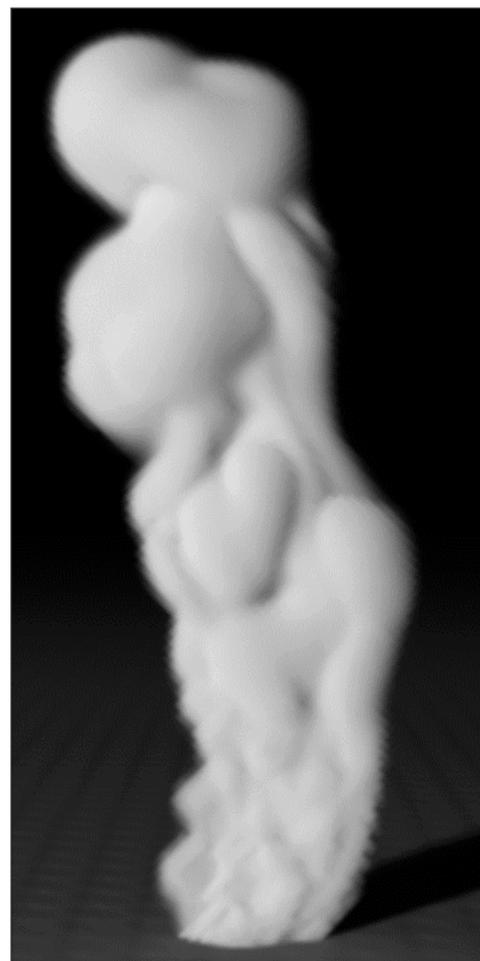


Optical tomography [Gkioulekas et al. 2015]



camera

thick smoke cloud



simulated camera
measurements



reconstructed
cloud volume



slice through
the cloud

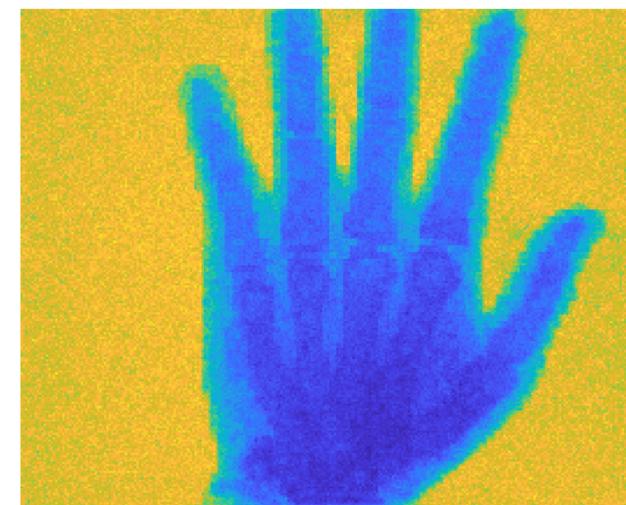
Active area of research



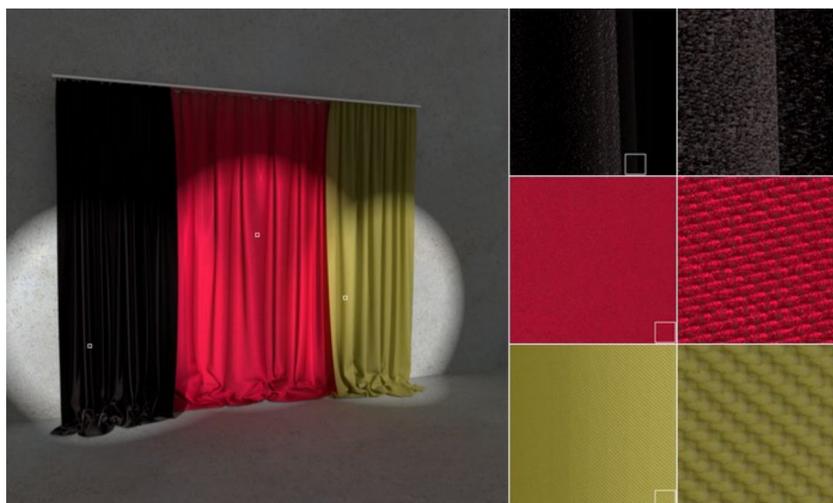
industrial dispersions
[Gkioulekas et al. 2013]



efficient algorithms
[Nimier-David et al. 2019, 2020]



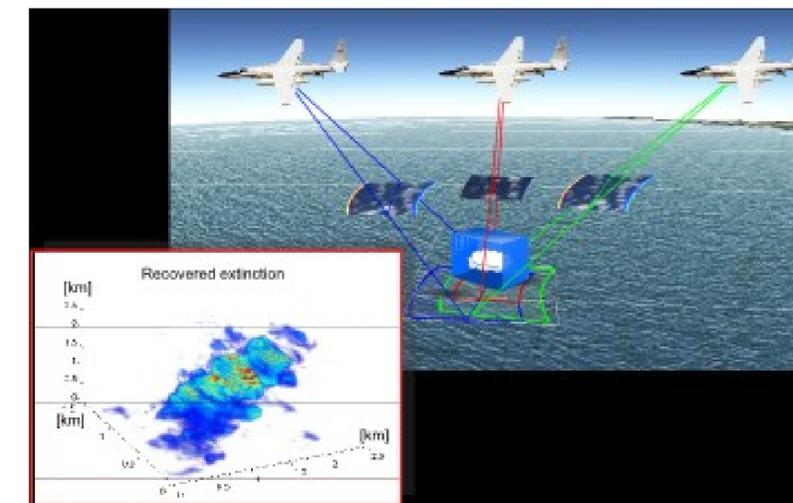
computed tomography
[Geva et al. 2018]



woven fabrics
[Khungurn et al. 2015,
Zhao et al. 2016]

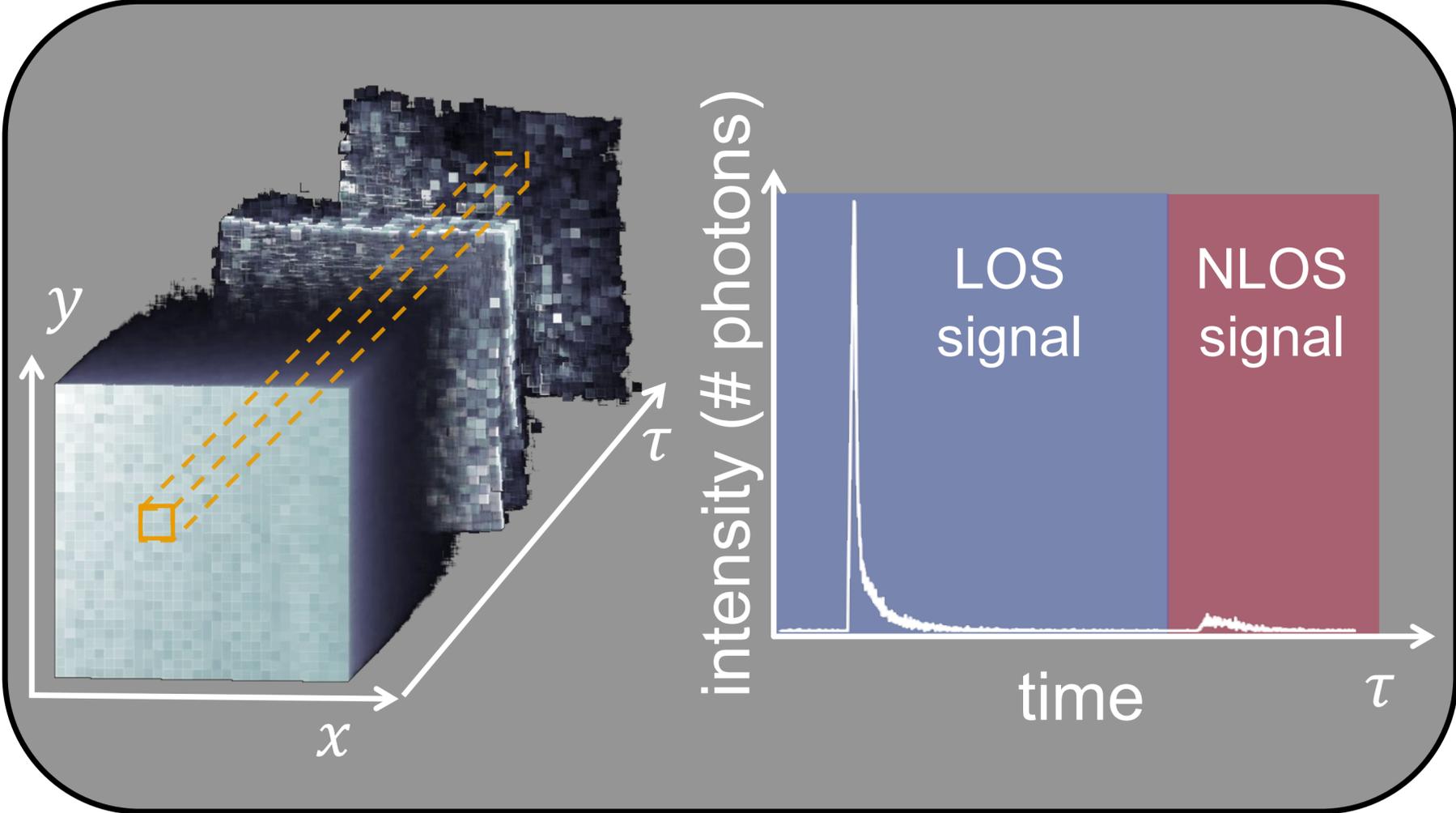
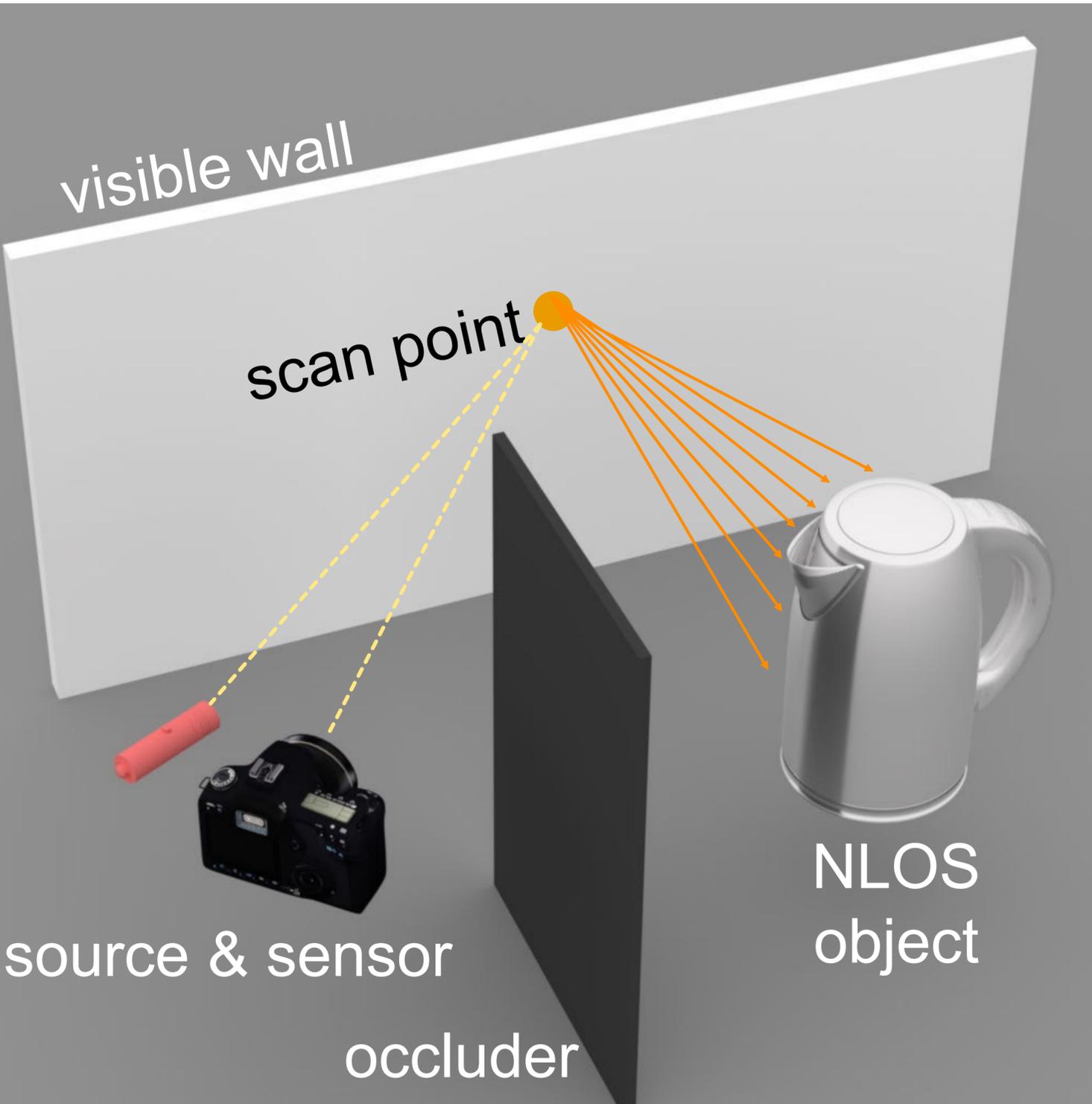


3D printing
[Elek et al. 2019,
Nindel et al. 2021]



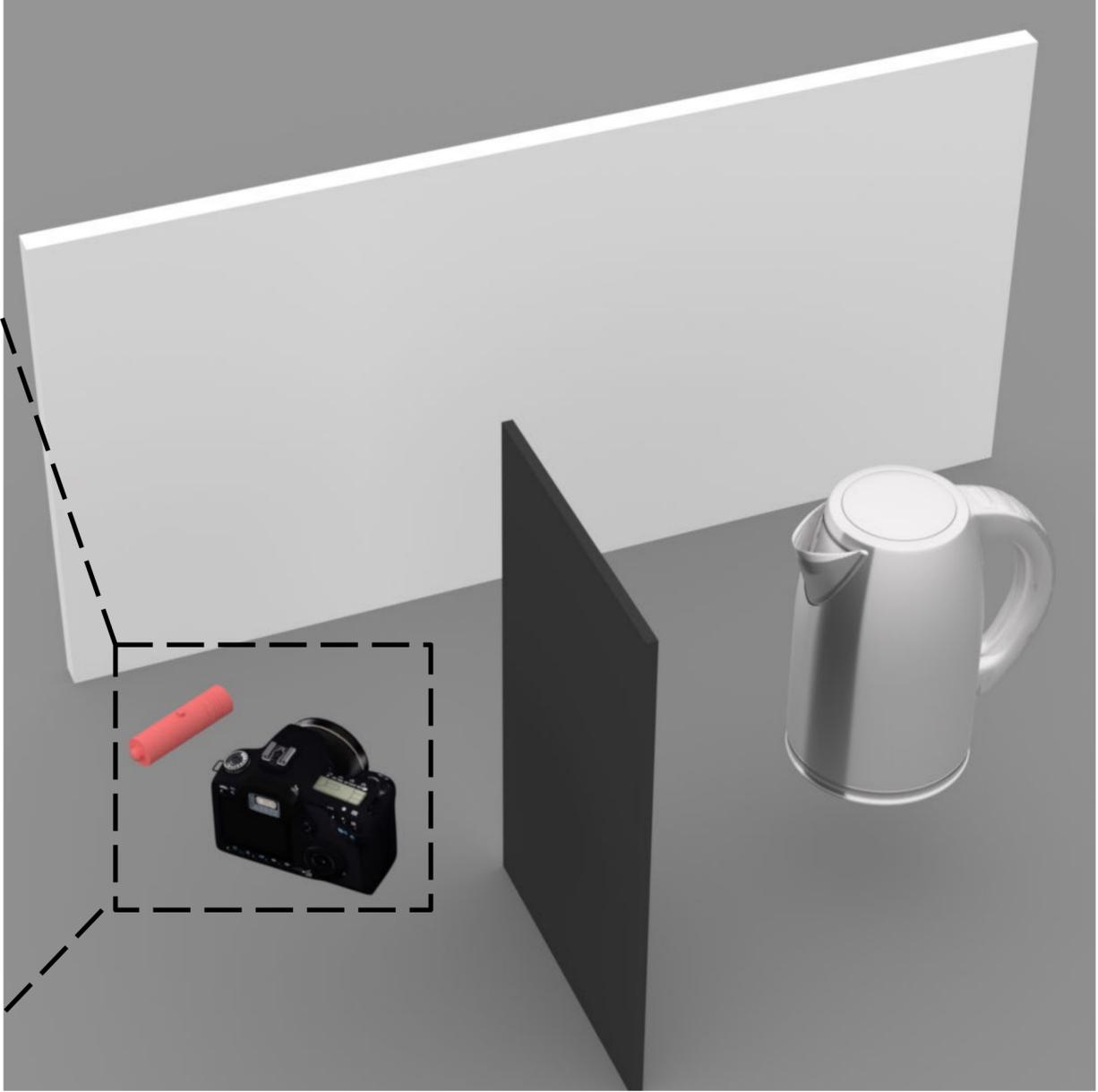
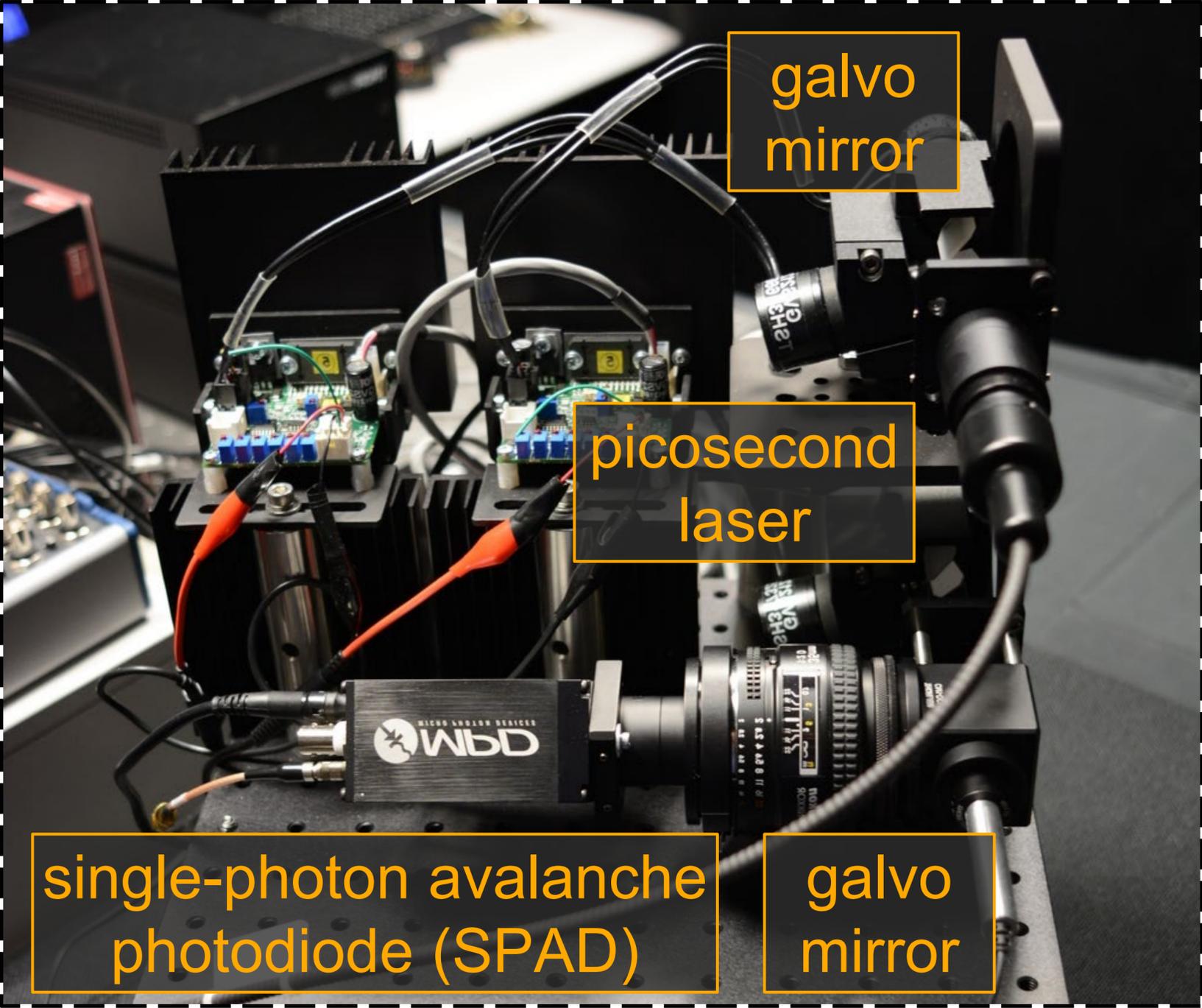
cloud tomography
[Levis et al. 2015,
2017, 2020]

Non-line-of-sight (NLOS) imaging

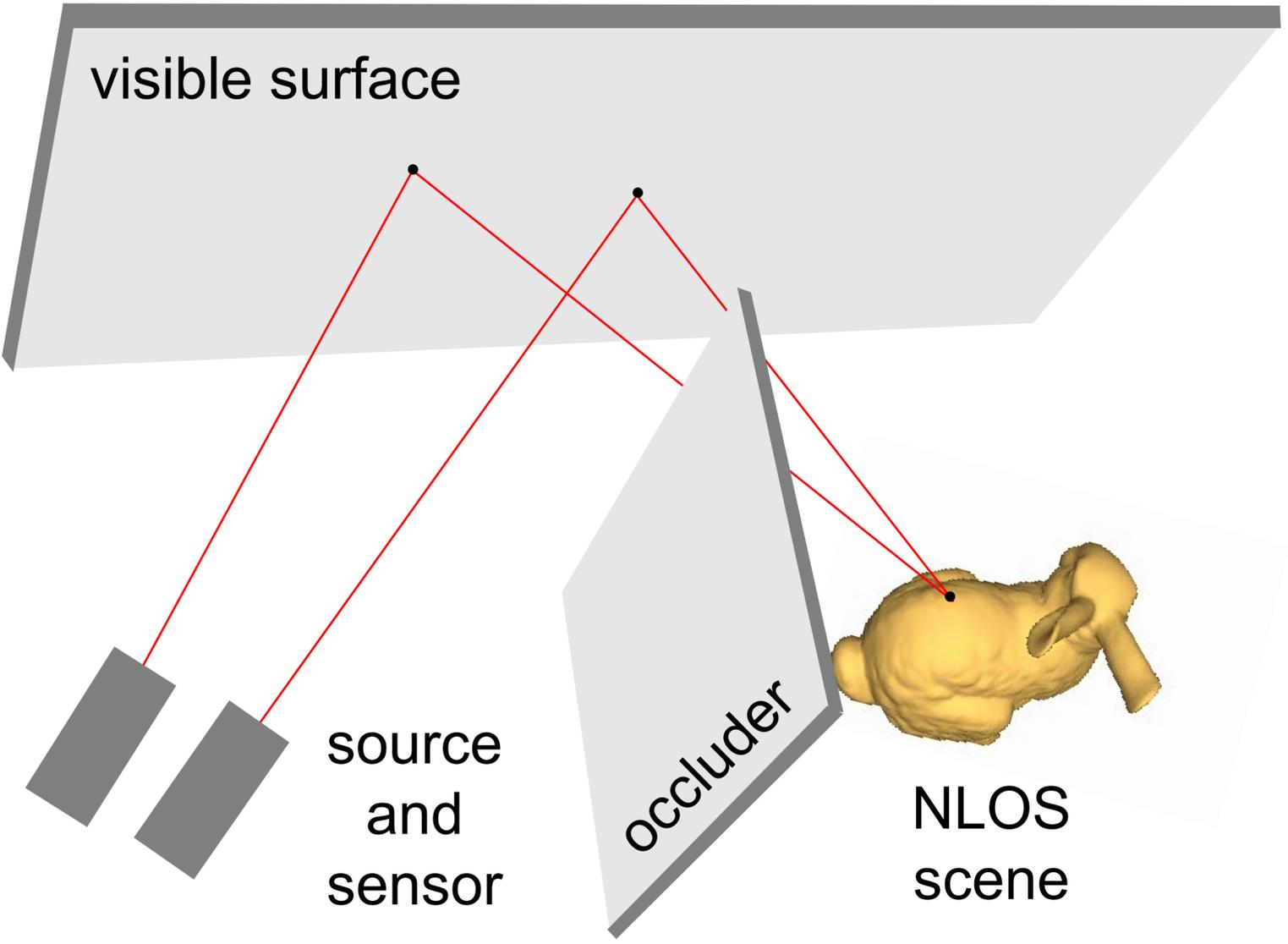


Time-of-flight measurements

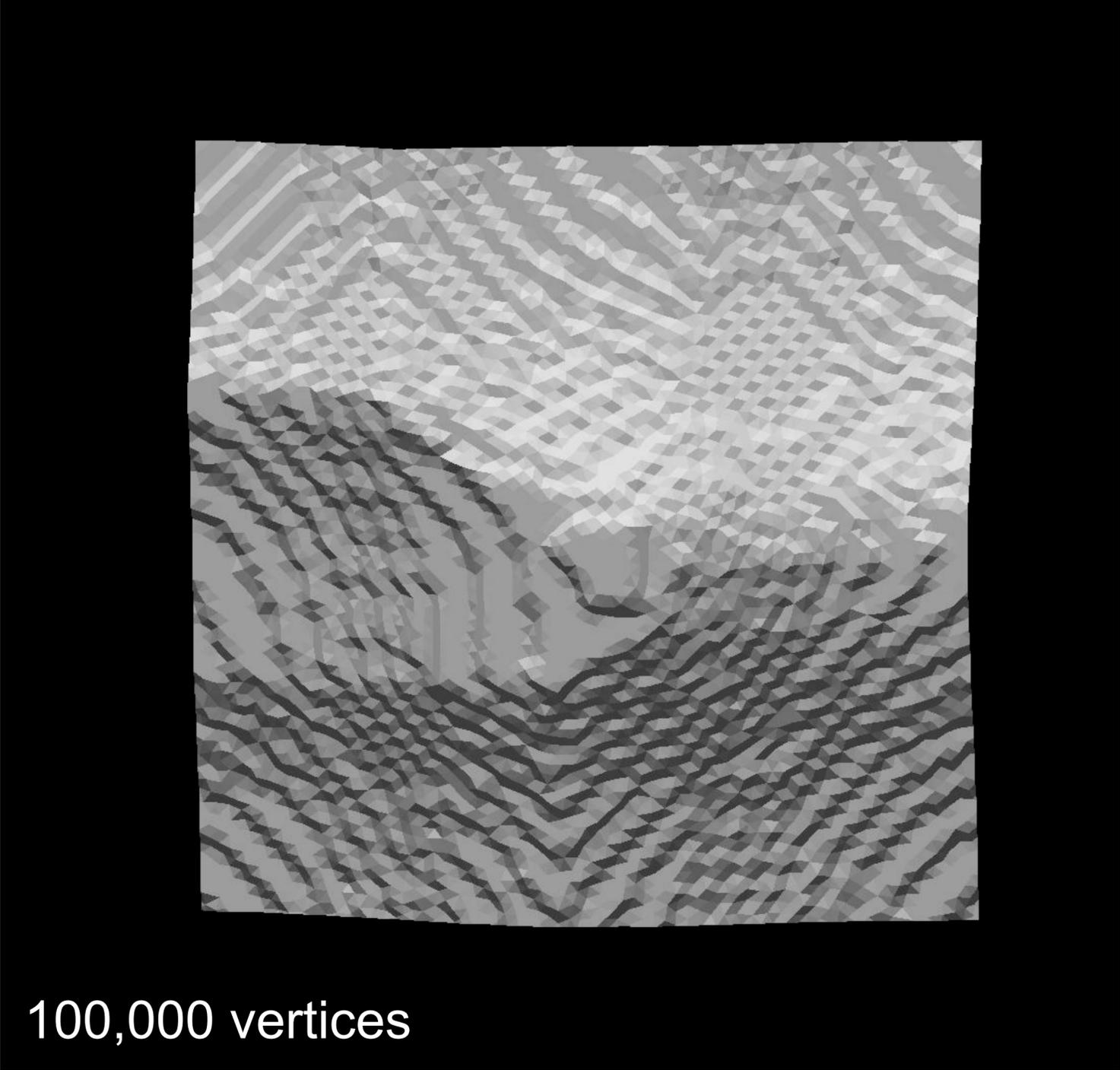
SPAD-based lidar



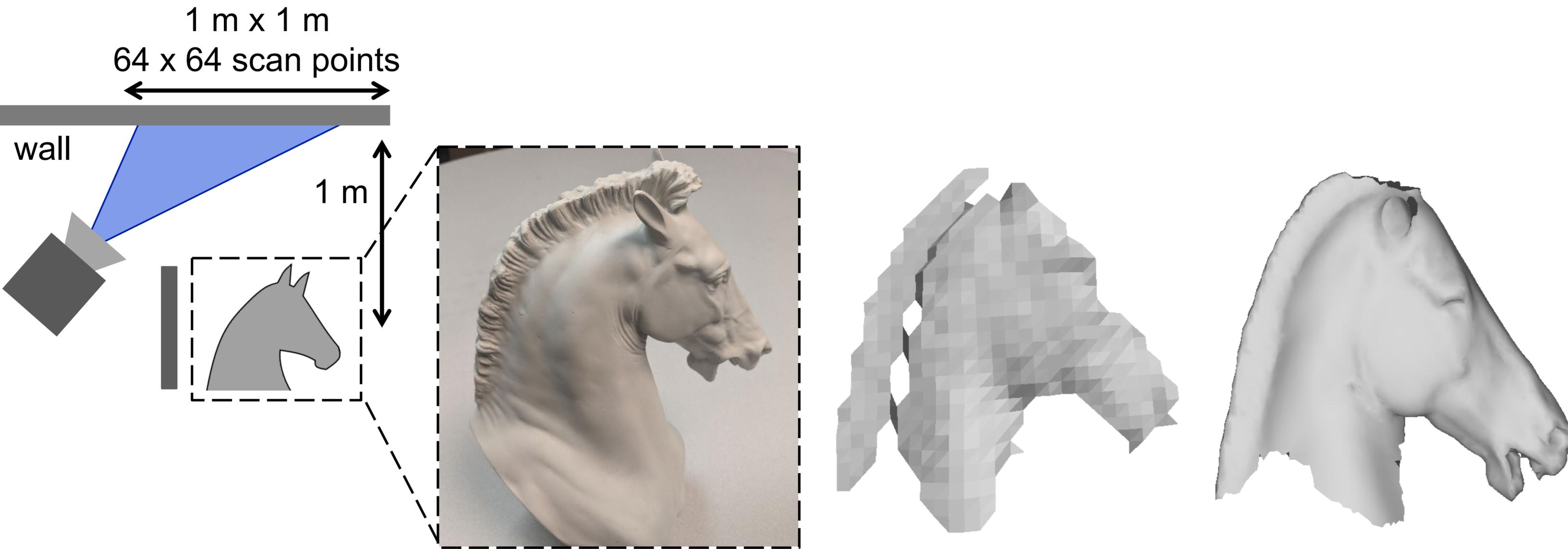
NLOS shape optimization [Tsai et al. 2019]



Simulated time-of-flight data



NLOS shape optimization [Tsai et al. 2019]



scene

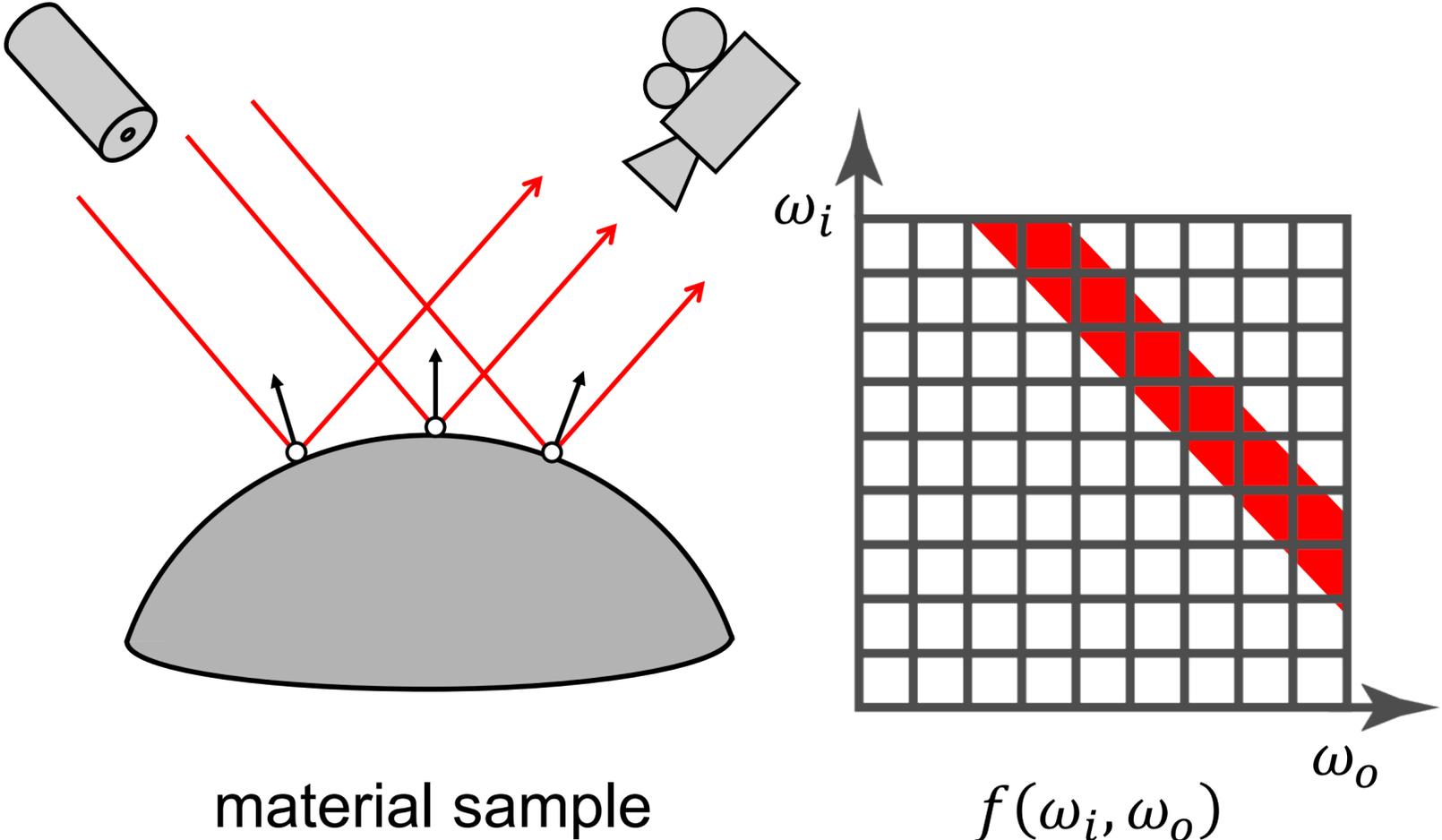
initial mesh
[O'Toole et al. 2018]

optimized mesh

Measured time-of-flight data

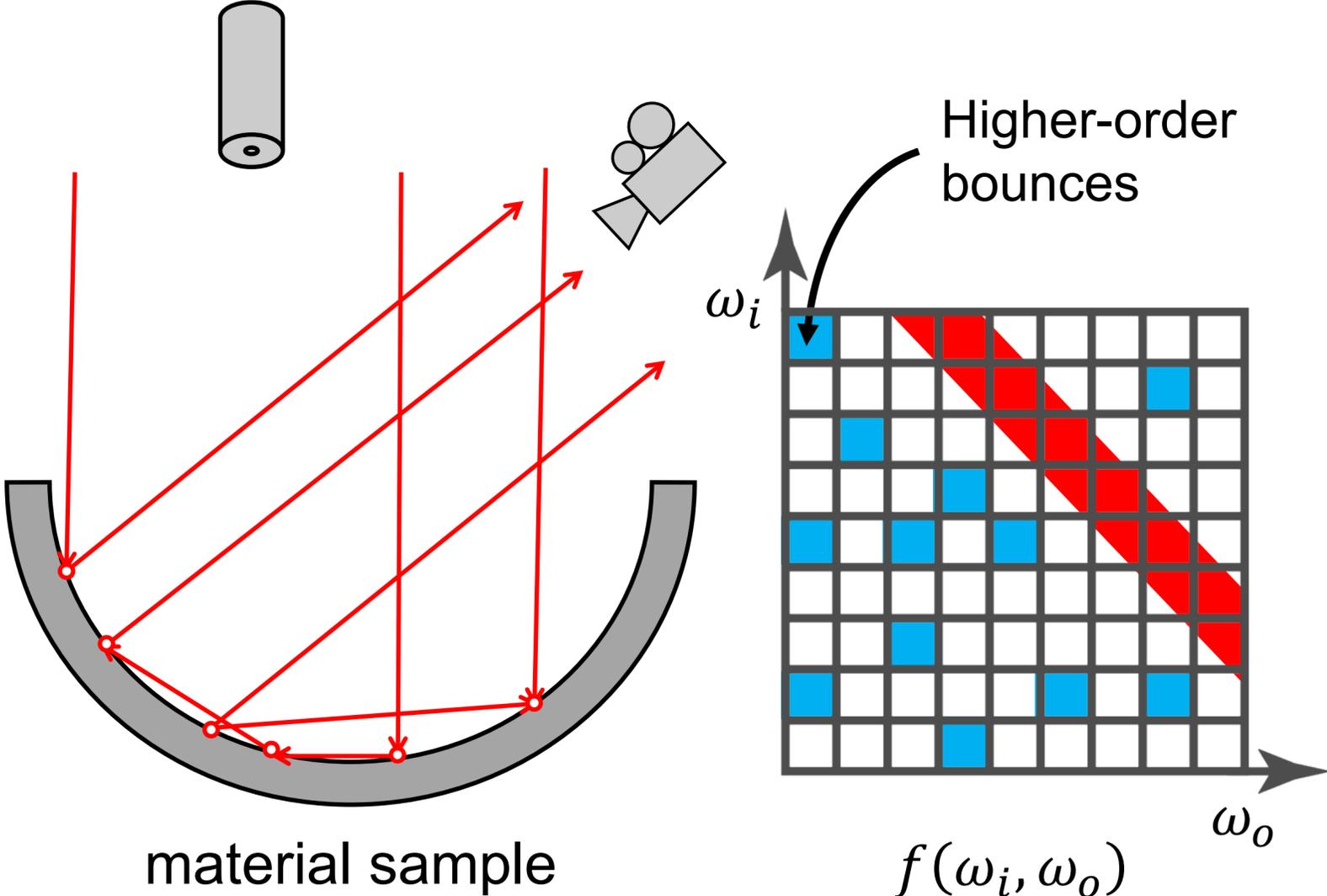
Reflectometry from interreflections [Shem-Tov et al. 2020]

Direct illumination measurements



- + Intensities map directly to BRDF entries
- Many measurements (2D scan of light & camera)

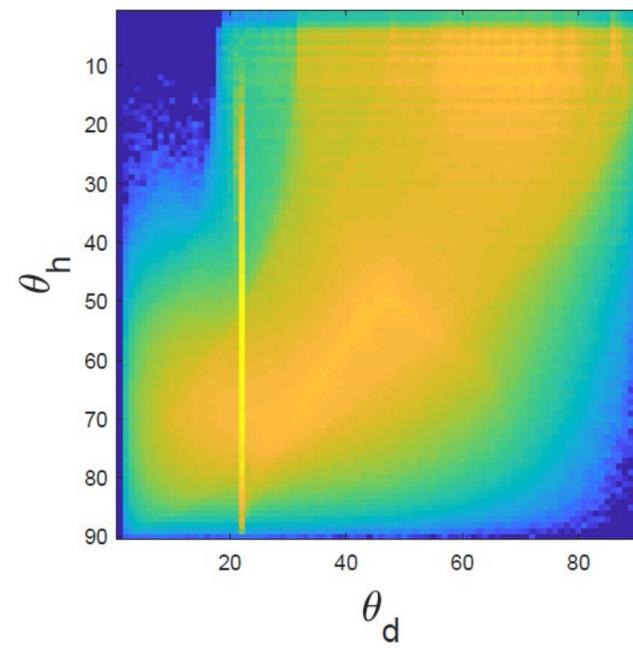
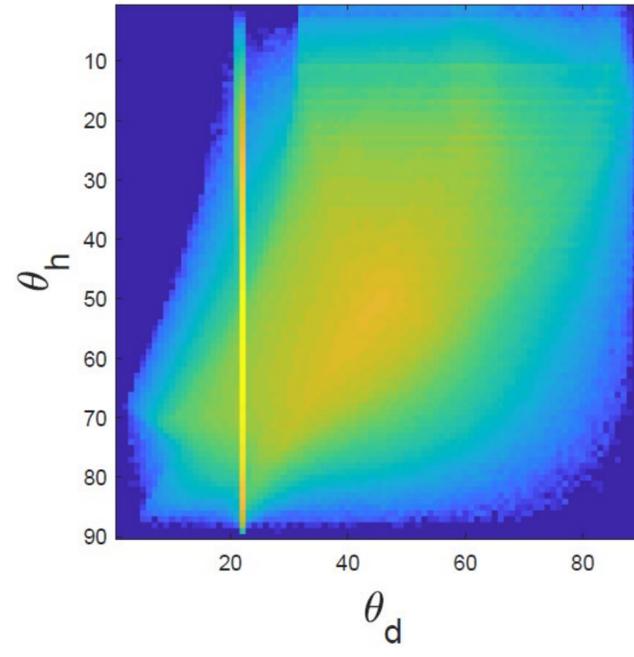
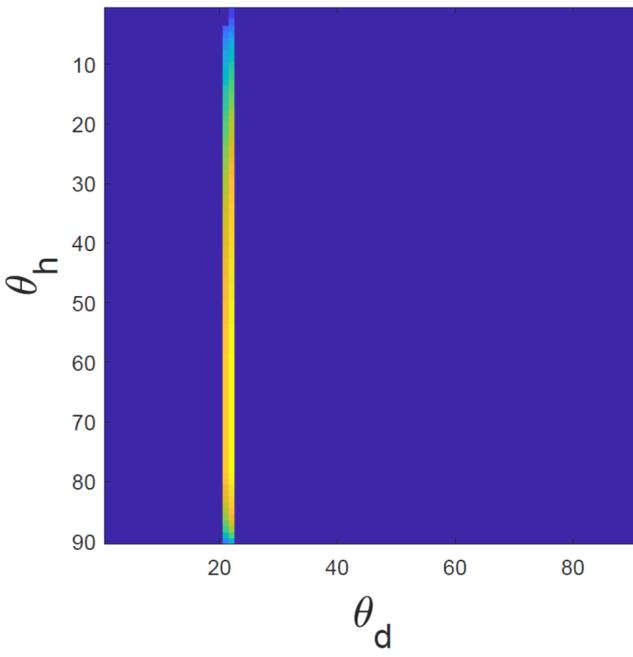
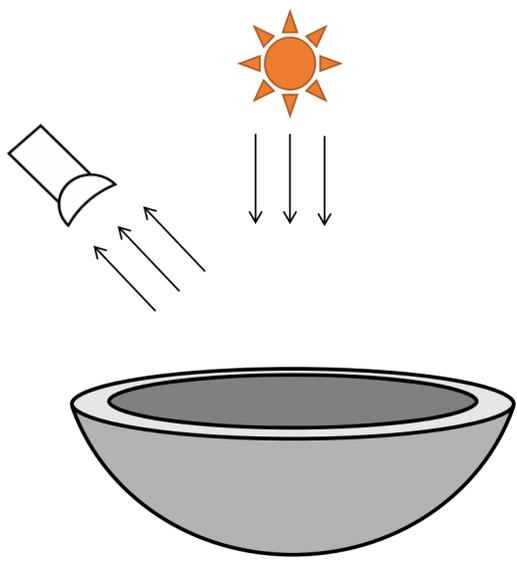
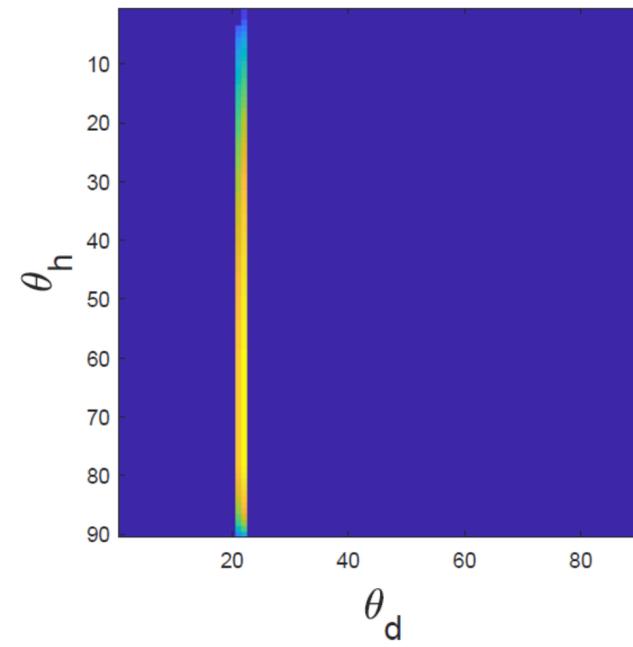
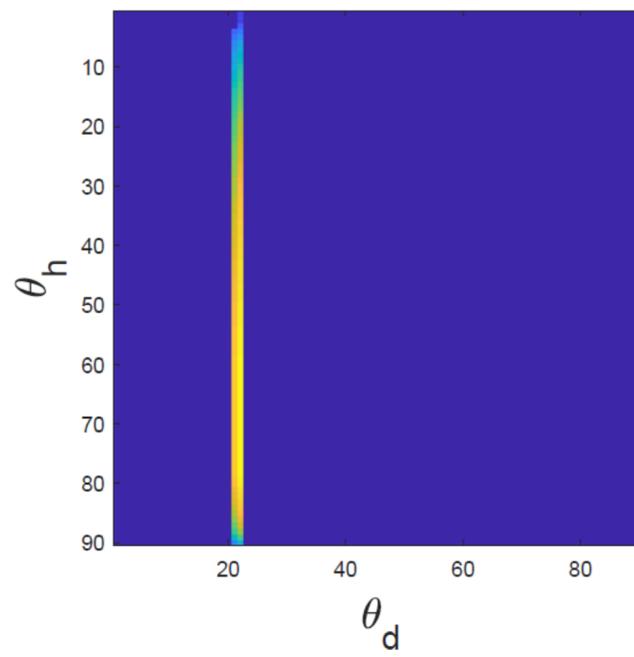
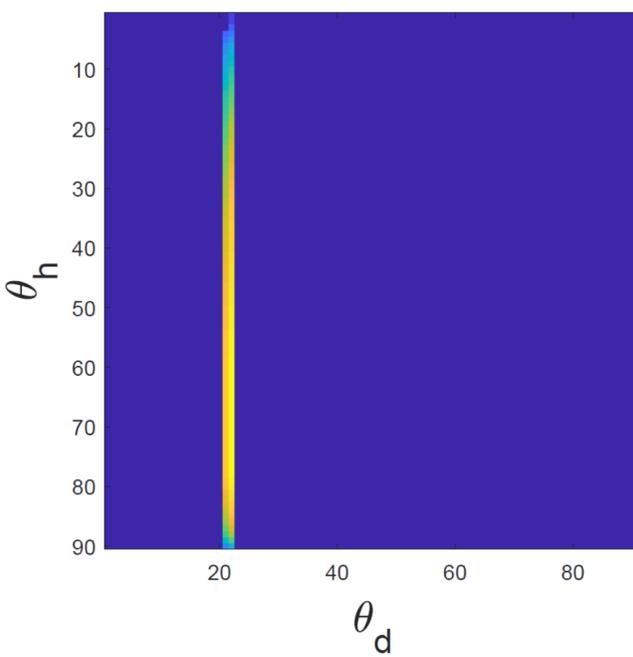
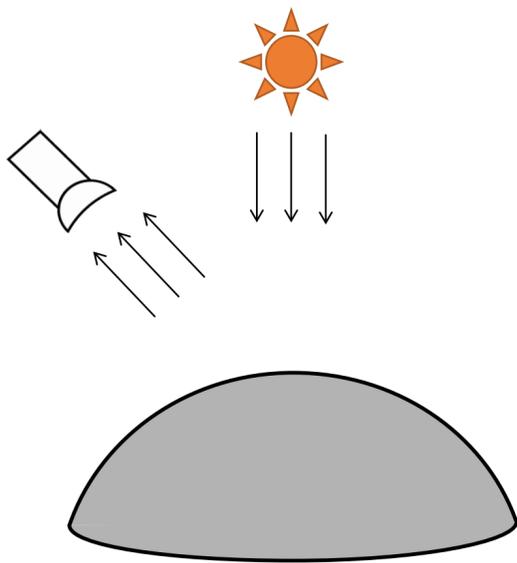
Global illumination measurements



- + Fewer measurements (single image)
- Non-linear analysis-by-synthesis optimization

Solvable using differentiable rendering

Single-image dense BRDF sampling



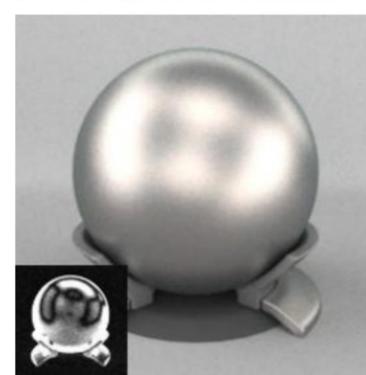
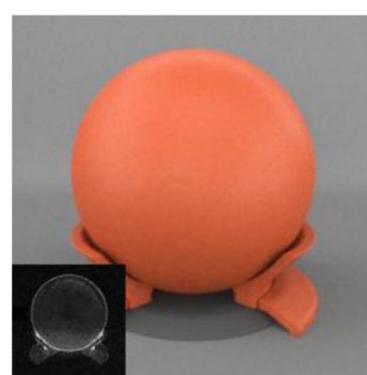
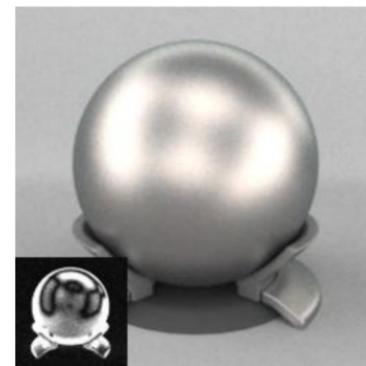
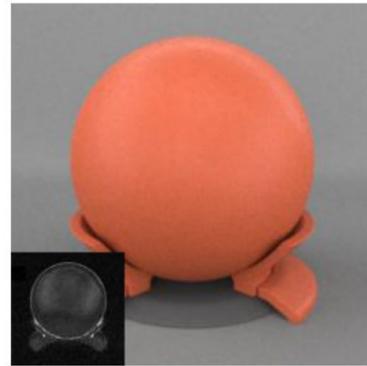
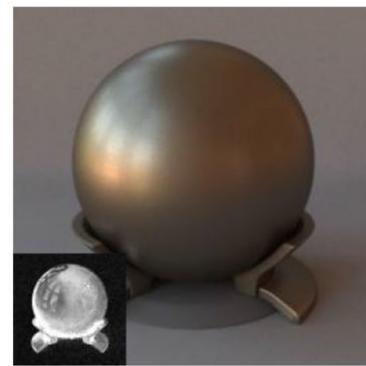
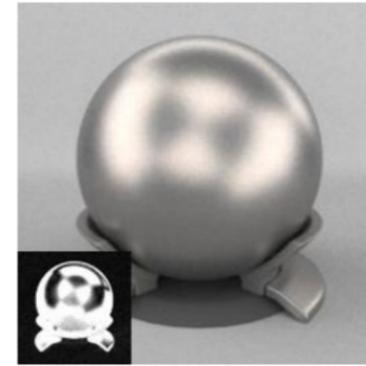
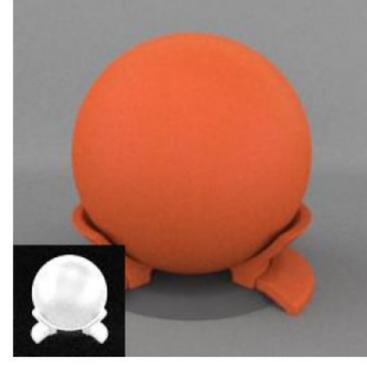
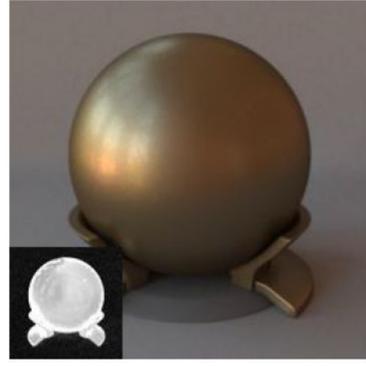
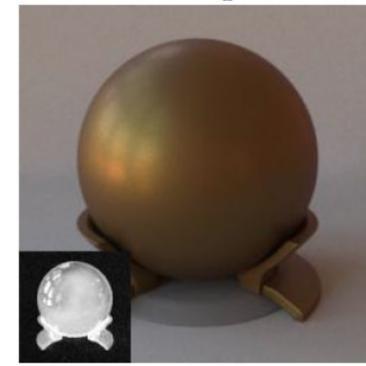
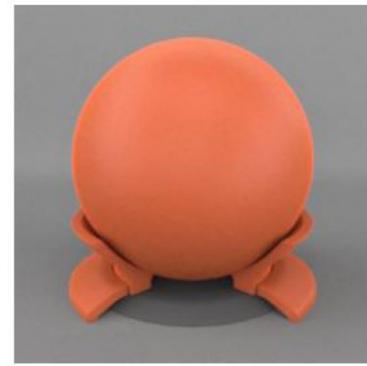
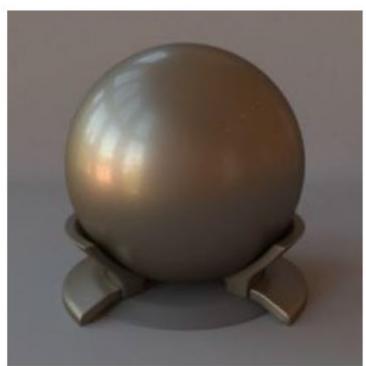
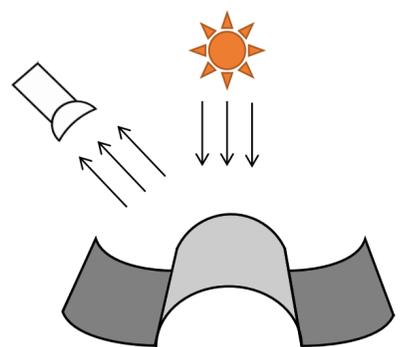
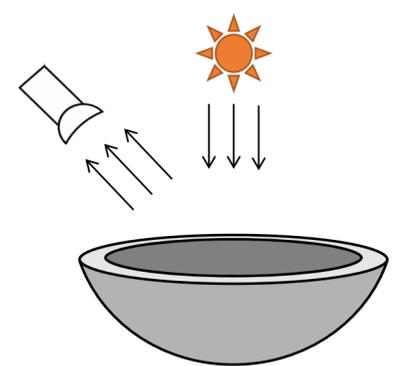
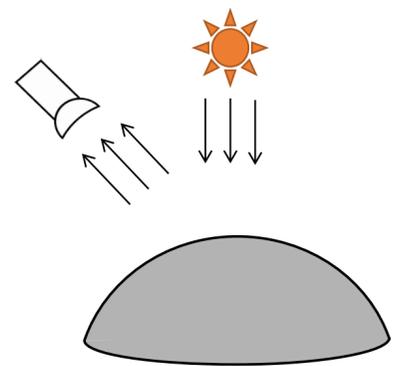
Single-bounce paths

Two-bounce paths

All-bounce paths

Results on MERL dataset

Groundtruth



Optimized shape

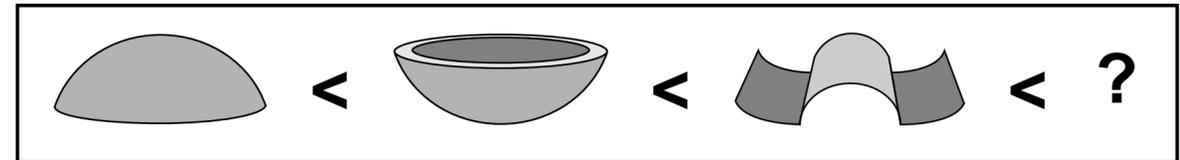
~ 6.3x better parameter recovery

~ 11.2x better parameter recovery

Global illumination can help...

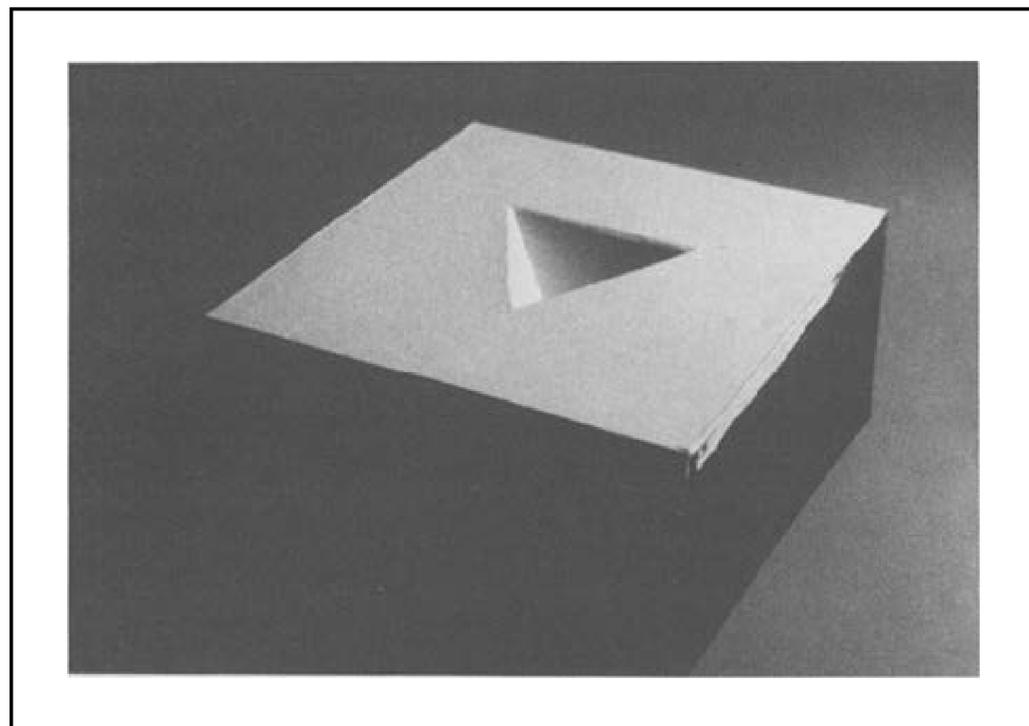
- Reduce number of measurements required for inverse rendering

- We should rethink “optimal” acquisition systems

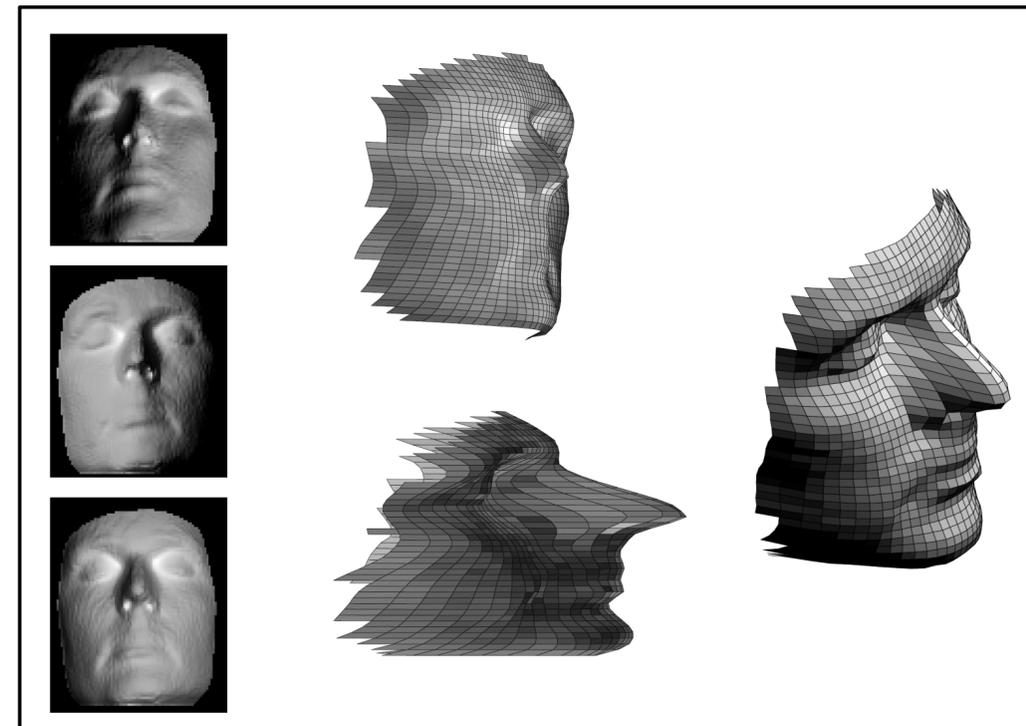


- Resolve ambiguities between different types of parameters

- We should revisit theory problems on uniqueness results



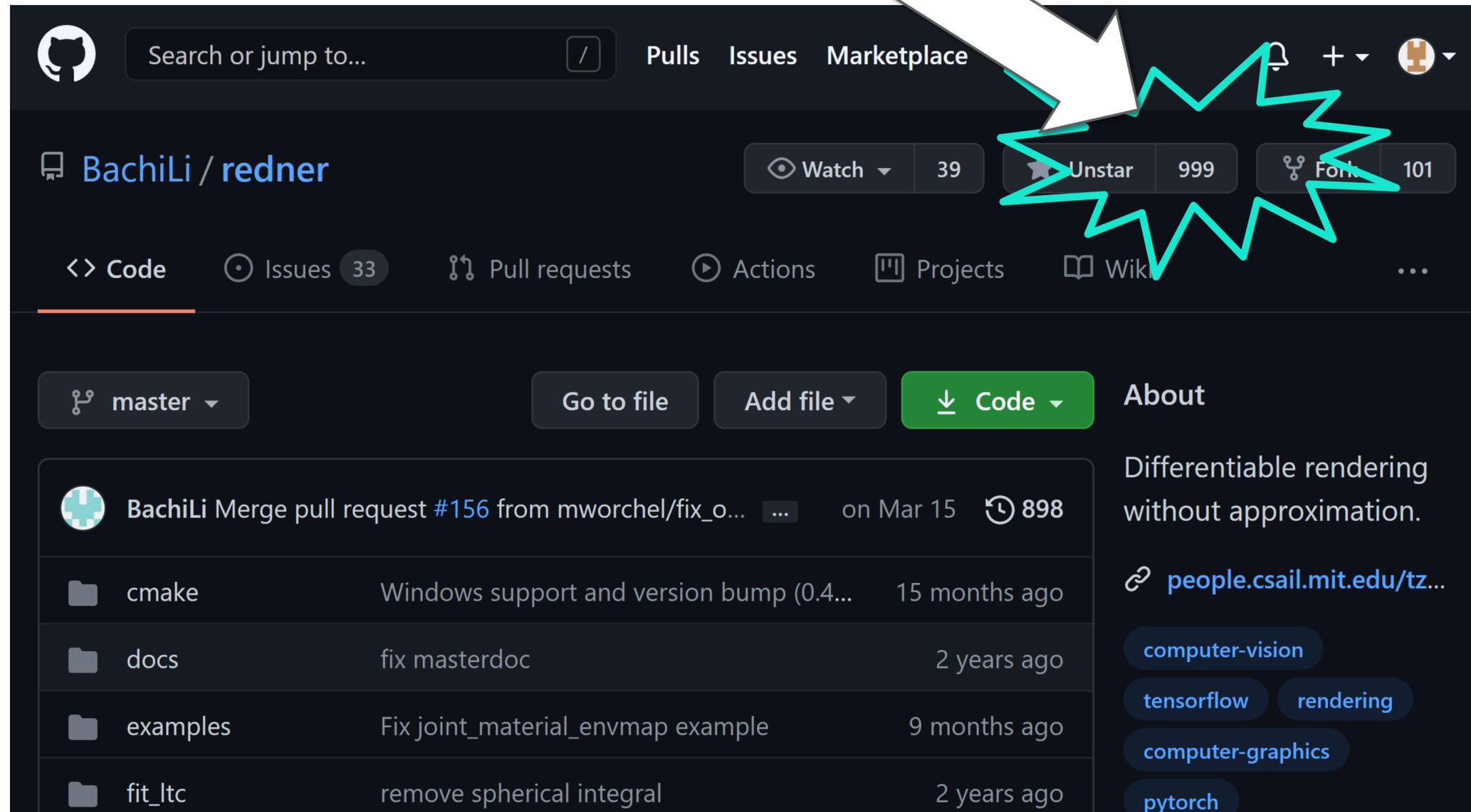
Shape from interreflections
[Nayar et al. 1990, Marr Prize]



Interreflections resolve the GBR ambiguity
[Chandraker et al. 2005]

The first physically-based differentiable renderer!

And also among the most widely used:



Search or jump to... / Pulls Issues Marketplace

BachiLi / redner Watch 39 Unstar 999 Fork 101

<> Code Issues 33 Pull requests Actions Projects Wiki

master Go to file Add file Code

BachiLi Merge pull request #156 from mworchel/fix_o... on Mar 15 898

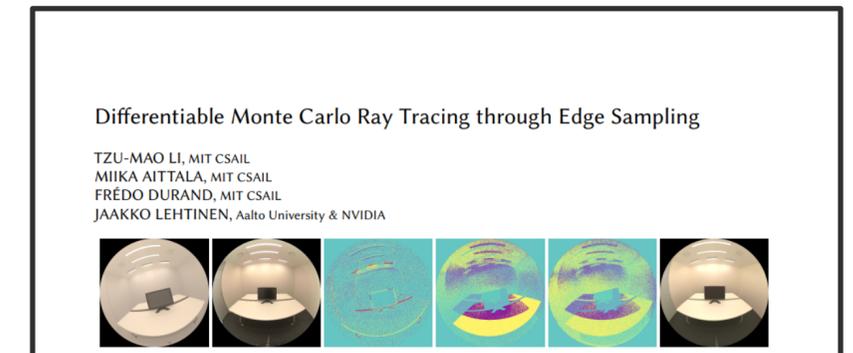
cmake	Windows support and version bump (0.4...	15 months ago
docs	fix masterdoc	2 years ago
examples	Fix joint_material_envmap example	9 months ago
fit_ltc	remove spherical integral	2 years ago

About

Differentiable rendering without approximation.

people.csail.mit.edu/tz...

computer-vision tensorflow rendering computer-graphics pytorch

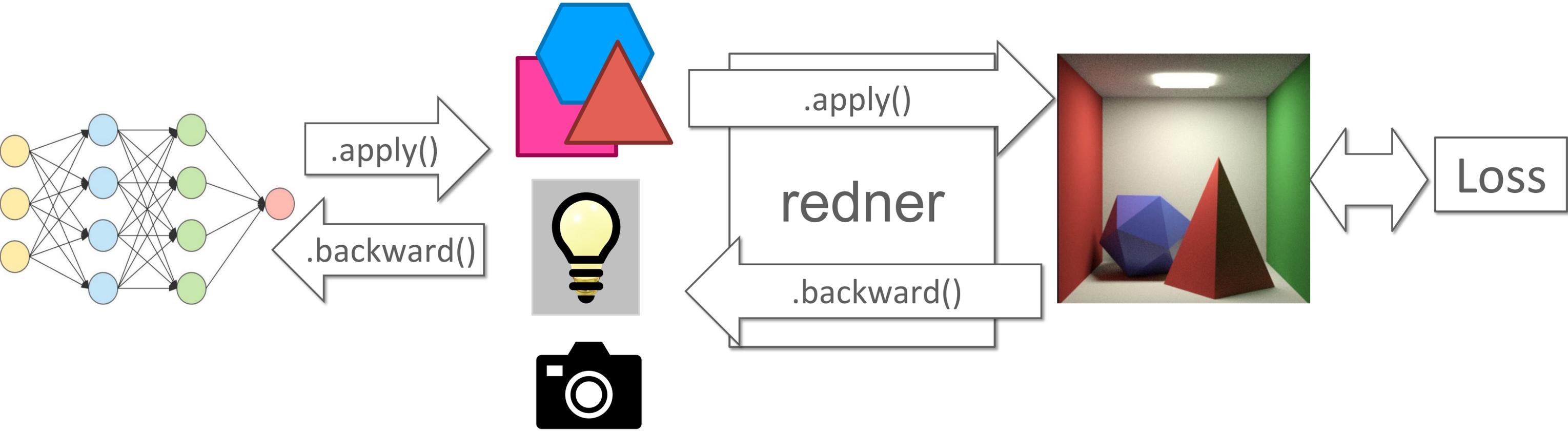


Introduced with the first unbiased differentiable rendering algorithm [Li. 2018]

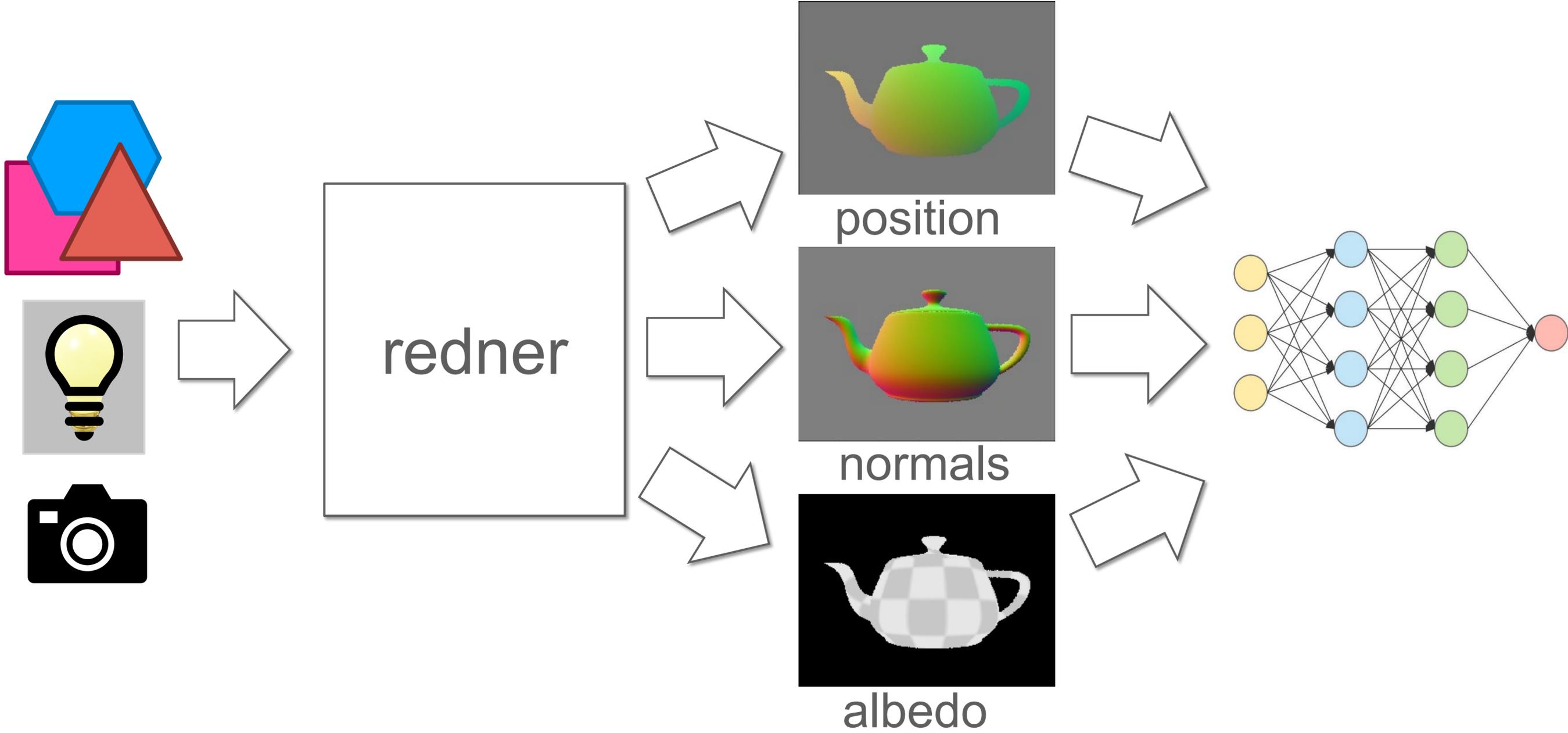
Redner is built from the ground up for ML applications

 PyTorch


TensorFlow
("Eager" mode only)



Redner supports *deferred* rendering (if realistic rendering is not the goal)



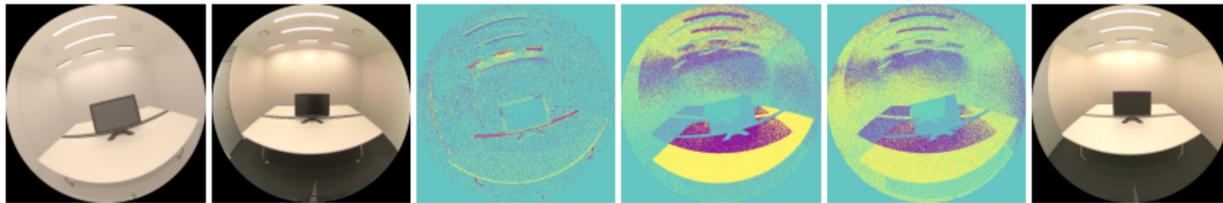
(say we can just use one bounce lighting)

Redner now contains *two* styles of differentiable rendering

```
pyredner.integrators.EdgeSamplingIntegrator()
```

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

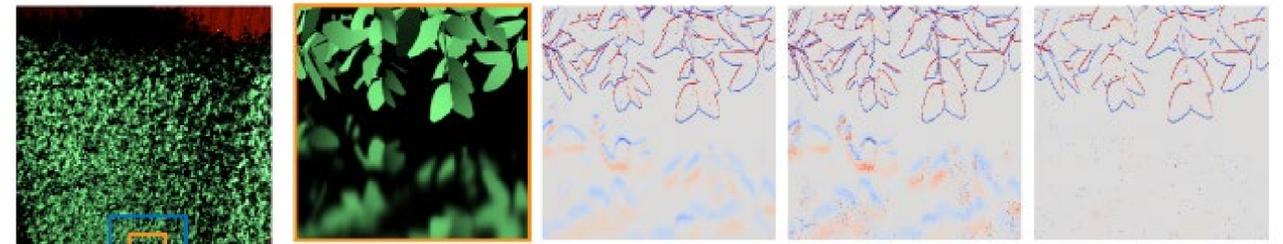


Edge-sampling [Li et al. 2018]

```
pyredner.integrators.WarpFieldIntegrator()
```

Unbiased Warped-Area Sampling for Differentiable Rendering

SAI PRAVEEN BANGARU, Massachusetts Institute of Technology
TZU-MAO LI, Massachusetts Institute of Technology
FRÉDO DURAND, Massachusetts Institute of Technology



Warped-area sampling [Bangaru et al. 2020]

Swap freely between two methods

```
pyredner.integrators.EdgeSamplingIntegrator()
```

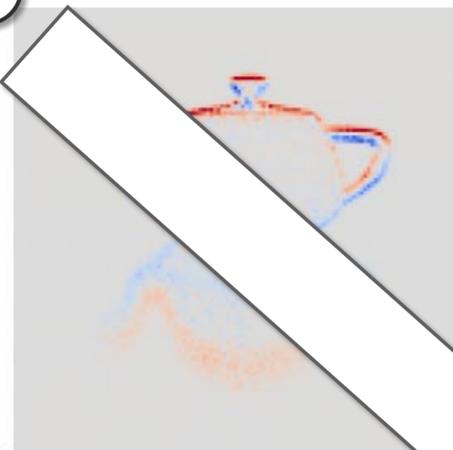
OR

```
pyredner.integrators.WarpFieldIntegrator()
```

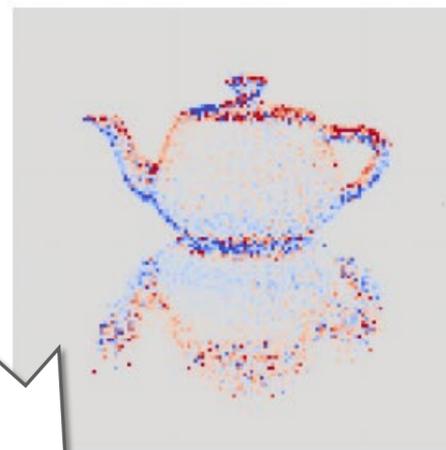
Use edge-sampling for primary visibility



Ground truth



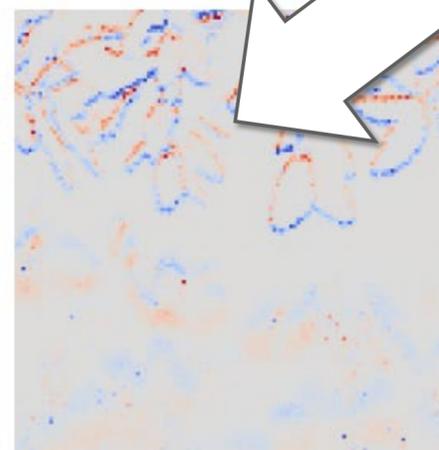
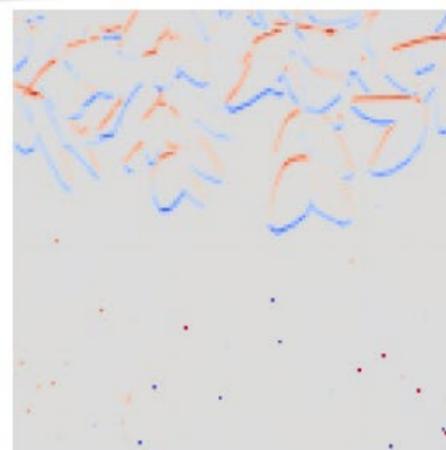
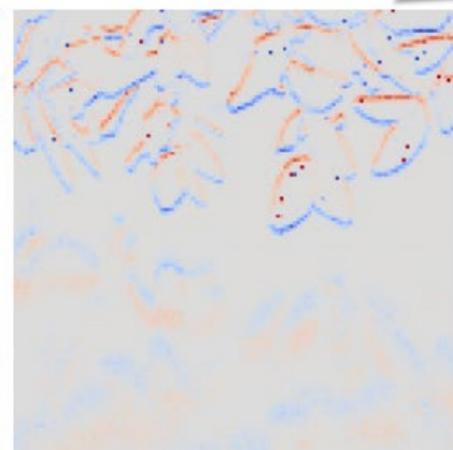
Edge-sampling



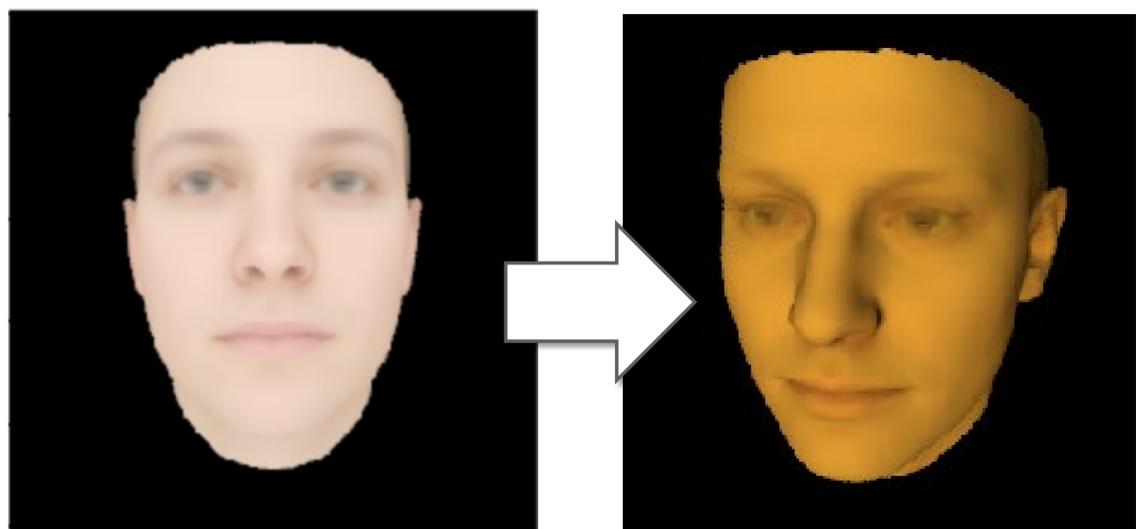
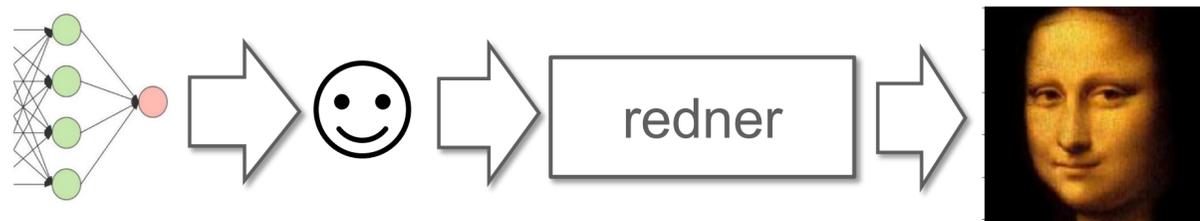
Warped-area



Use WAS for higher-order effects



Use redner anywhere in your pipeline



Optimize/Train morphable models

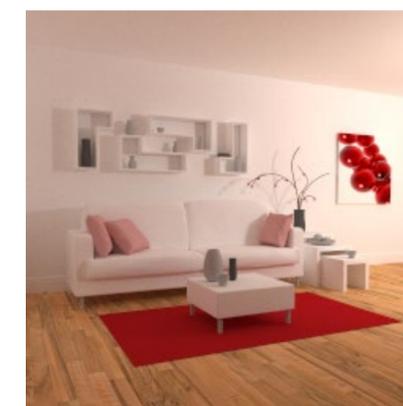
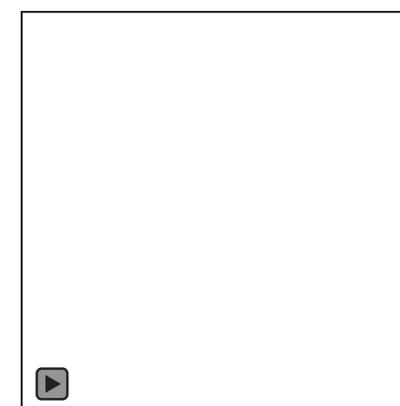


53% street sign
14.5% traffic light
6.7% handrail



23.3% handrail
3.39% street sign or traffic light

Find adversarial examples



Optimize for fine-grained pose

Use Redner today!



Github codebase

Start using on your system:



```
$pip install redner
```

```
$pip install redner-gpu
```

Or on Google Colab immediately:



Sample notebook

Summary

Take-Home Messages

- Great progress has been made in physics-based **differentiable** rendering
 - Now capable of handling global illumination, arbitrary types of camera (e.g., transient), and global scene parameters (e.g., object geometry) with decent efficiency
 - Can be applied to solve many general inverse problems
- **Ray tracing** is no longer slow
 - Many efficient systems are being actively developed (e.g., Redner, PSDR-CUDA, Mitsuba 2, Teg)
 - And differentiable rendering is usually not the performance bottleneck
- **Gradient accuracy** matters!
 - Approximated gradients can yield reduced result quality
- **Discontinuities** always exist (due to visibility) and need to be properly handled
 - Auto-diffing a path tracer may not always work

Thank you!

Funding agencies



Tutorial website

<https://diff-render.org>

