Physics-based Differentiable Rendering

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

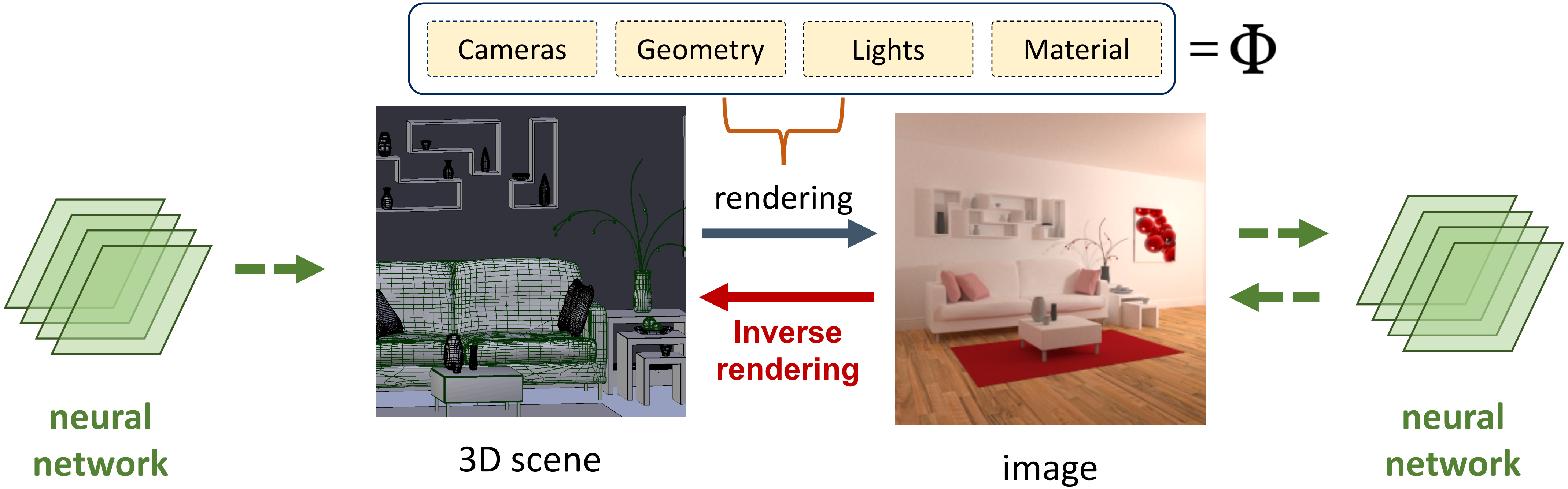
Date: 10 Mar. 2021

Jingkang Wang

UNIVERSITY OF TORONTO

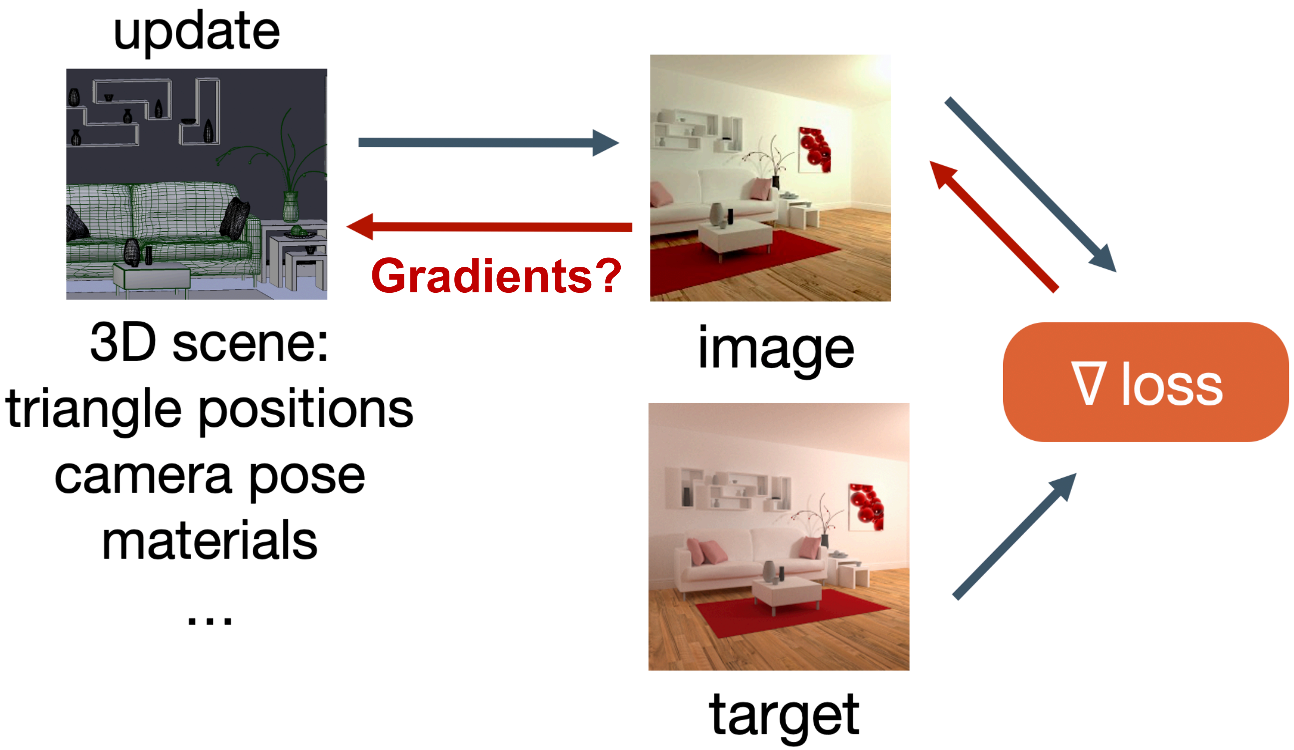
Differentiable Rendering is Important!

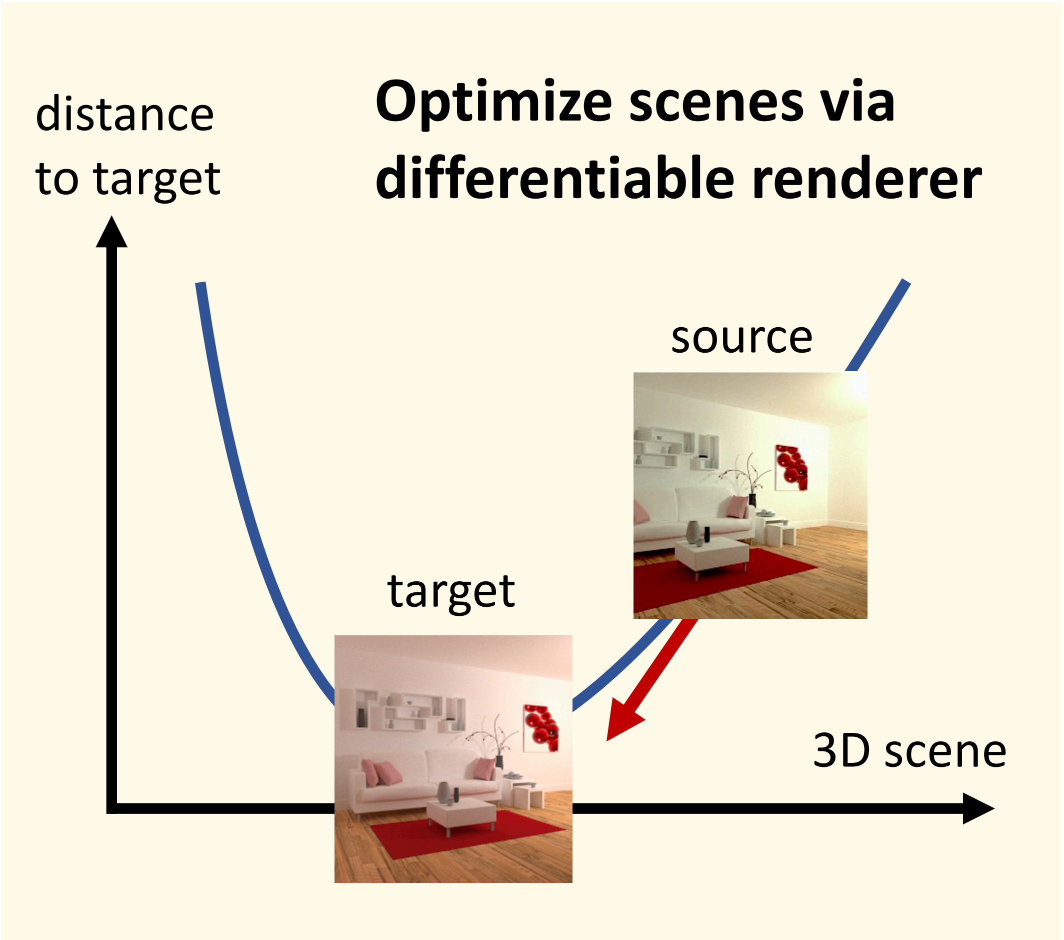
- The ability of calculating gradients are crucial to optimization
  - (a) inverse problems, (b) deep learning



Differentiable Rendering is Important!

- Render and compare approach





Differentiable Rendering is Challenging!

- Computing the gradient of rendering is **challenging**

rendered image

Rendering integral includes visibility terms that are not differentiable

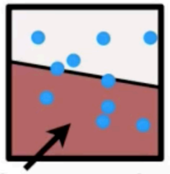
$$I = \iint \underbrace{k(x,y)}_{\text{Pixel filter}} \underbrace{L(x,y)}_{\text{Radiance (another integral)}} dxdy$$

Scene function: $f(x,y;\Phi) = k(x,y)L(x,y)$

$$\nabla I = \nabla \iint f(x,y;\Phi)dxdy$$

Differentiable Rendering is Challenging!

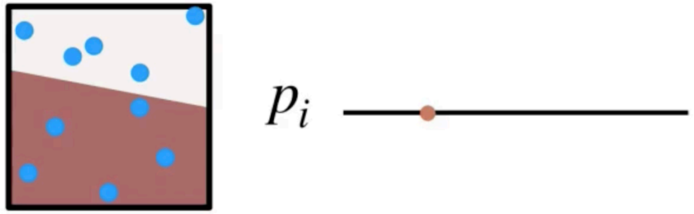
$I = \iint$ 

$E = \frac{1}{N}\sum$ 

Monte Carlo samples

Can we just use $\frac{\partial E}{\partial p_i}$ **to estimate** $\frac{\partial I}{\partial p_i}$ **?**

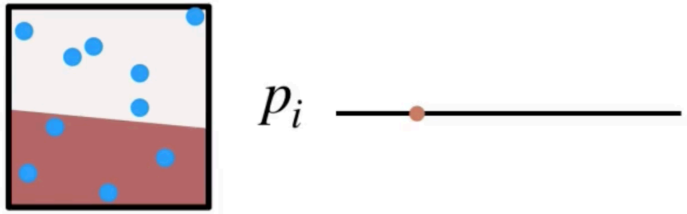
Differentiable integrand: Yes



$$\left(\frac{\partial}{\partial p_i}\int = \int \frac{\partial}{\partial p_i}\right)$$

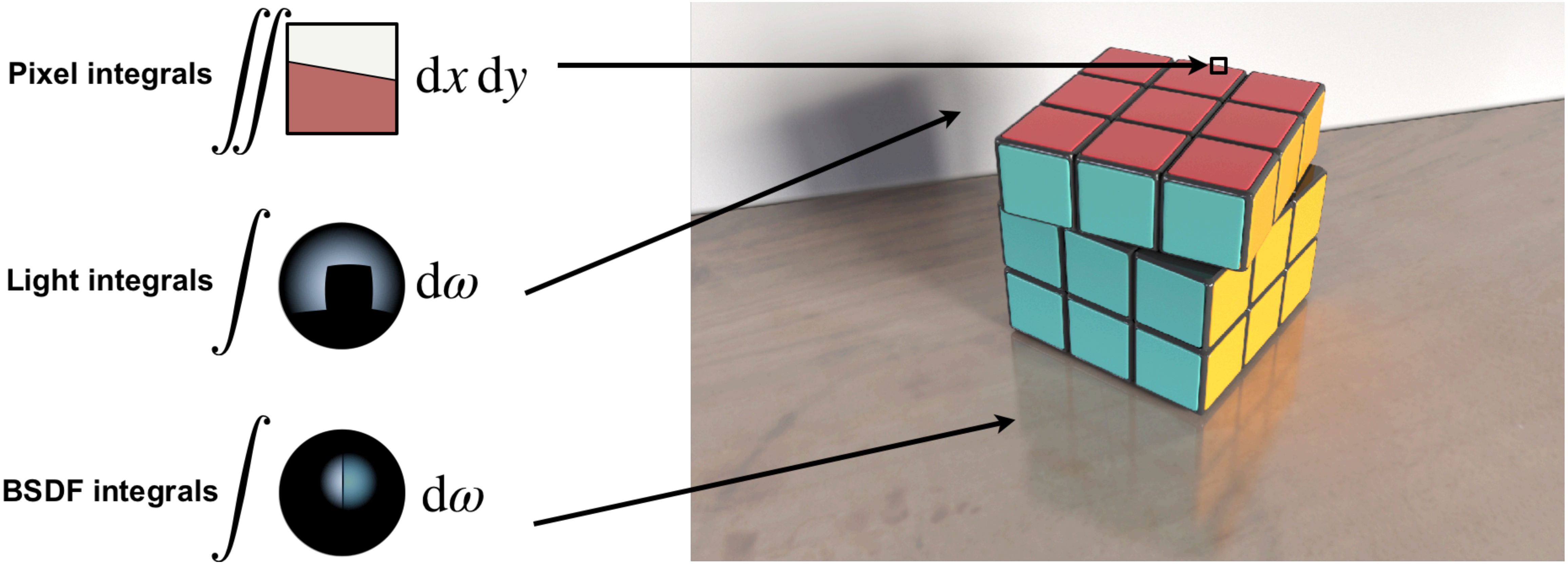
Easy to compute (e.g., automatic differentiation)

Non-differentiable integrand: No

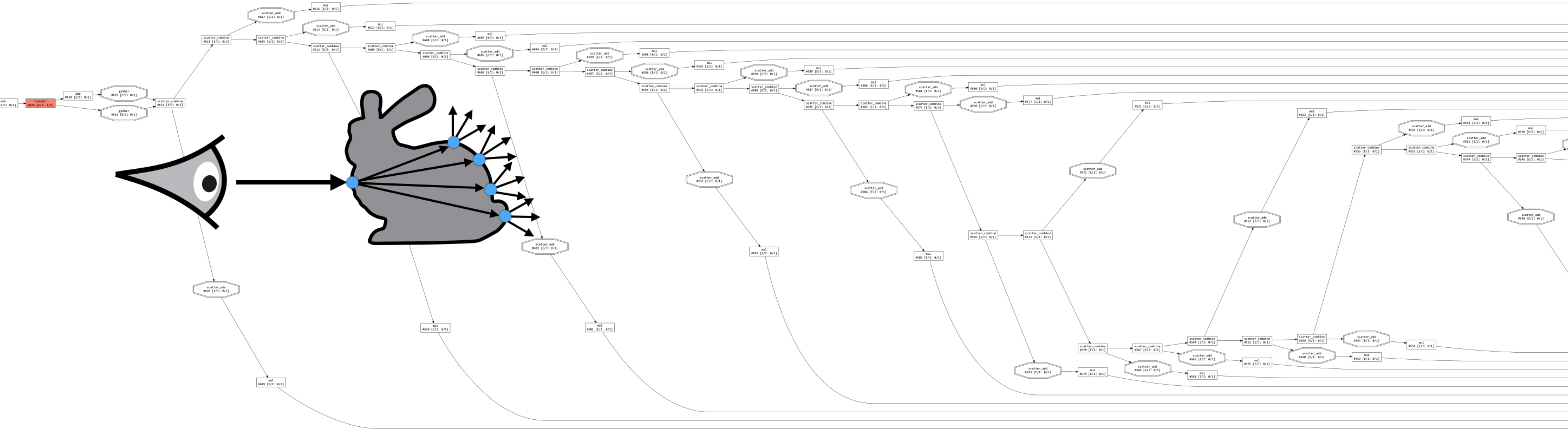


$$\left(\frac{\partial}{\partial p_i}\int \neq \int \frac{\partial}{\partial p_i}\right)$$

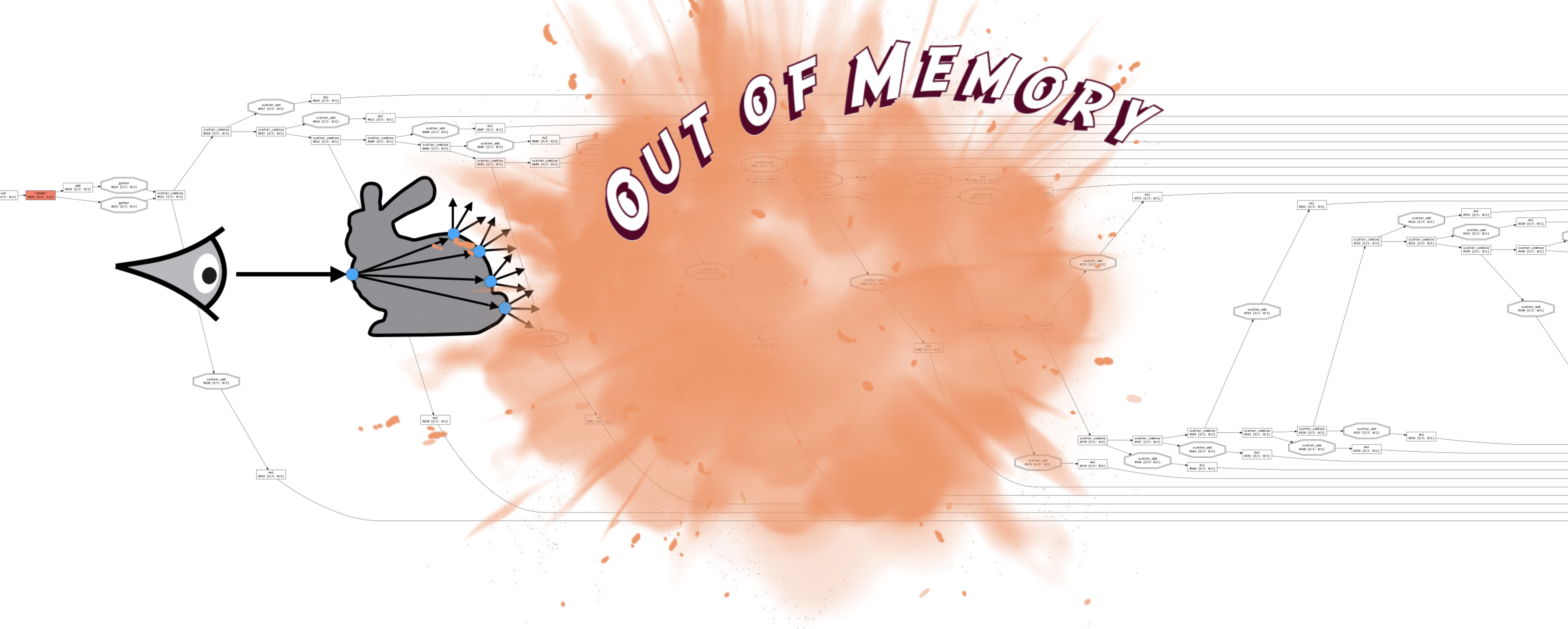
Differentiable Rendering is Challenging!



Issues with Automatic Differentiation

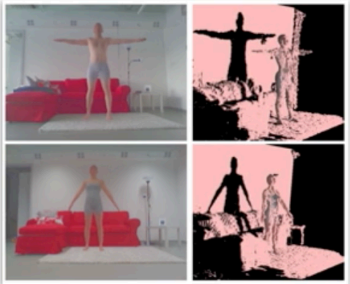


Issues with Automatic Differentiation

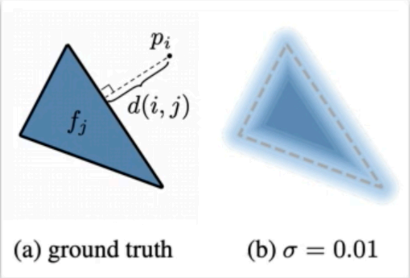


Related Work

Rasterization

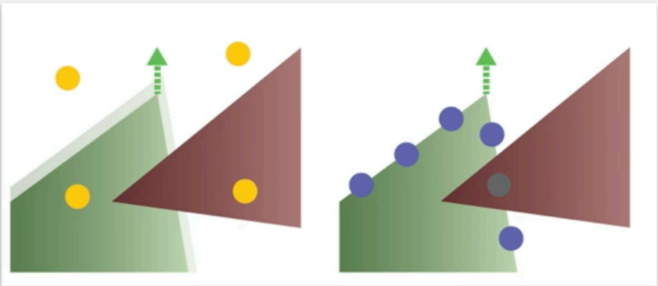


OpenDR: an Approximate Differentiable Renderer
Matthew Loper, et al.

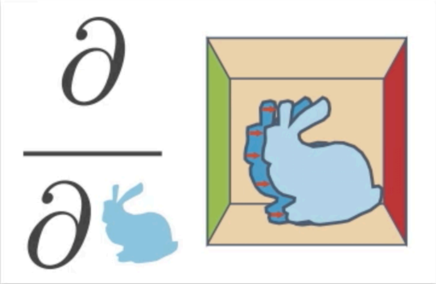


Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction
Shichen Liu, et al.

Physically based rendering



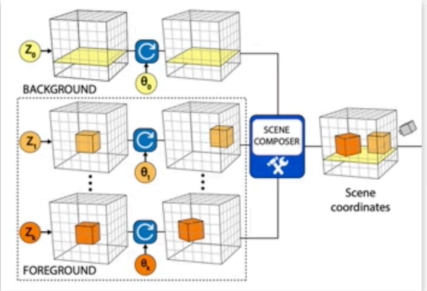
Differentiable Monte Carlo Ray Tracing Through Edge Sampling
Tzu-Mao Li, et al.



Mitsuba 2: a Retargetable Forward and Inverse Renderer
Merlin Nimier-David, et al.

Neural rendering

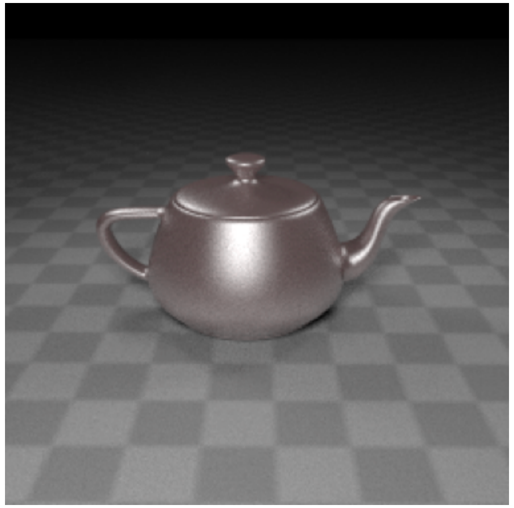
Scene Representation Networks: Continuous 3D-Structure-Aware Neural Scene Representations
Vincent Sitzmann, et al.



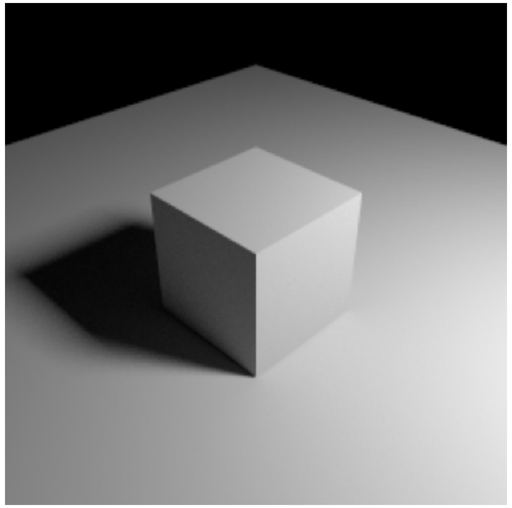
BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images
Thu Nguyen-Phuoc, et al.

Contributions

- This paper proposed a general physically-based differentiable render


glossy reflection


mirror reflection


shadow


global illumination

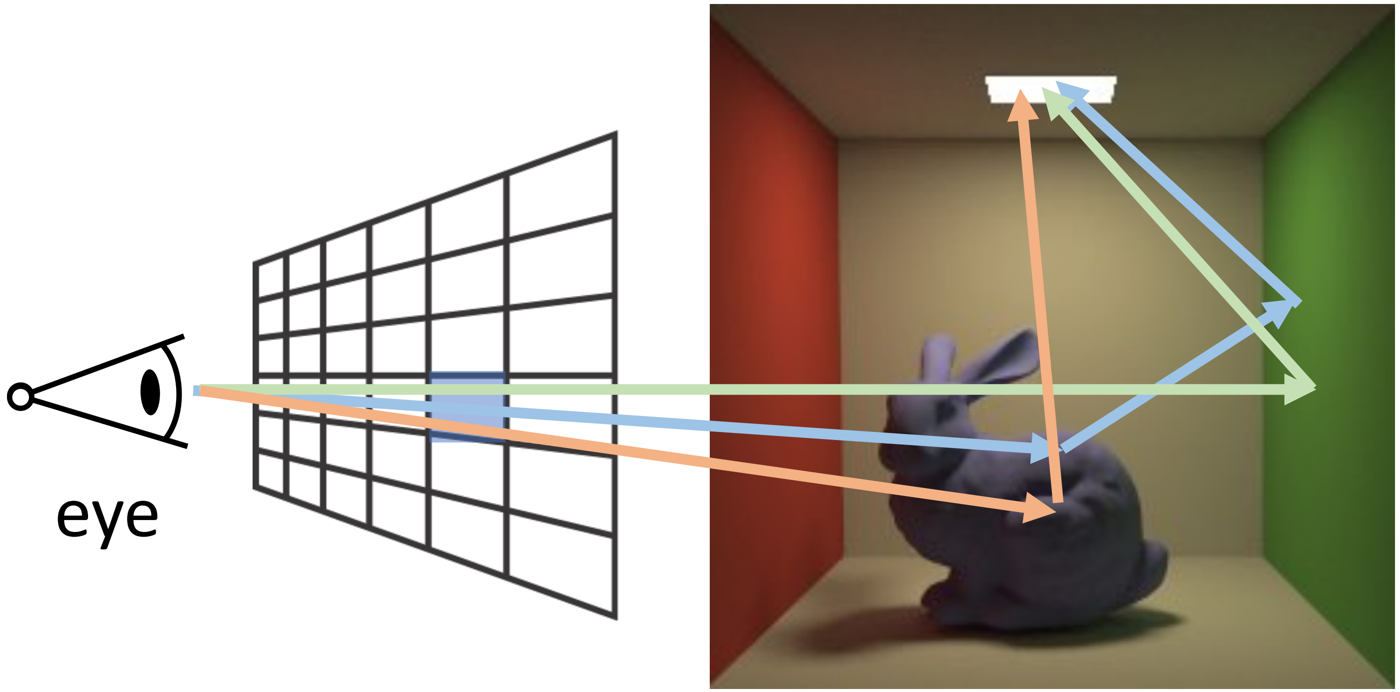
Contributions

- This paper proposes a general physically-based differentiable renderer
  - **General differentiable path tracer**
    - a stochastic approach based on **Monte Carlo** ray tracing to estimate both the integral and the gradients of the pixel filter's integral
  - **Handling geometric discontinuities**
    - a combination of standard area sampling and novel **edge sampling** to deal with smooth and discontinuous regions

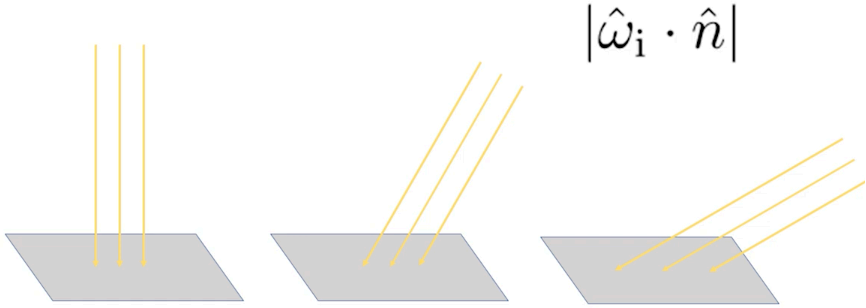
- This paper shows
  - The utility of proposed differentiable renderer in several applications (inverse rendering, 3D adversarial examples)
  - Better performance than two previous differentiable renderers (OpenDR & Neural Mesh Rendering)

Physically-based Rendering

- The Rendering Equation



The Rendering Equation



Outgoing direction

Incoming direction

$$L_{\mathrm{o}}(X, \hat{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(X, \hat{\omega}_{\mathrm{o}}) + \int_{\mathbf{S}^2} L_{\mathrm{i}}(X, \hat{\omega}_{\mathrm{i}}) \; f_X(\hat{\omega}_{\mathrm{i}}, \hat{\omega}_{\mathrm{o}}) \; |\hat{\omega}_{\mathrm{i}} \cdot \hat{n}| \; d\hat{\omega}_{\mathrm{i}}$$

A point in the scene

Surface normal

All incoming directions
(a sphere)

Credit: https://news.developer.nvidia.com/ray-tracing-essentials-part-6-the-rendering-equation/

The Rendering Equation

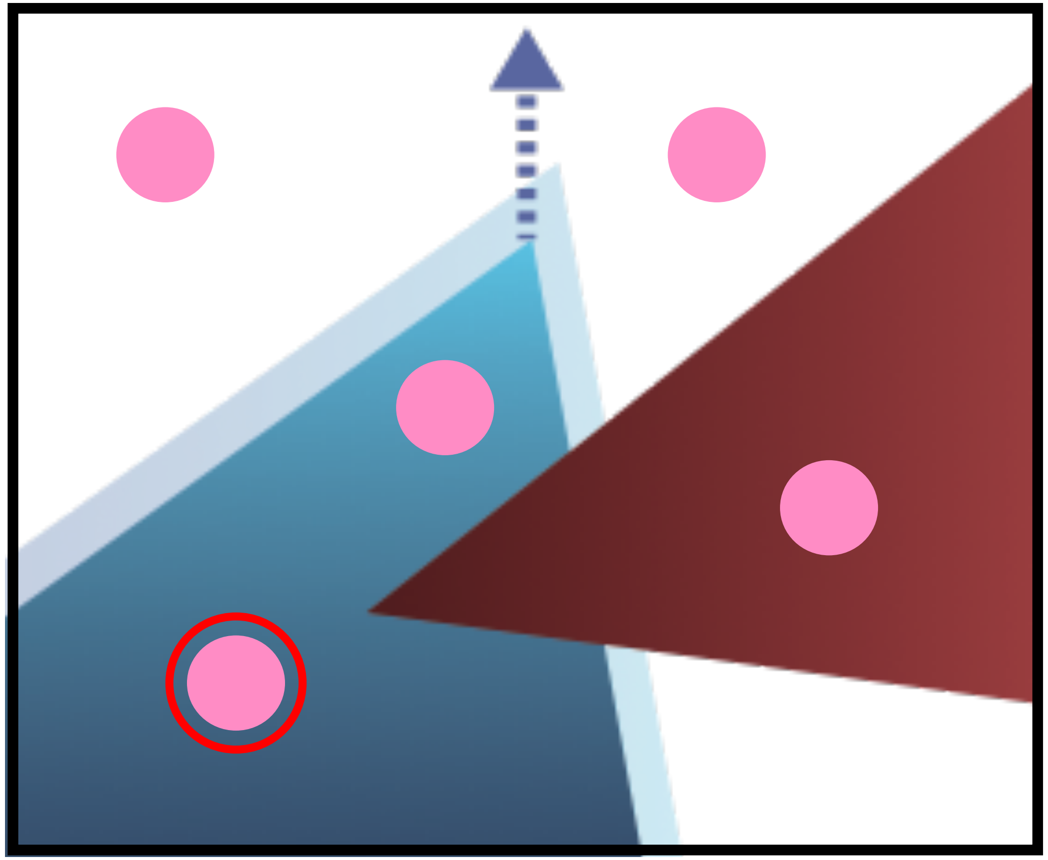
$$L_{\mathrm{o}}(X, \hat{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(X, \hat{\omega}_{\mathrm{o}}) + \int_{\mathbf{S}^2} L_{\mathrm{i}}(X, \hat{\omega}_{\mathrm{i}}) \, f_X(\hat{\omega}_{\mathrm{i}}, \hat{\omega}_{\mathrm{o}}) \, |\hat{\omega}_{\mathrm{i}} \cdot \hat{n}| \, d\hat{\omega}_{\mathrm{i}}$$

Outgoing light | Emitted light | Incoming light | Material | Lambert

Credit: https://news.developer.nvidia.com/ray-tracing-essentials-part-6-the-rendering-equation/

Rendering = Sampling

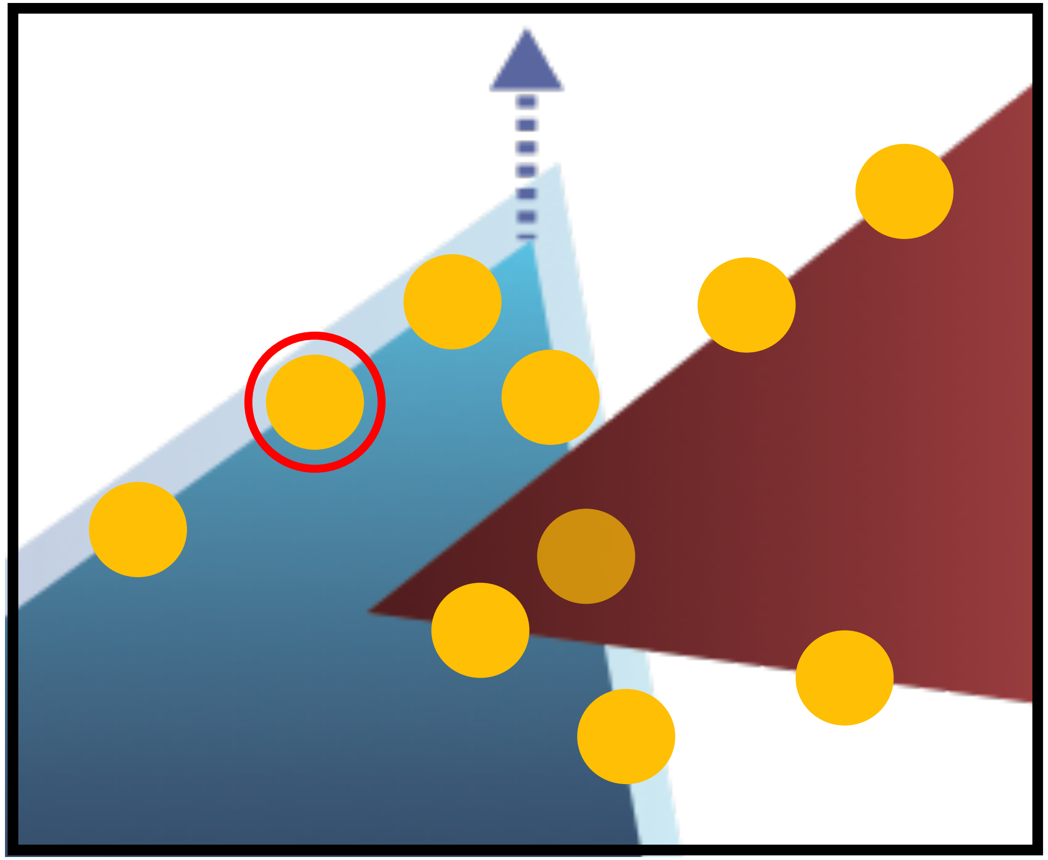
color change
when blue triangle
moves up?



pixel

Key idea: Explicitly integrate the boundaries

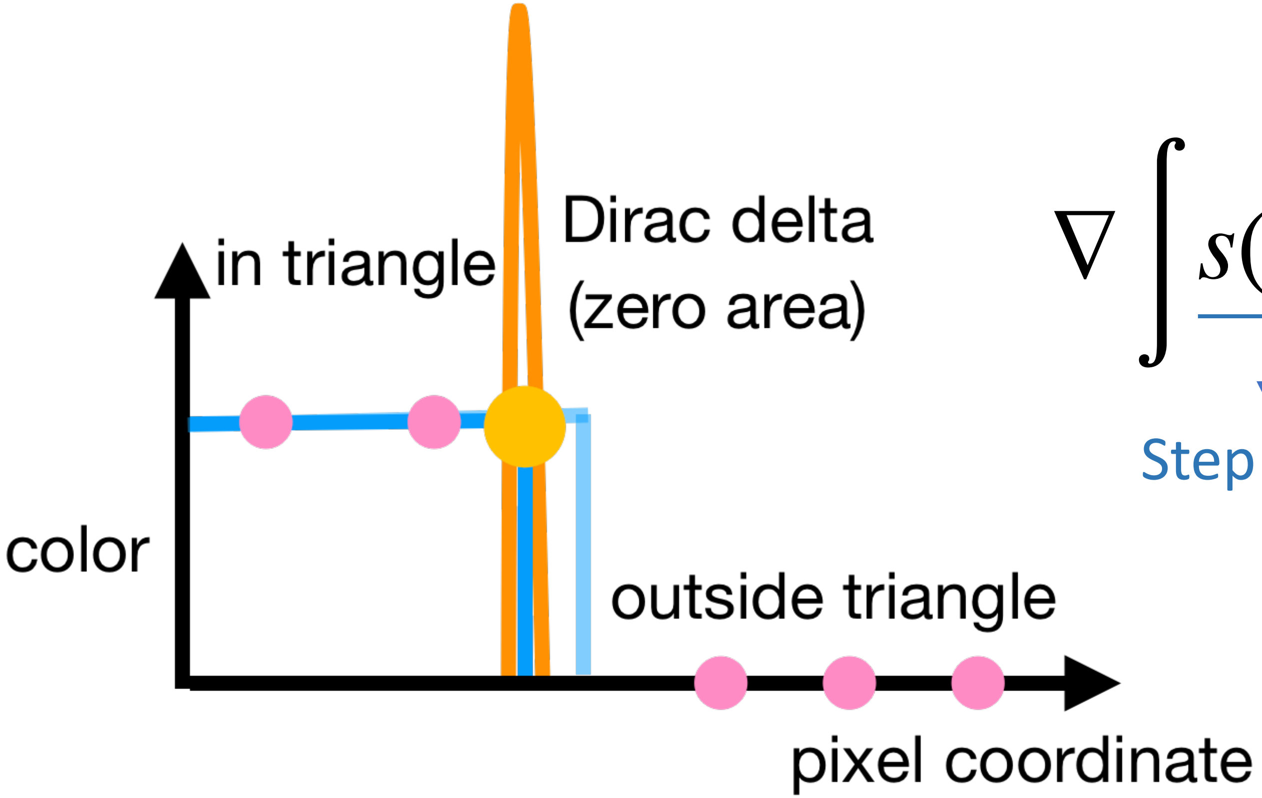
color change
when blue triangle
moves up?



pixel

Mathematical formulation

- Model the edge as the step function
- Each pixel is an integral over the step functions



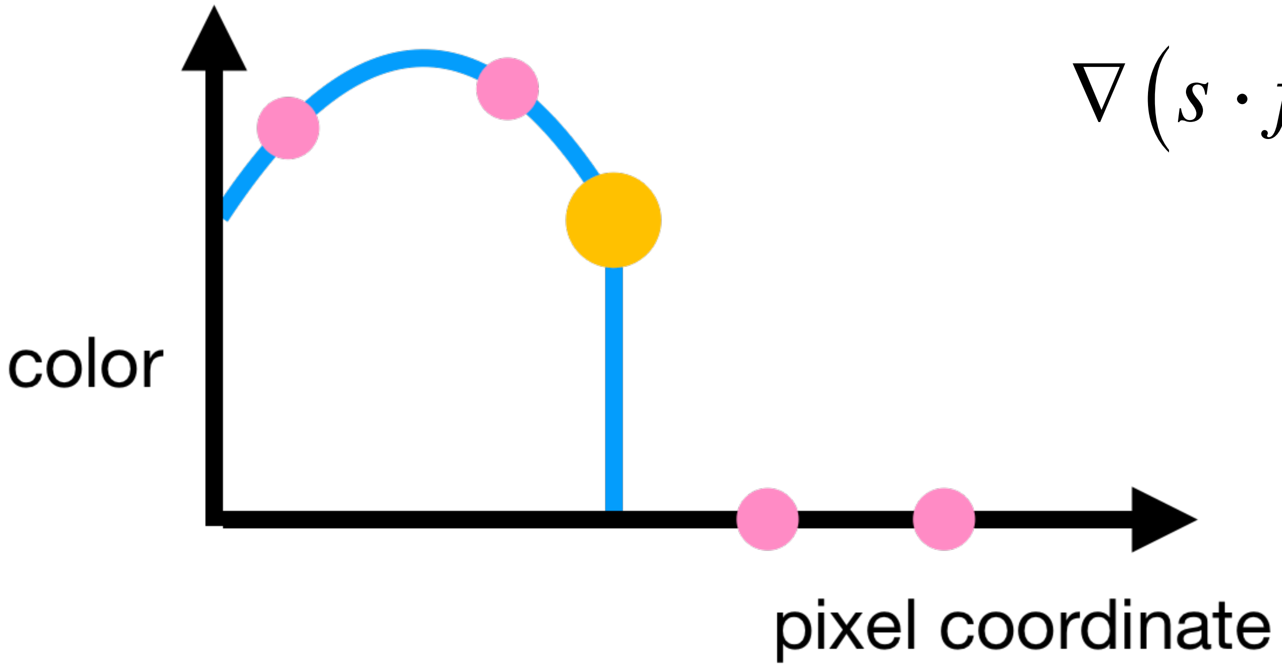
$$\nabla \int \underline{s(x)} dx = \int \nabla s(x) dx \qquad \underline{\delta(x)}$$

Step function

Dirac delta

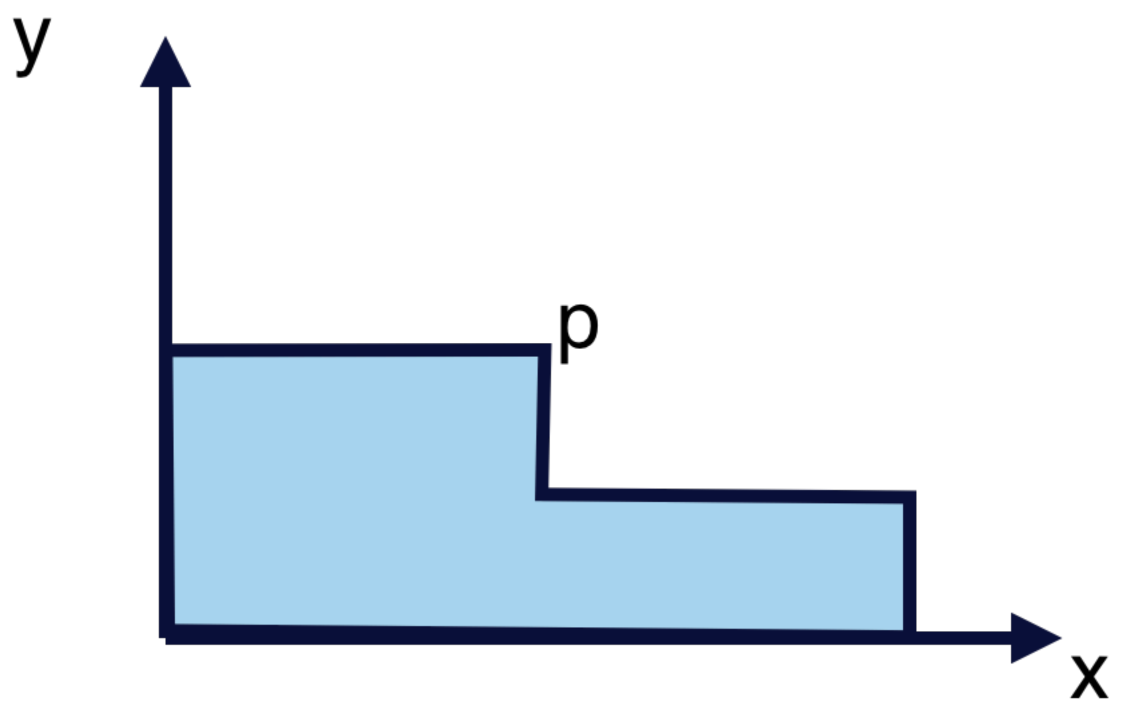
Mathematical formulation

- A smooth shading function f multiples to the step function s



$$\nabla\left(s \cdot f\right) = \underbrace{(\nabla s)}_{\text{Dirac delta}} \cdot f + s \cdot \underbrace{\left(\nabla f\right)}_{\text{Shading derivatives}}$$

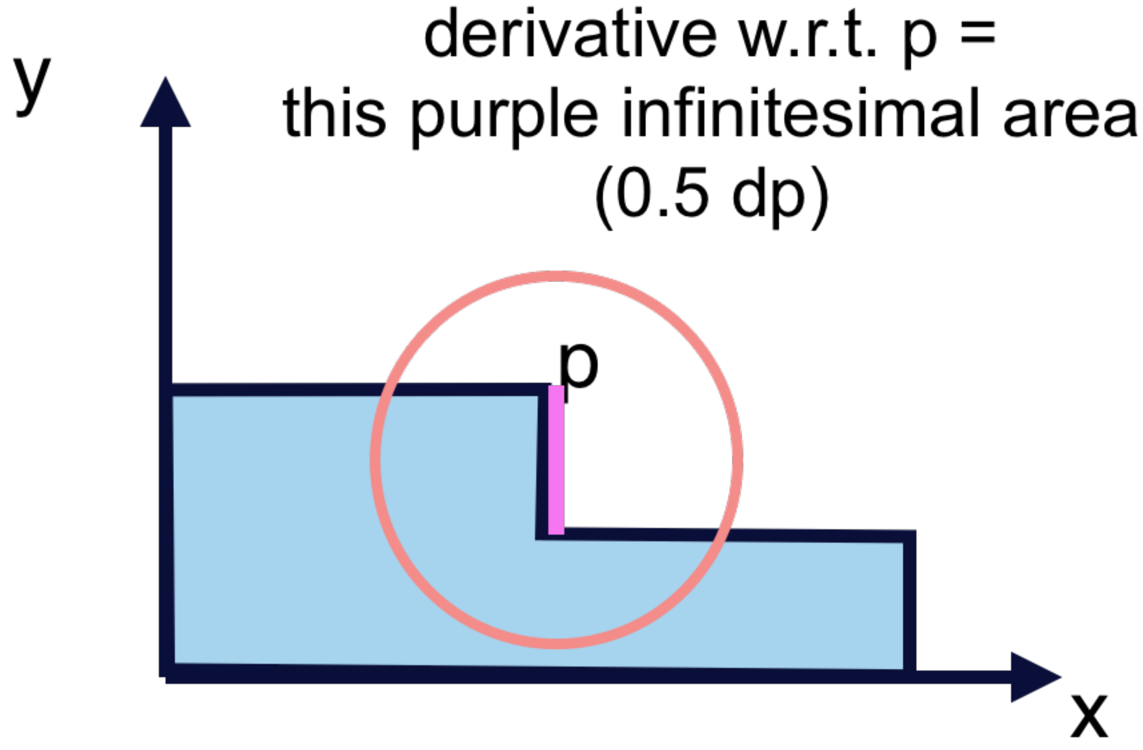
1D Derivatives



(the blue area)

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

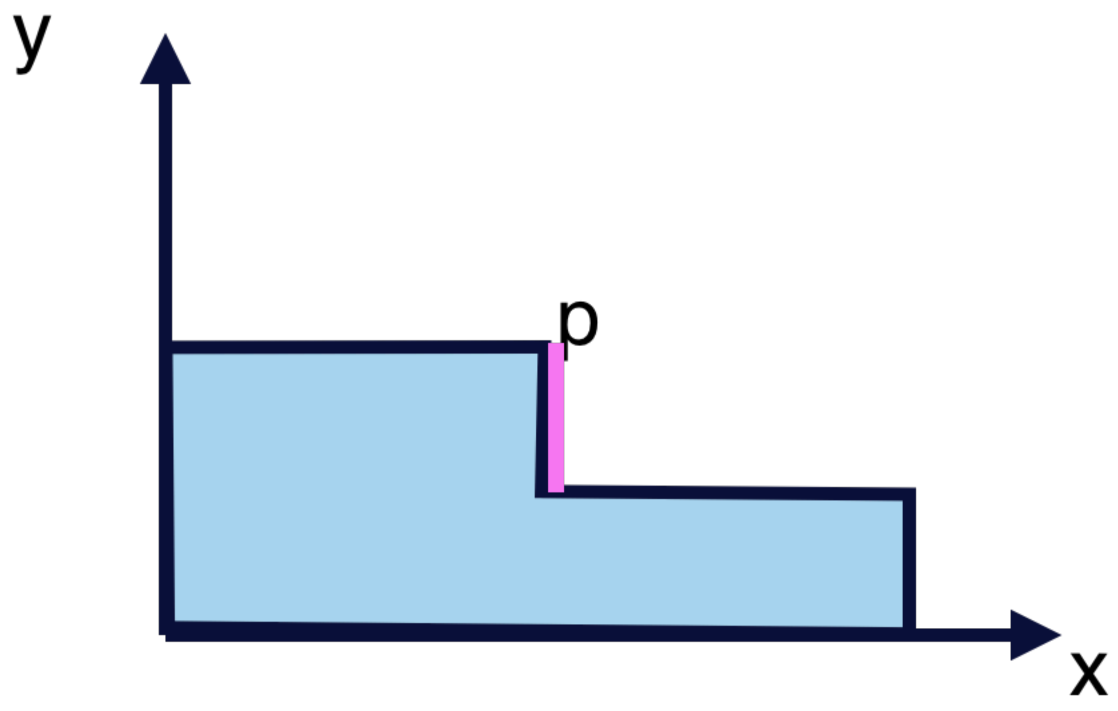
1D Derivatives



(the blue area)

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

- Trick: move the discontinuities to the integral boundaries



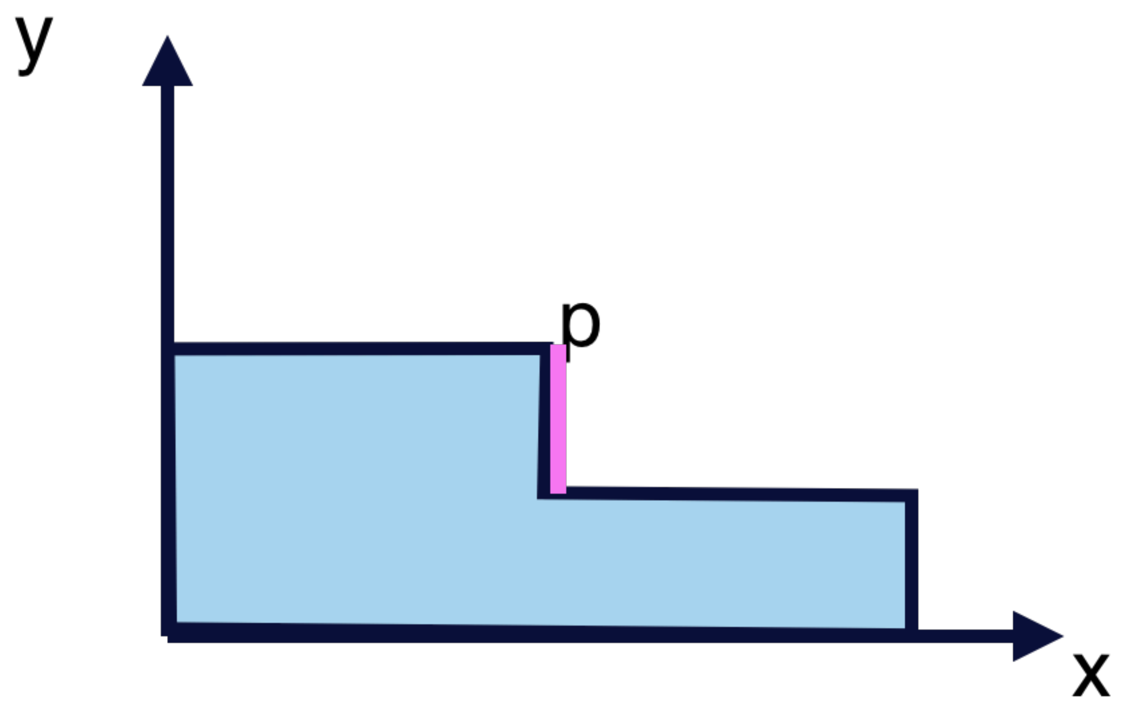
(the blue area)

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

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

1D Derivatives

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

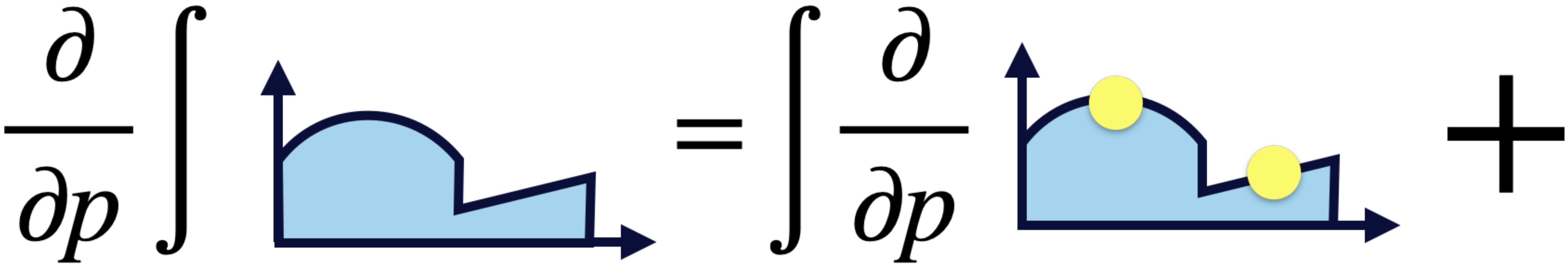


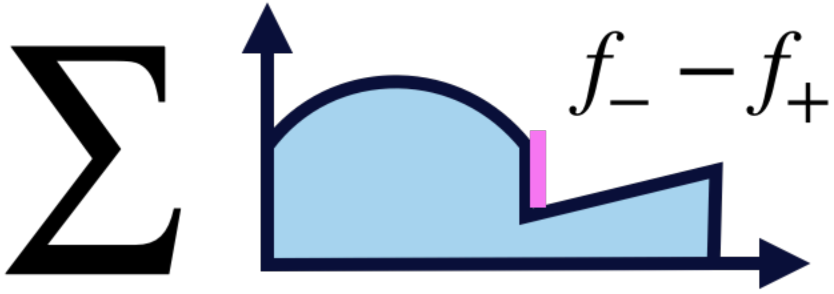
(derivative of blue area w.r.t. p)

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

$$= 1 - 0.5$$

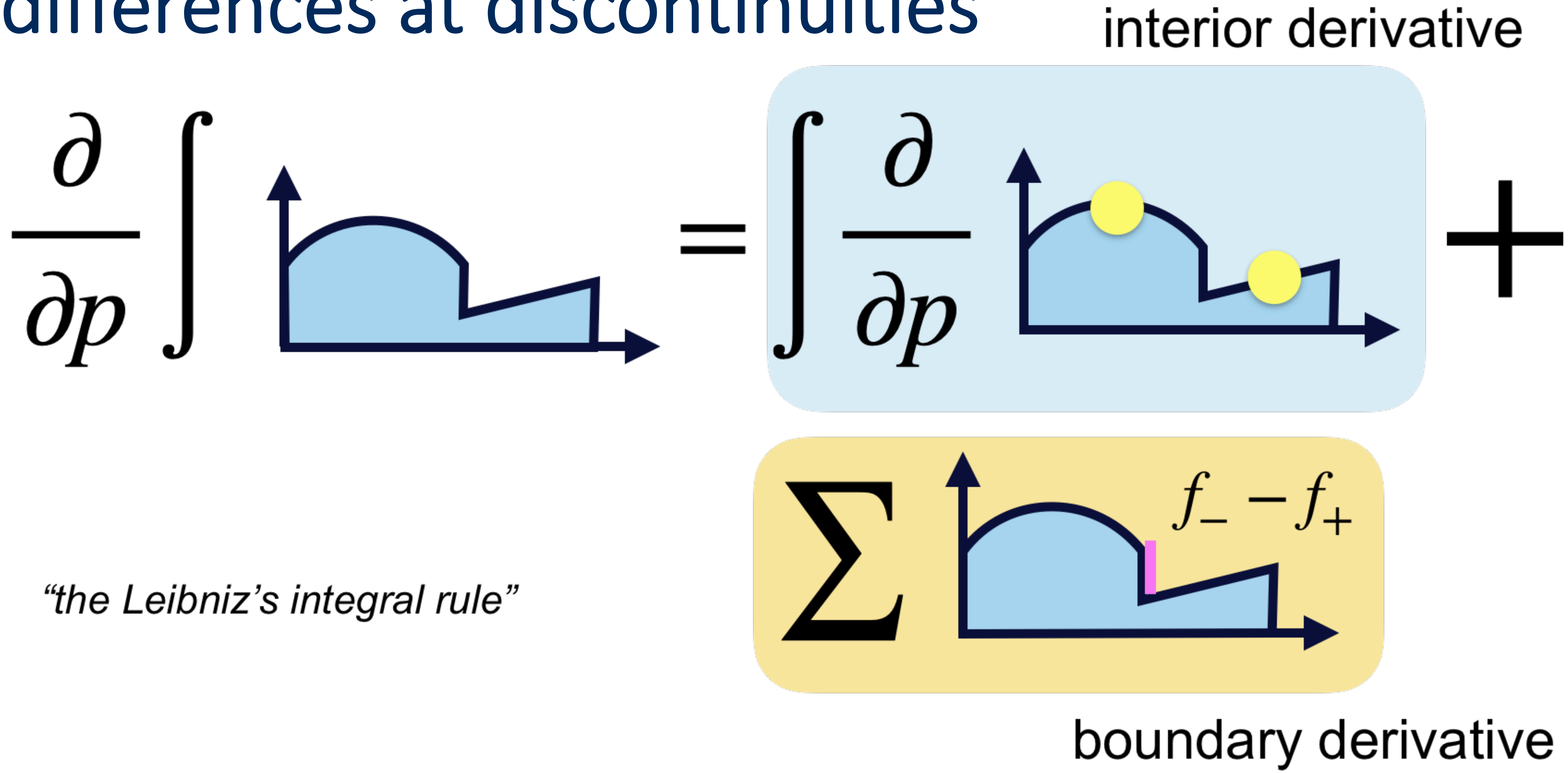
Discontinuity derivatives = differences at discontinuities



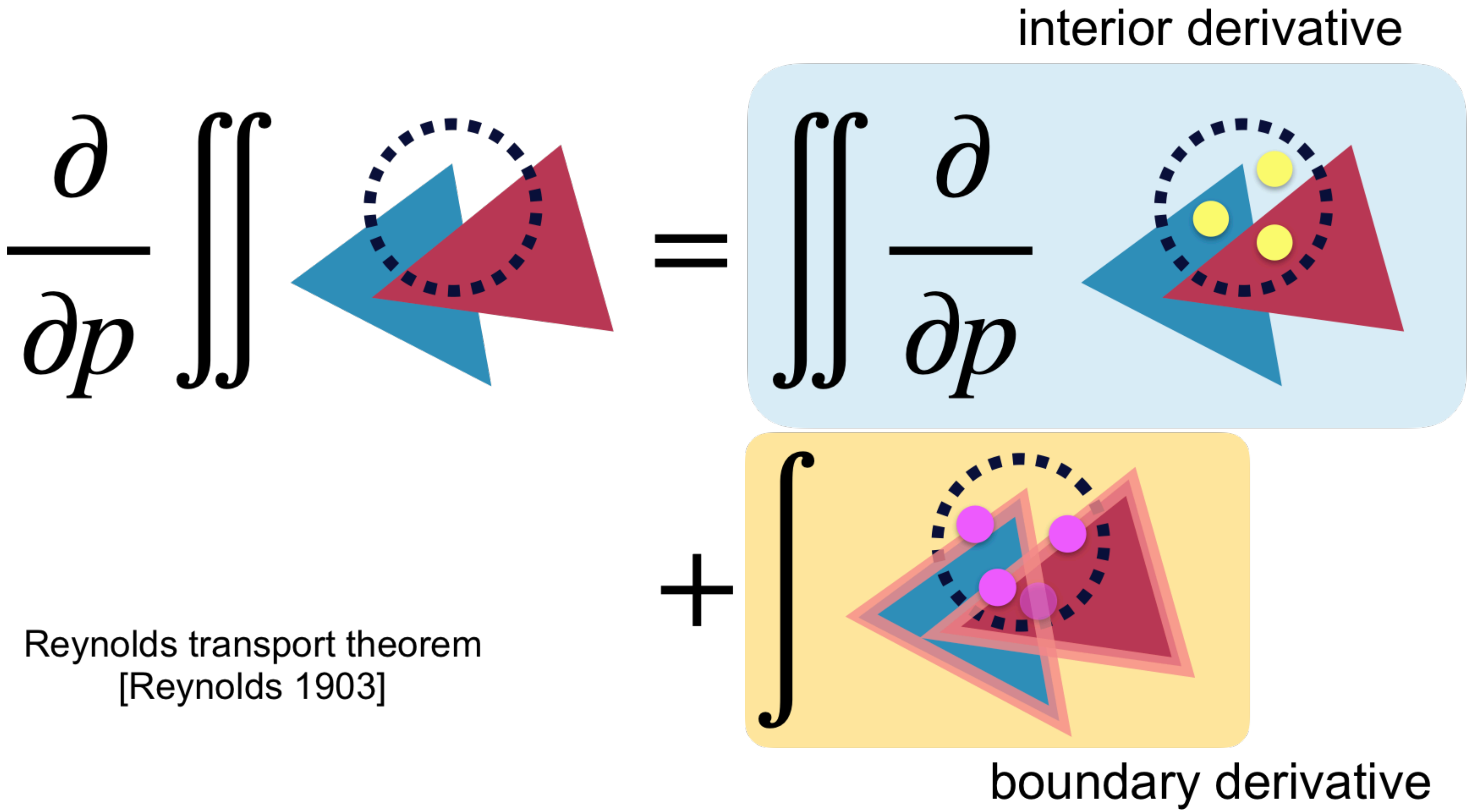


"the Leibniz's integral rule"

Discontinuity derivatives = differences at discontinuities



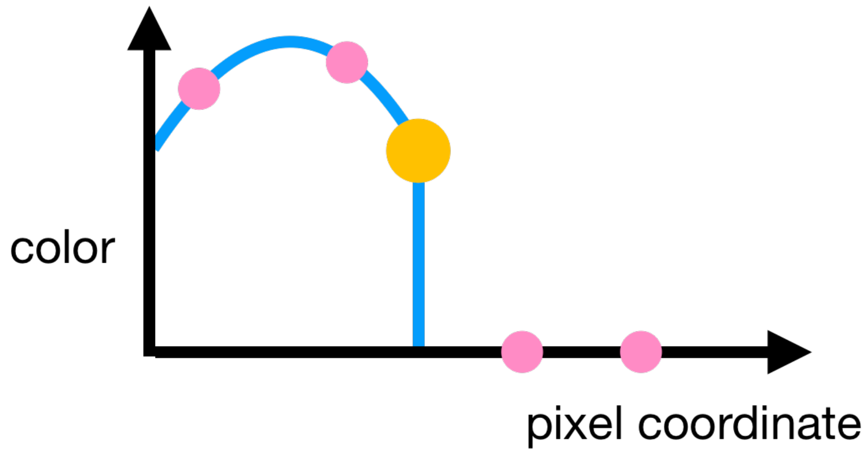
"the Leibniz's integral rule"



Reynolds transport theorem
[Reynolds 1903]

Mathematical formulation

- Scene function $f(x, y; \Phi)$
- Pixel Color $I = \iint f(x, y; \Phi) dxdy$
- Gradient $\nabla I = \nabla \iint f(x, y; \Phi) dxdy$

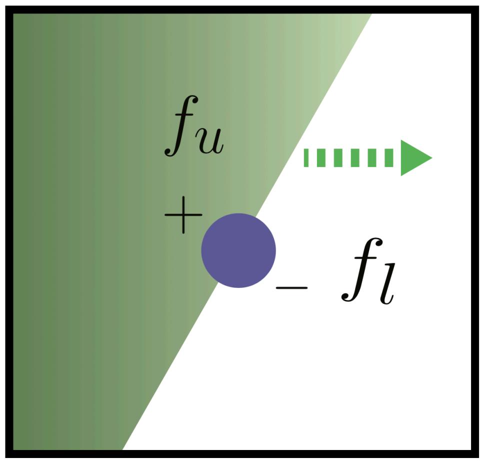


- All discontinuities happen in the scene edges

$$\alpha(x, y) = Ax + By + C$$

$$f(x, y; \Phi) = \theta(\alpha(x, y)) f_u(x, y; \Phi) + \theta(-\alpha(x, y)) f_l(x, y; \Phi)$$

$$I = \iint f(x, y; \Phi) dxdy = \sum_i \iint \theta\left(\alpha_i(x, y)\right) f_i(x, y; \Phi) dxdy$$

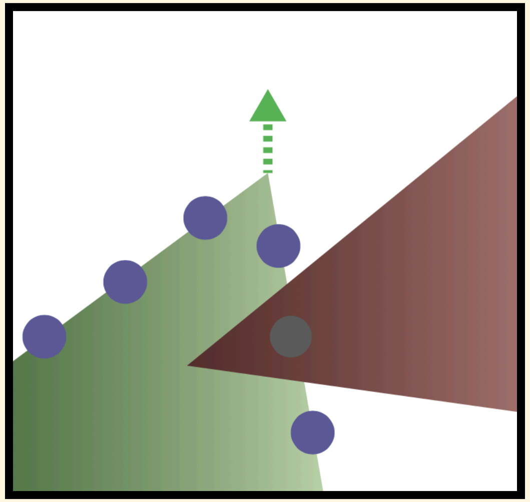


Mathematical formulation

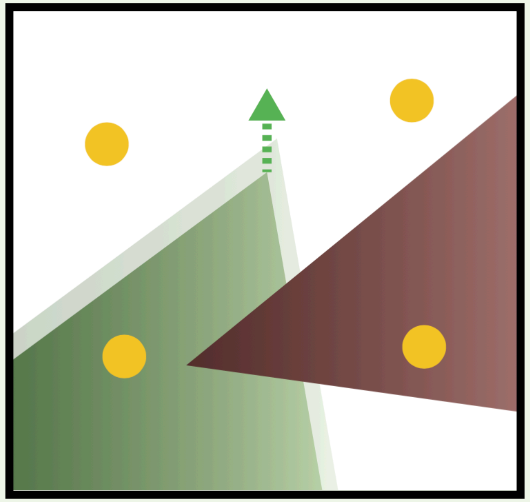
- Using the Chain rule

$$\nabla \iint \theta(\alpha(x, y))f(x, y; \Phi)dxdy = \iint \delta(\alpha(x, y))\nabla \alpha(x, y)f(x, y; \Phi)dxdy + \iint \nabla f(x, y; \Phi)\theta(\alpha(x, y))dxdy$$

Edge sampling

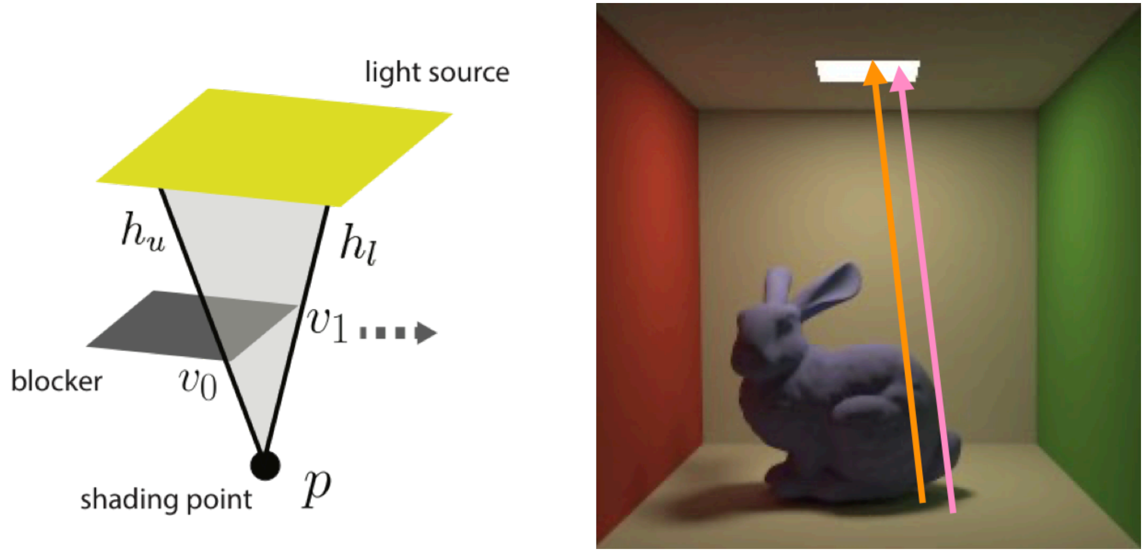


Area sampling

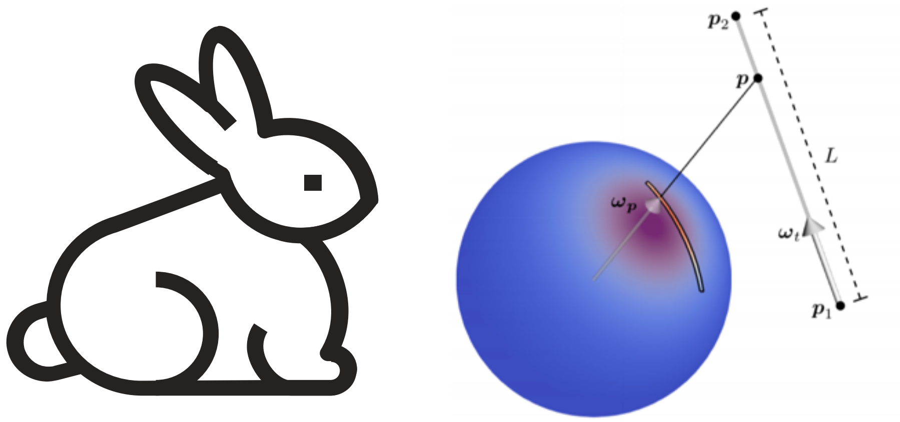


Generalization & Scalability

- Generalizable to shadow & interreflection
- Use importance sampling to sample edges and pick points (Hill and Heitz 2017)



area of a light source



select an edge & pick a point

Algorithms

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

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

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

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

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

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

Standard PT w/ symbolic differentiation

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

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

Monte Carlo edge sampling

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

Rendering equation

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

Differential rendering equation

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

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

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

Algorithms

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

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

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

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

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

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

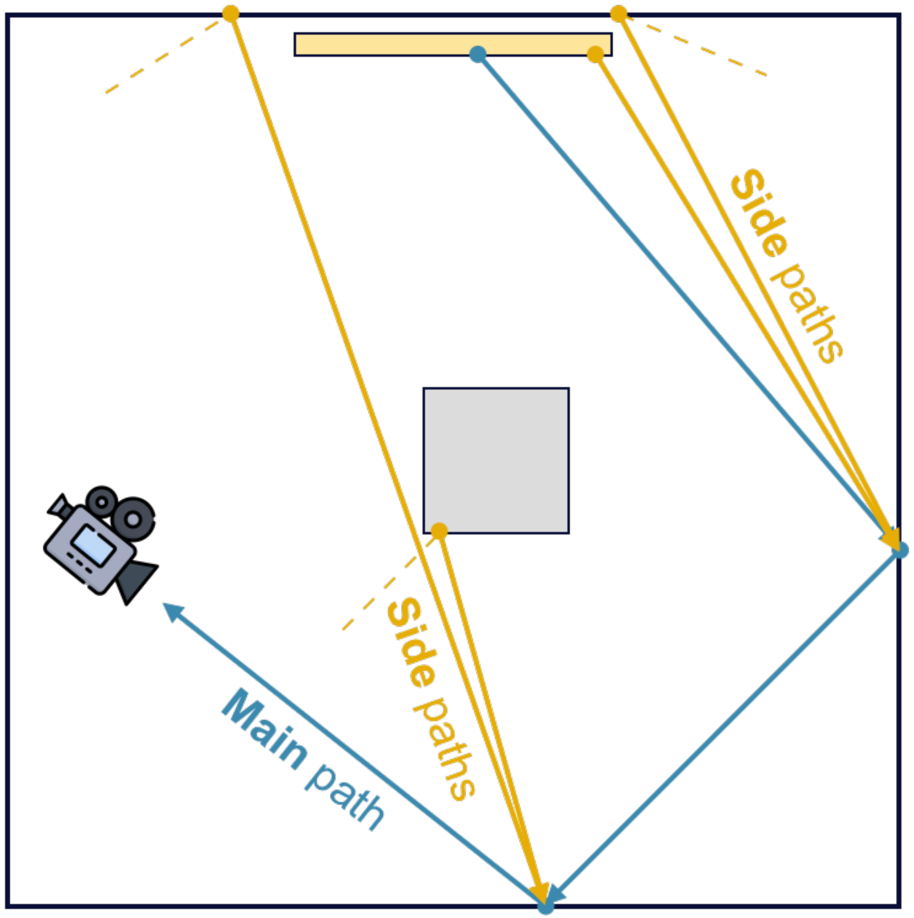
Standard PT w/ symbolic differentiation

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

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

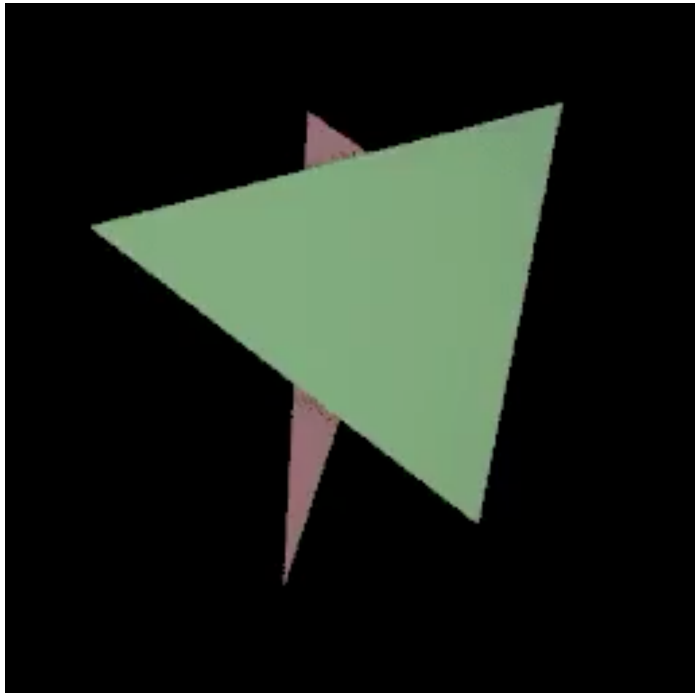
Monte Carlo edge sampling

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



Experiments – Synthetic examples

- Optimizing 6 triangle vertices

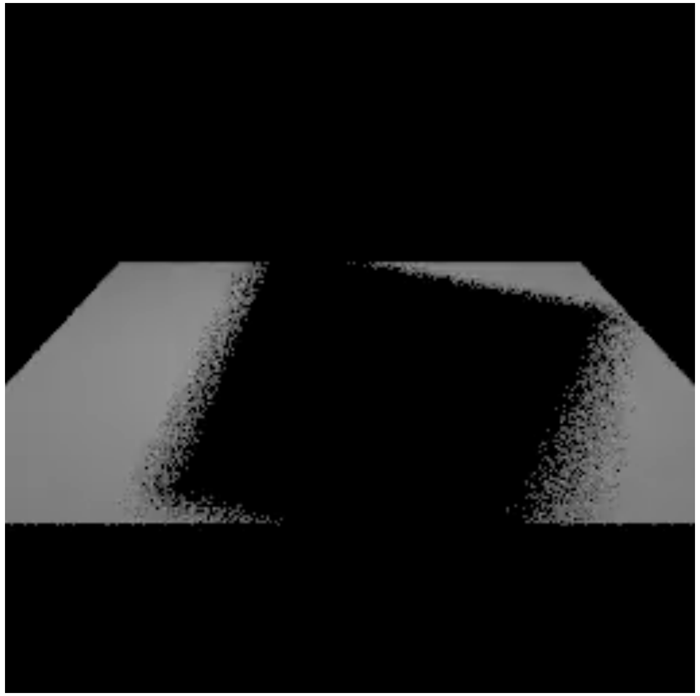


Source

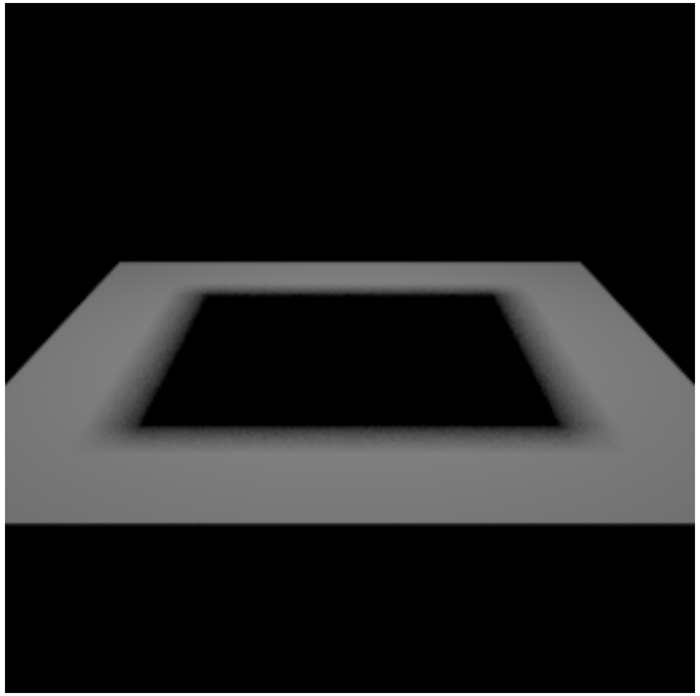
Target

Experiments – Synthetic examples

- Optimizing blocker vertices



Source



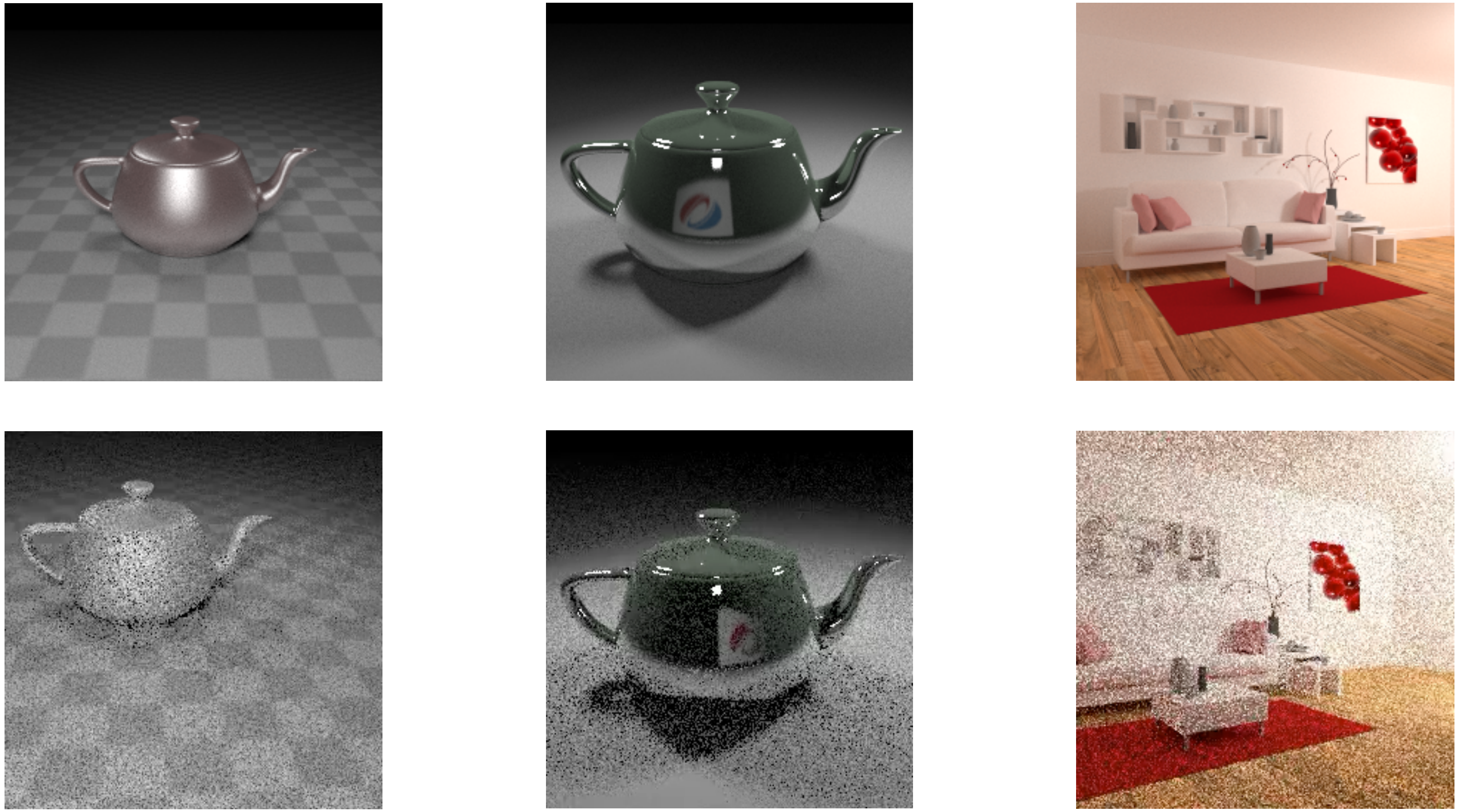
Target

Experiments – Synthetic examples

camera & teapot material | logo translation | camera

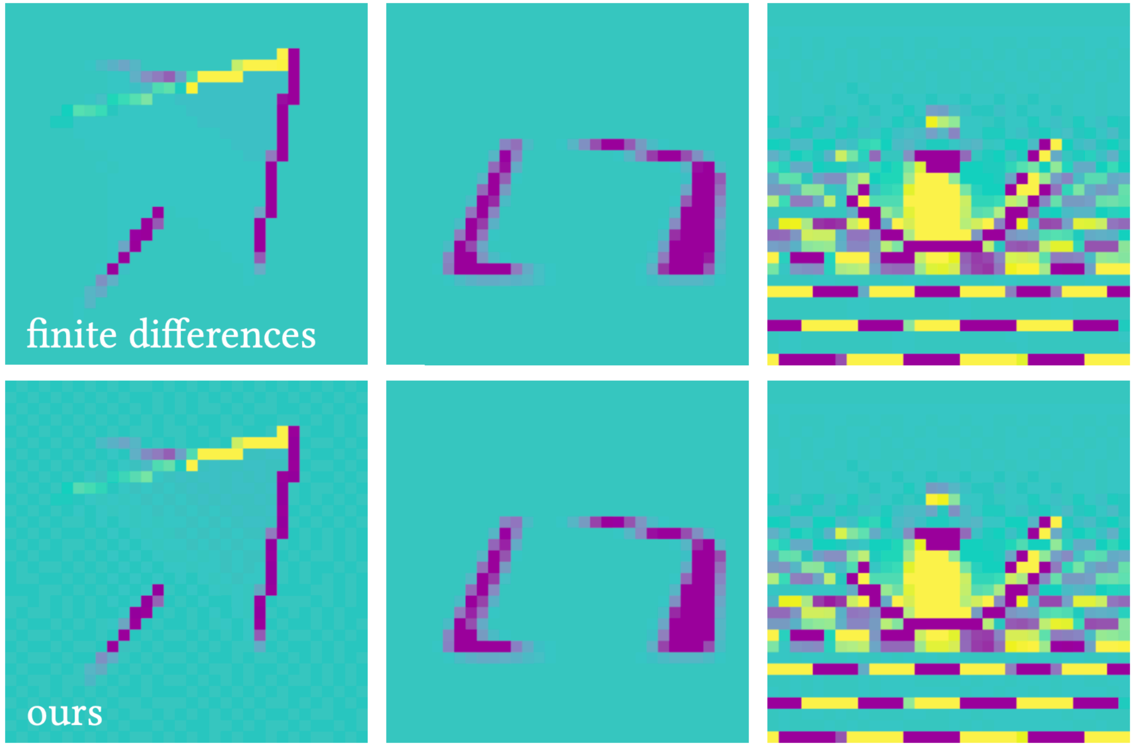
Target

Source



Experiments – Synthetic examples

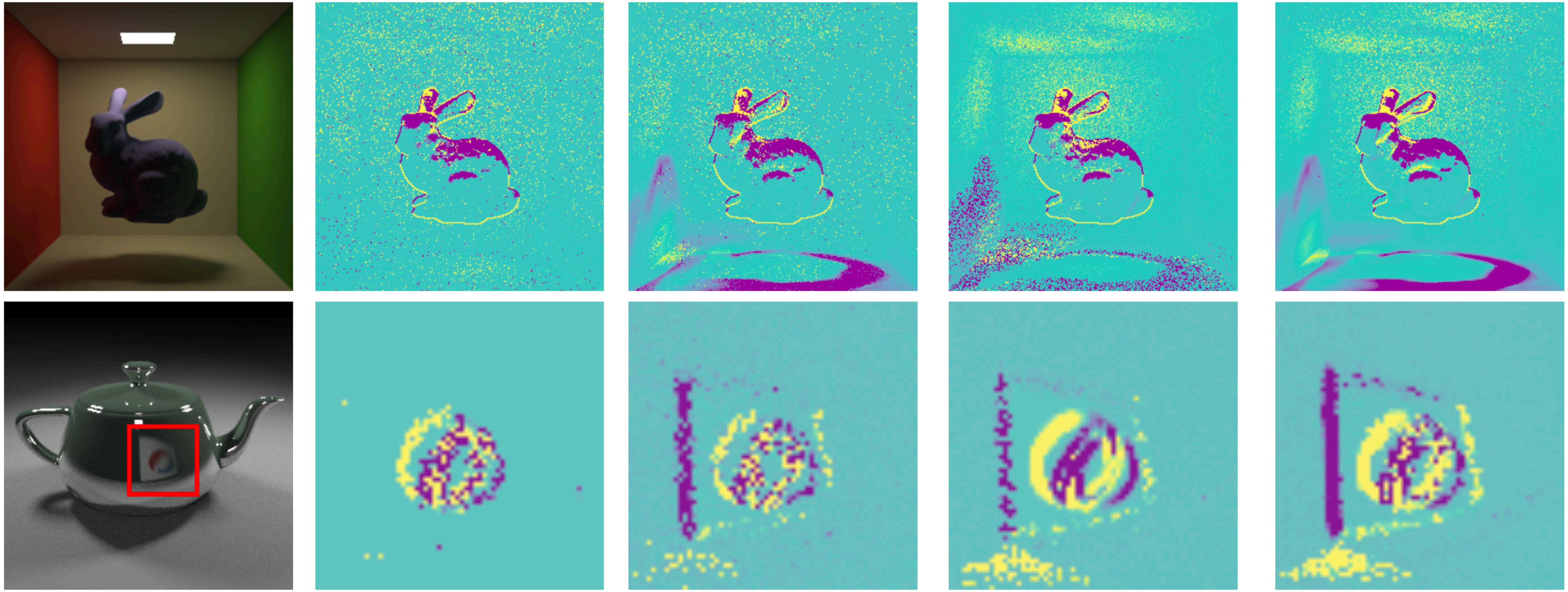
- Compare with central finite differences (32 x 32 scenes)



(a) triangles (b) shadow (c) teapot

Experiments – Synthetic examples

- Sampling with or without edge importance sampling



scenes 10s, w/o importance samp. 10s, w/ importance samp. 350s, w/o importance samp. 350s, w/ importance samp.

Experiments – Inverse rendering

- Optimizing camera pose, light emission and materials

initial guess

target

reconstructed

Experiments – Inverse rendering

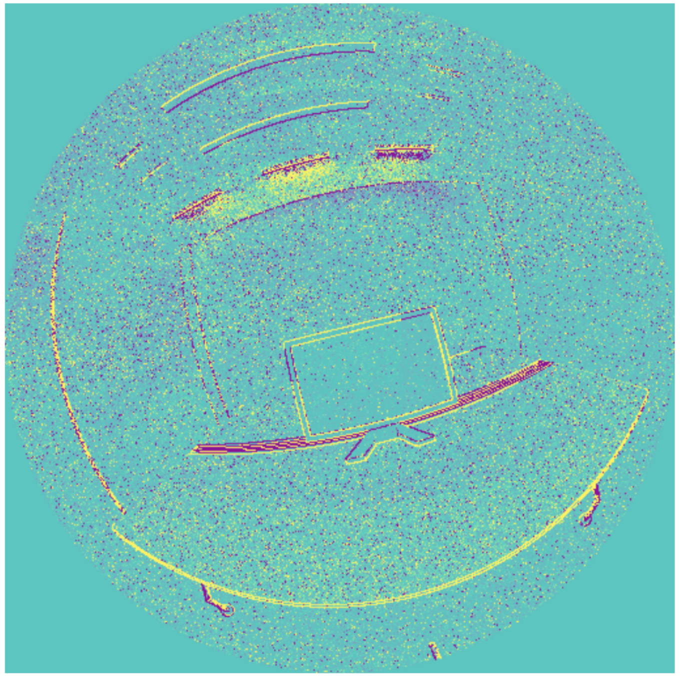
- Optimizing camera pose, light emission and materials

optimization

target

Experiments – Inverse rendering

- Optimizing camera pose, light emission and materials



camera gradient

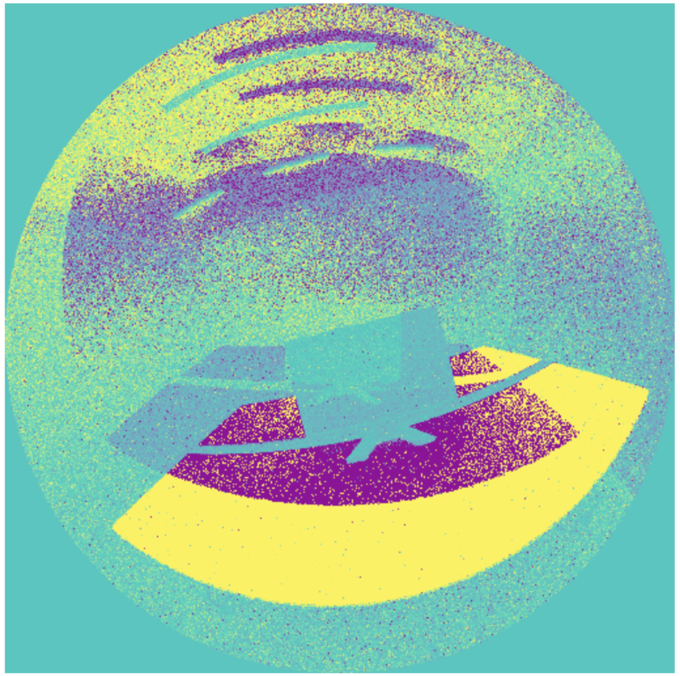
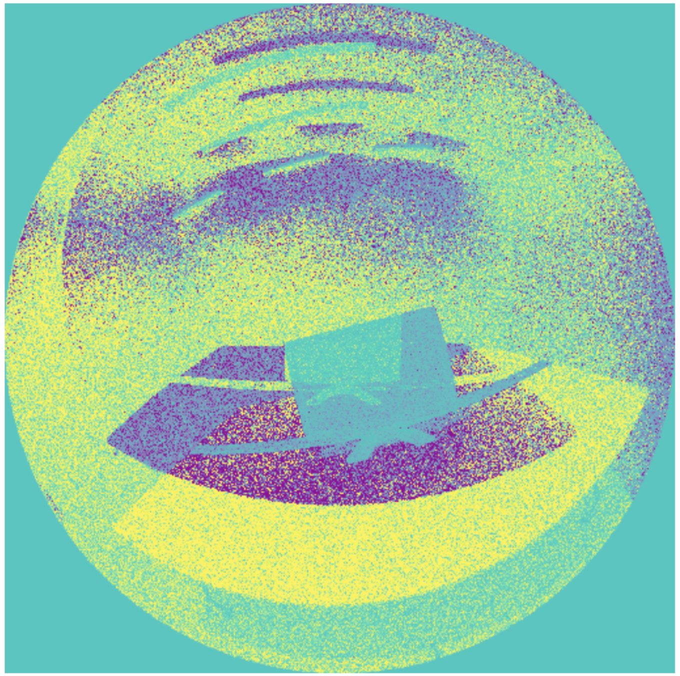


table albedo gradient



light gradient

Experiments – 3D adversarial examples

- Optimizing vertex position, camera pose, light intensity, position

VGG 16:
53% street sign
6.7% handrail

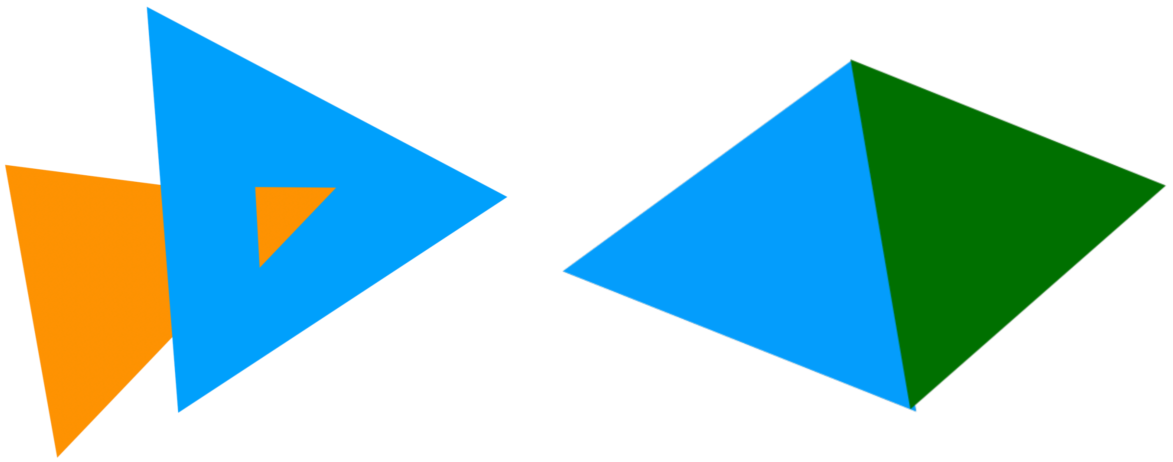
5 iterations:
26.8% handrail
20.2% street sign

25 iterations:
23.3% handrail
3.4% street sign

Limitations

- **Performance** (rendering speed & large variance):
  - Edge sampling and auto differentiation are slow (bottleneck)
  - It is a challenging task to find all object edges and sampling them

- **Assumptions:**
  - Interpenetrating geometries
  - parallel edges (non-differentiable)
  - Surface only light transport

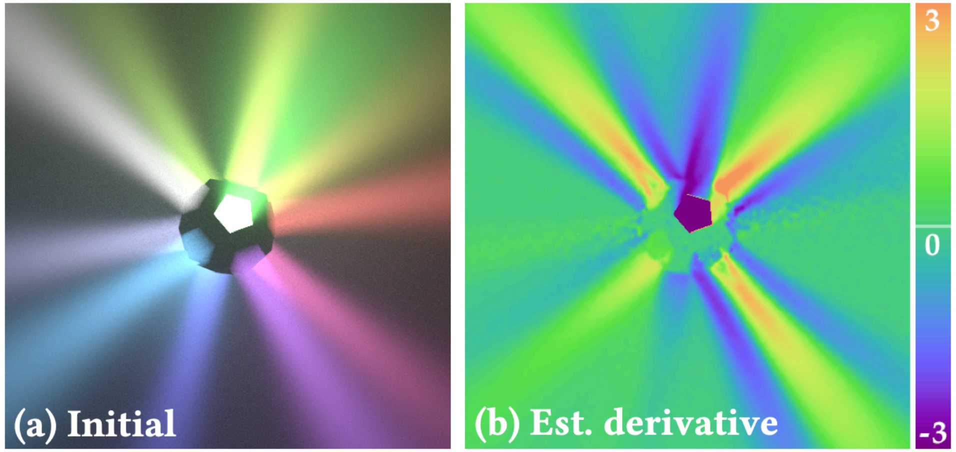


Contributions (recap)

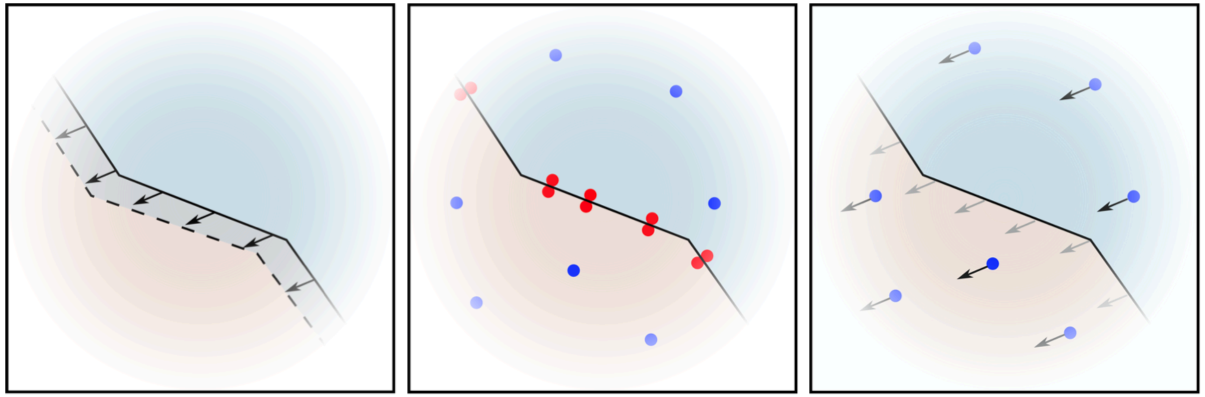
- Previous works
  - Differentiable rendering that targets specific cases (faces, hands, etc.) => *hard to generalize*
  - Fast, approximate general renderers (OpenDR, Neural Mesh Rendering) => *simplified models*
  - **challenges:** estimating the derivative corresponding to the integral of the rendering equation
- This paper proposes a general physically-based differentiable renderer
  - **General differentiable path tracer**
  - **Handling geometric discontinuities**
- This paper shows
  - The utility of proposed differentiable renderer in several applications (inverse rendering, 3D adversarial examples)
  - Better performance than two previously proposed differentiable renderers

Follow-up works

- Addressing the discontinuity problem in the rendering equation

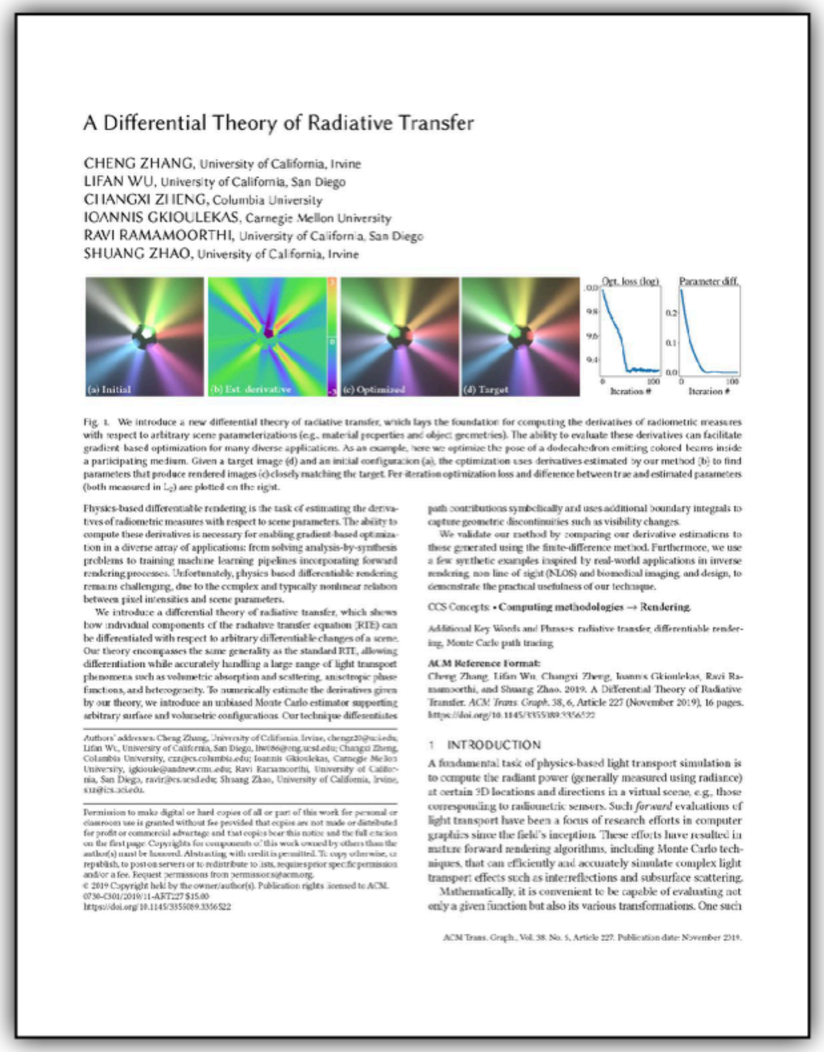


Handle volumetric light transport (Zhang et al., 2019)



(a) Integrand with discontinuity
(b) Edge sampling [Li et al. 2018]
(c) Using changes of variables (ours)

Re-parameterize the integral (Loubet et al., 2019)


A Differential Theory of Radiative Transfer

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

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

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

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

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

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

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

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

1 INTRODUCTION

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

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

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

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

Transport operator | Collision operator | Source

$$L = K_T K_c L + Q$$

Radiative transfer equation (RTE) in operator form

A Differential Theory of Radiative Transfer

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

SIGGRAPH Asia 2019

Challenges

Rendering equation

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

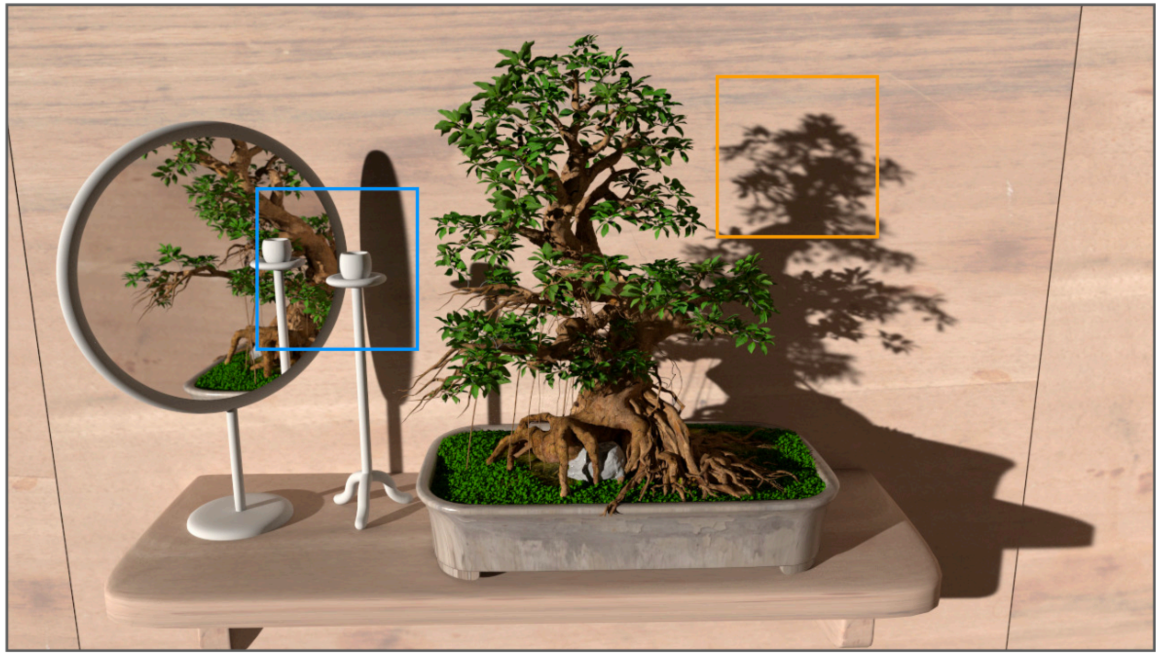
Differential rendering equation

Interior integral | Boundary integral

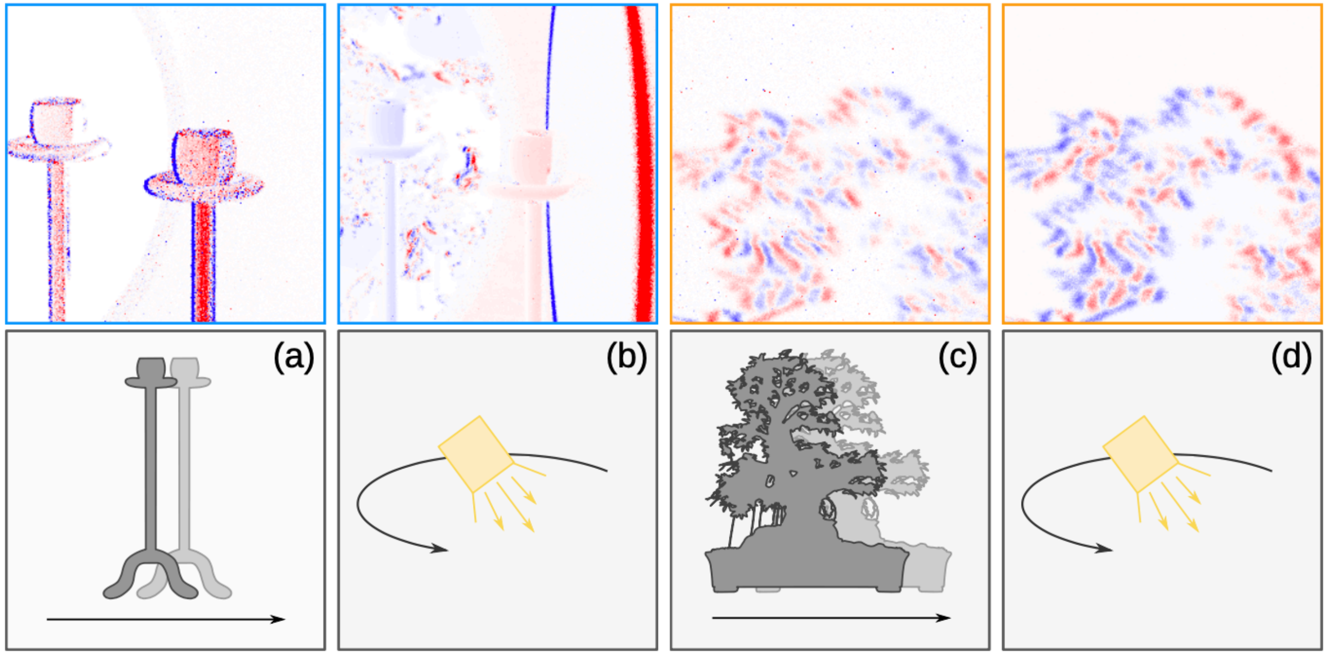
$$\frac{\mathrm{d}}{\mathrm{d}\pi} L(\boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} \frac{\mathrm{d}}{\mathrm{d}\pi} f_\mathrm{RE}(\boldsymbol{\omega}_\mathrm{i}) \, \mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}) + \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\boldsymbol{\omega}_\mathrm{i})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

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

Reparameterizing Discontinuous Integrands for Differentiable Rendering



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

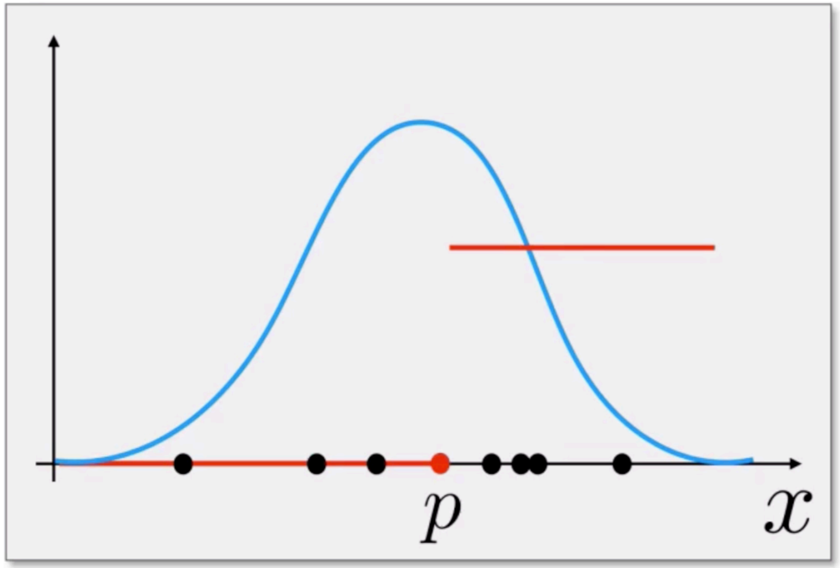


Gradients with respect to scene parameters that affect visibility

https://doi.org/10.1145/3355089.3356510

of objective functions hold significant untapped potential in areas

Key Idea: Re-parameterizing Integrals



$$I = \int k(x) \mathbb{1}_{x>p} \, \mathrm{d}x \qquad \frac{\partial I}{\partial p} = ?$$

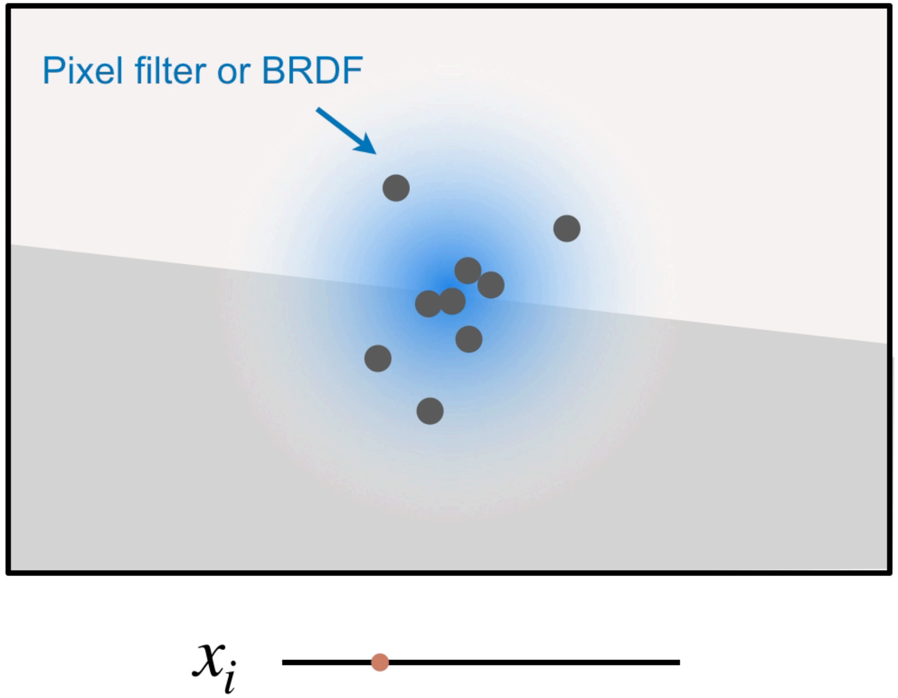
Change of variable: $X = x - p$

$$I = \int k(X+p) \mathbb{1}_{X>0} \, \mathrm{d}x$$

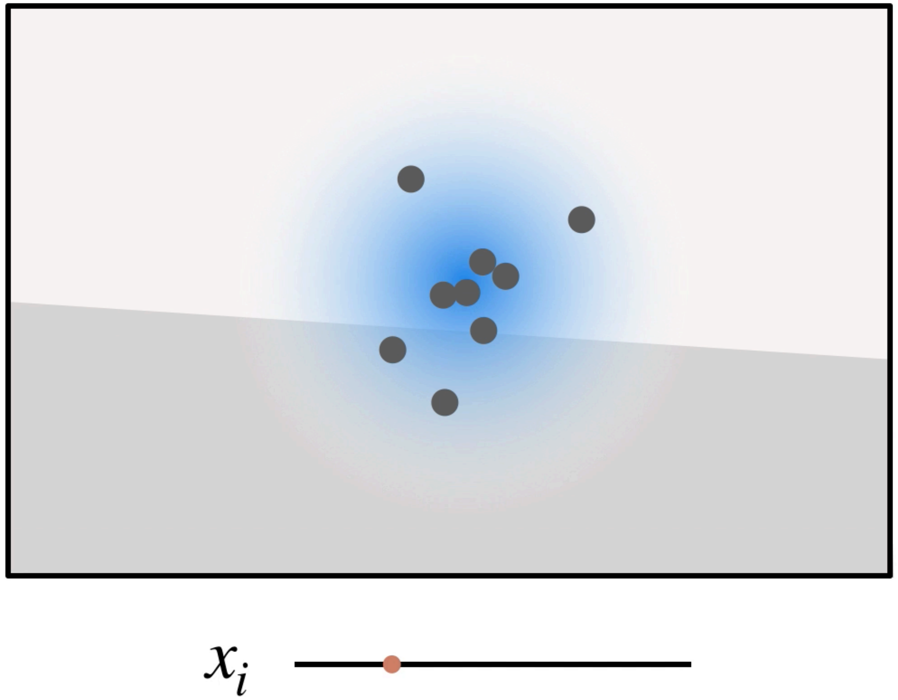
- Same value of the integral
- Same sample positions
- Different partial derivatives for MC samples

Key Idea: Re-parameterizing Integrals

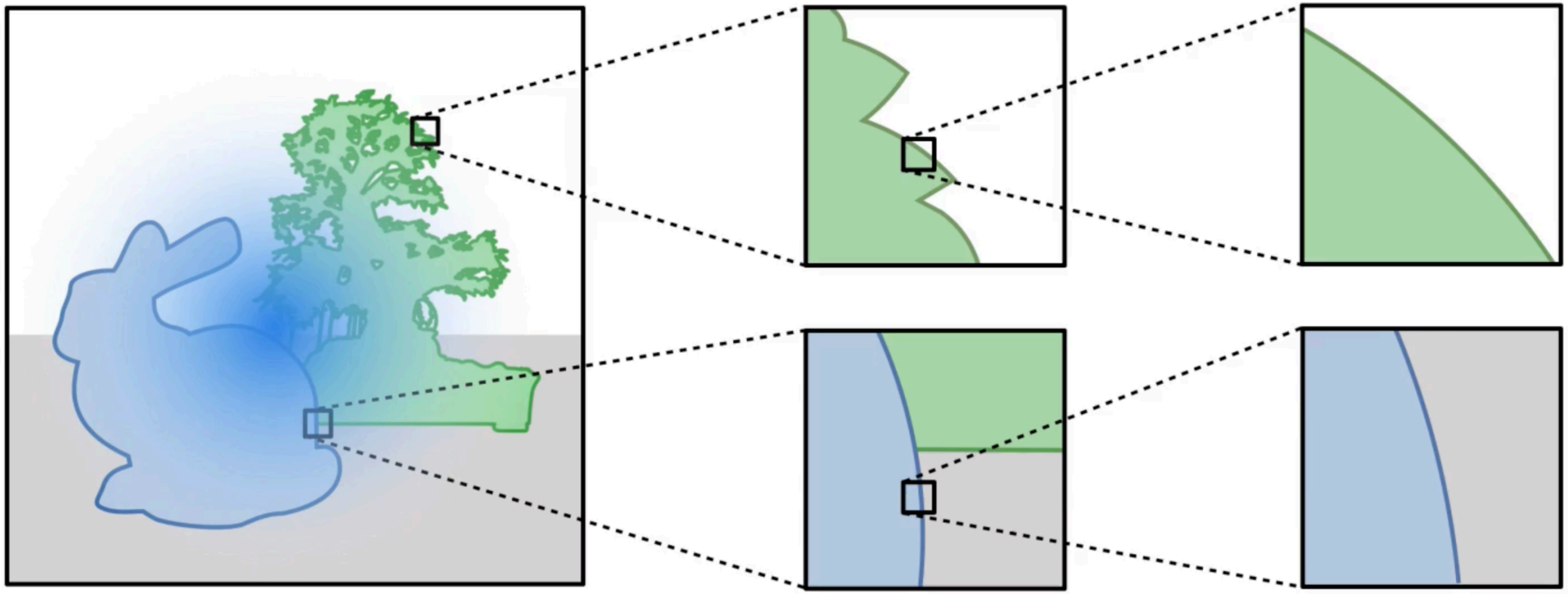
Non-differentiable Monte Carlo estimates



Differentiable Monte Carlo estimates



Integrals with Large Support

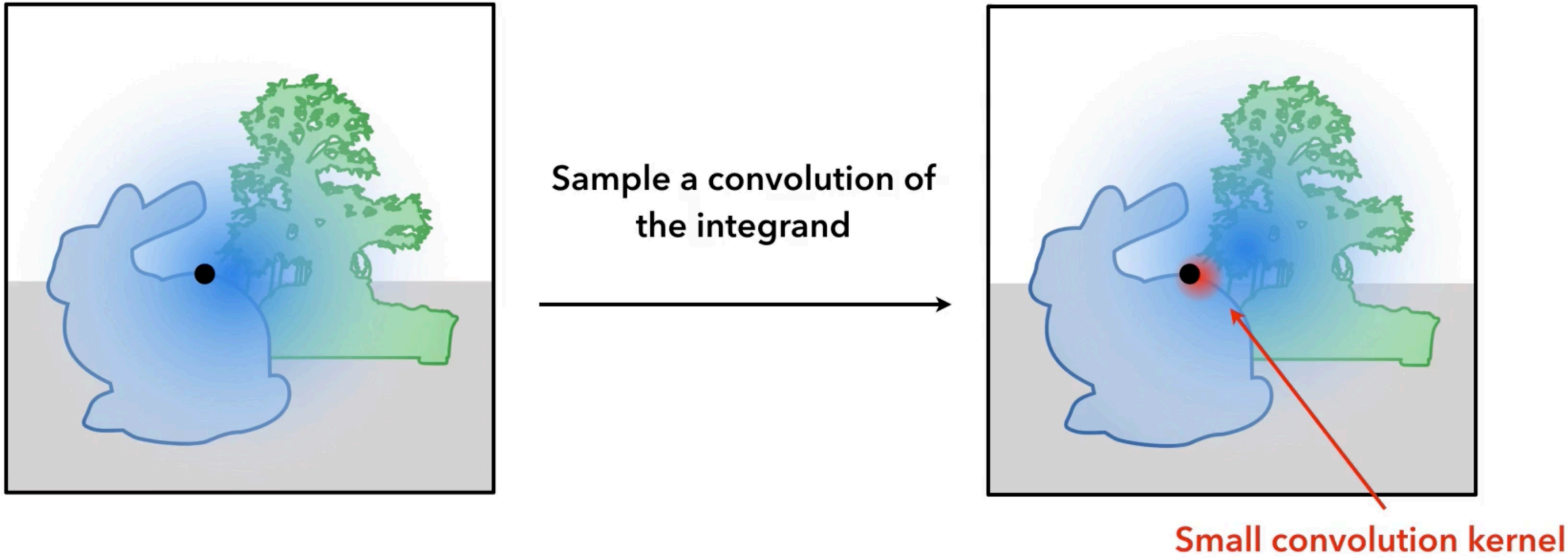


No useful reparameterization

Simple changes of variables make estimates differentiable

(assumption: infinitesimal translation)

Integrals with Large Support



- **Estimating the same integral with a different sampling technique**

Integrals with Large Support

Assumption (Small angular support):

Removing discontinuities using rotations

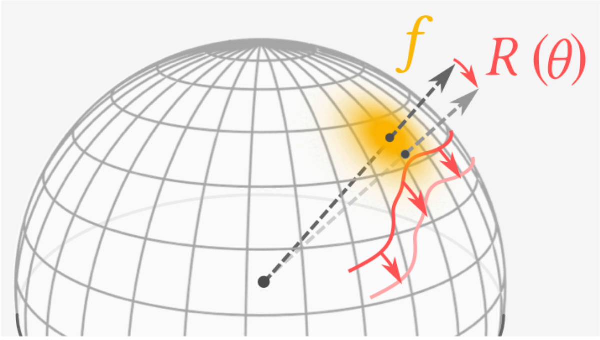
$$I = \int_{S^2} f(\omega, \theta)\, \mathrm{d}\omega = \int_{S^2} f(R(\omega, \theta), \theta)\, \mathrm{d}\omega$$

$$E = \frac{1}{N} \sum \frac{f(R(\omega_i, \theta), \theta)}{p(\omega_i, \theta_0)} \approx I$$

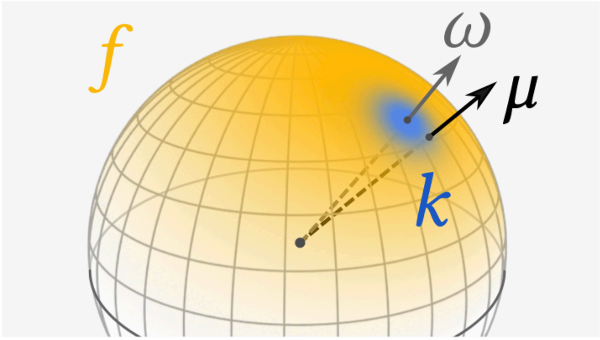
Handling with large support

$$\int_{S^2} f(\omega)\, \mathrm{d}\omega = \int_{S^2} \int_{S^2} f(\mu)\, k(\mu, \omega)\, \mathrm{d}\mu\, \mathrm{d}\omega, \quad \int_{S^2} k(\mu, \omega)\, \mathrm{d}\mu = 1. \quad \forall \omega \in S^2$$

$$I \approx E = \frac{1}{N} \sum \frac{f(R_i(\mu_i, \theta), \theta)\, k(R_i(\mu_i, \theta), \omega_i(\theta), \theta)}{p(\omega_i(\theta), \theta)\, p_k(\mu_i)}$$

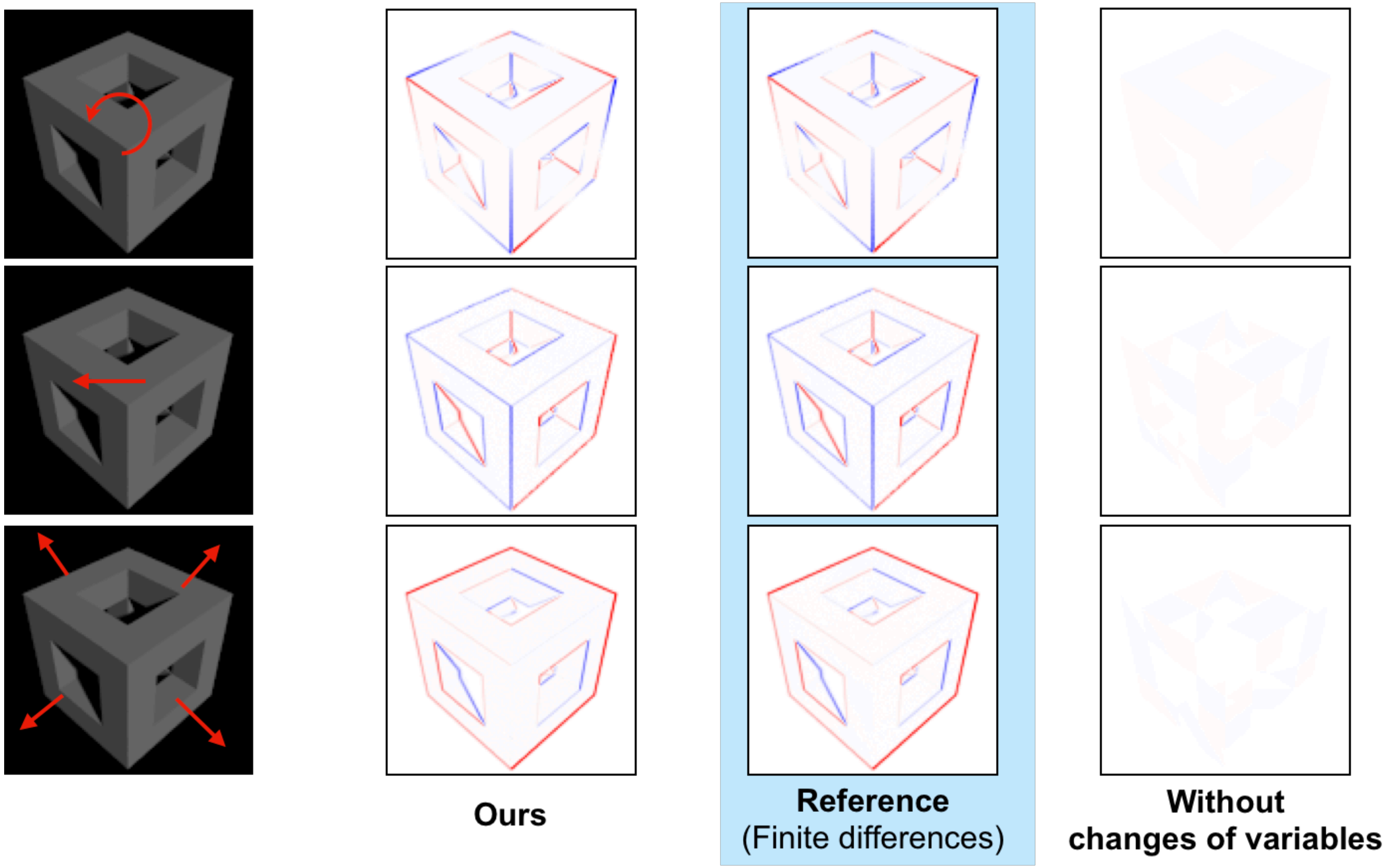


(a) Differentiable rotation of directions

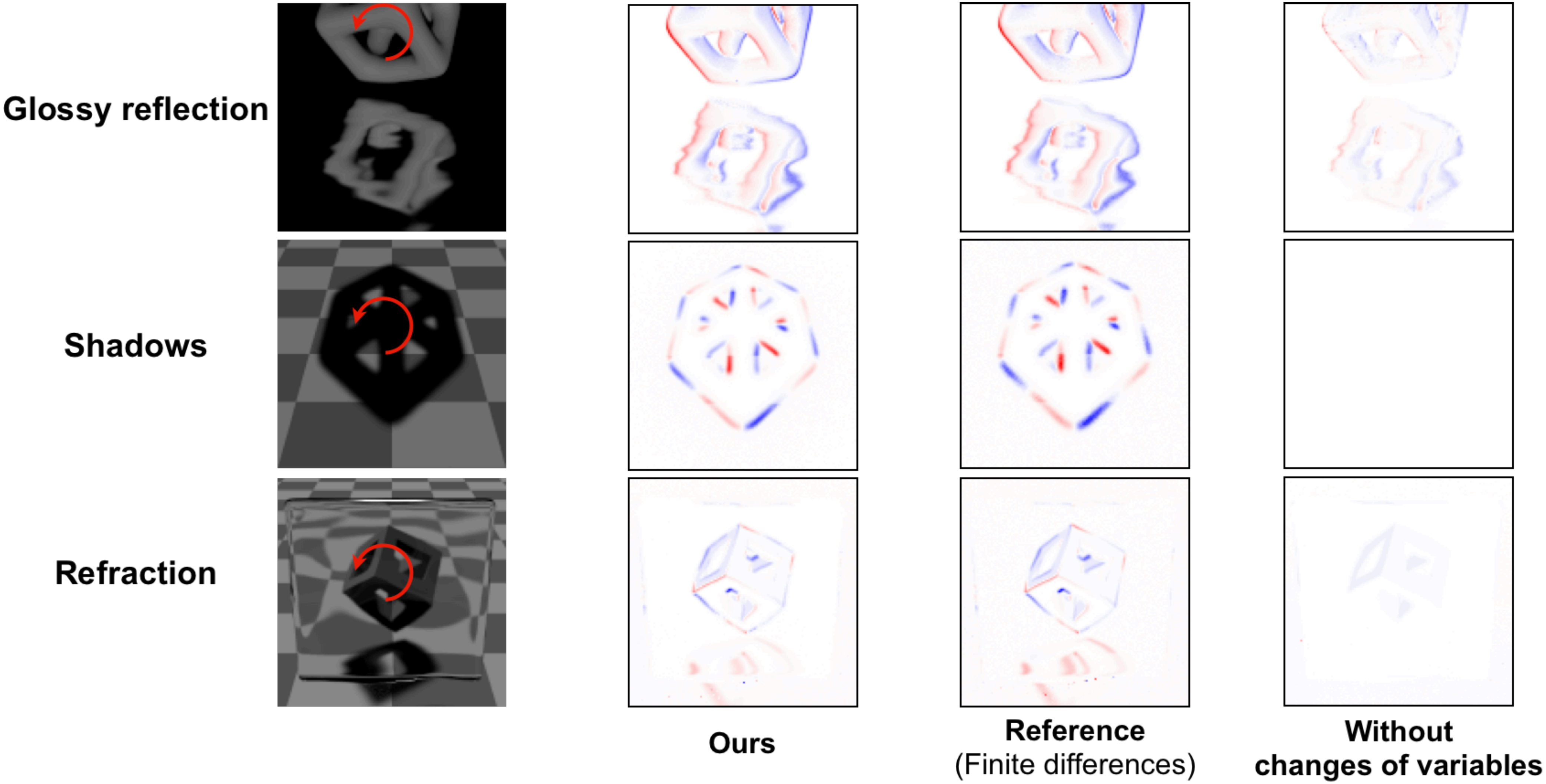


(b) Notations for our spherical convolutions

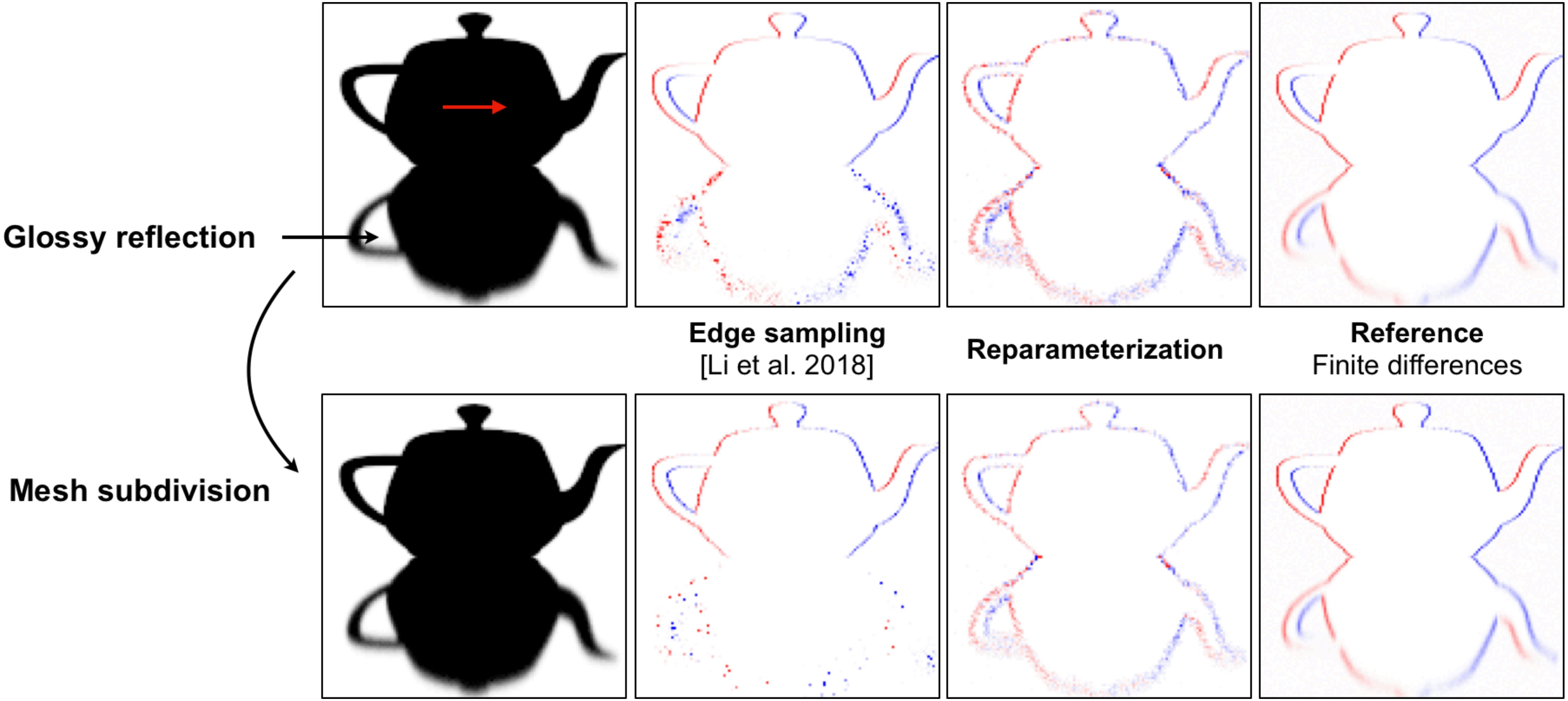
Results


Ours
Reference
(Finite differences)
Without
changes of variables

Results

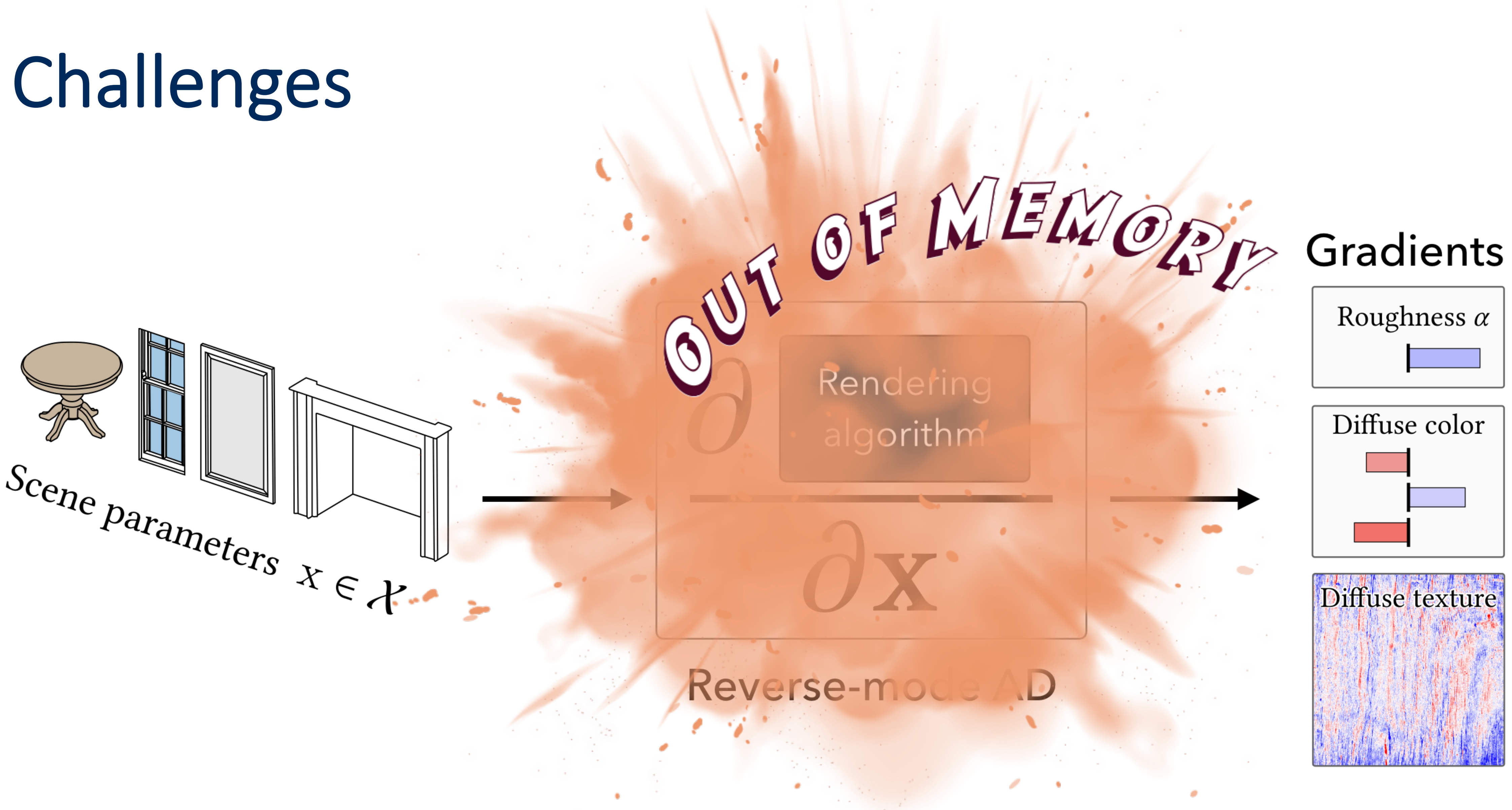

Glossy reflection
Shadows
Refraction
Ours
Reference
(Finite differences)
Without
changes of variables

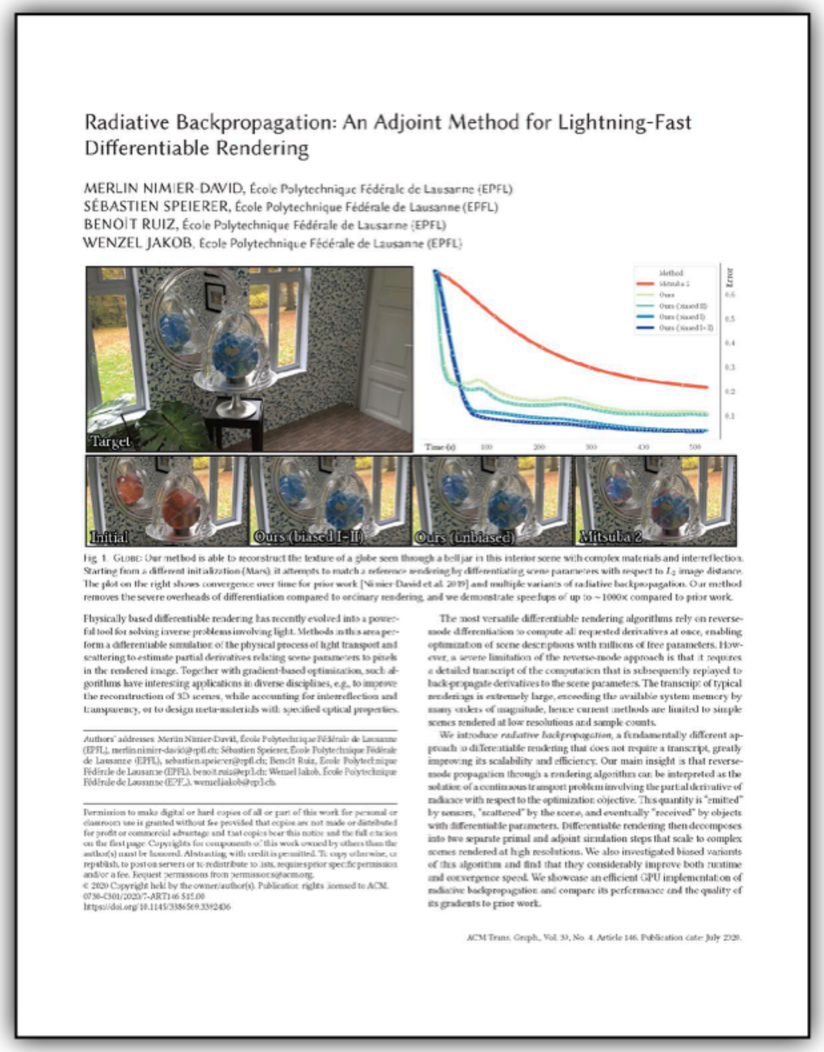
Results



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

Challenges


Scene parameters $x \in \mathcal{X}$
OUT OF MEMORY
Rendering algorithm
$\frac{\partial}{\partial x}$
Reverse-mode AD
Gradients
Roughness α
Diffuse color
Diffuse texture



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

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

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

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

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

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

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

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

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

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

SIGGRAPH 2020

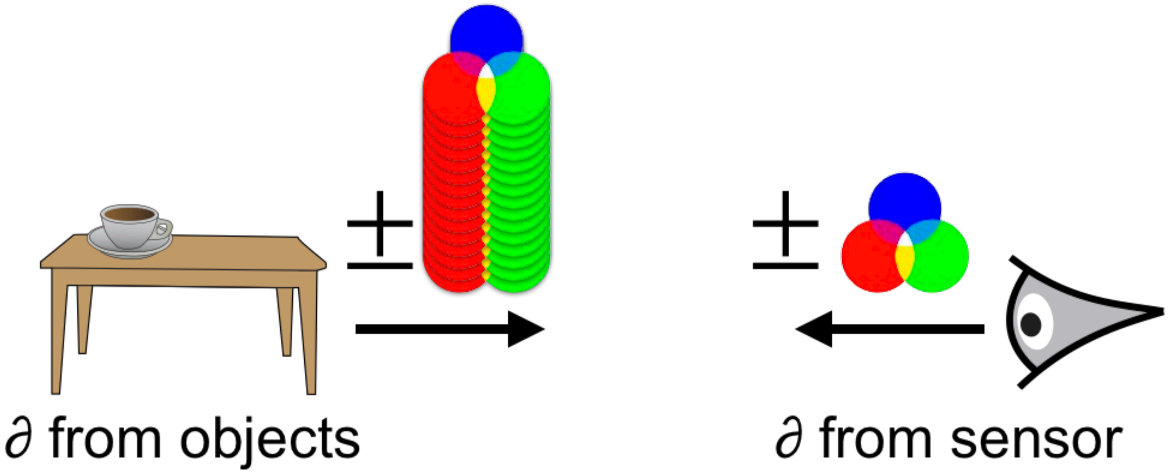
Radiative Backpropagation

Normal rendering

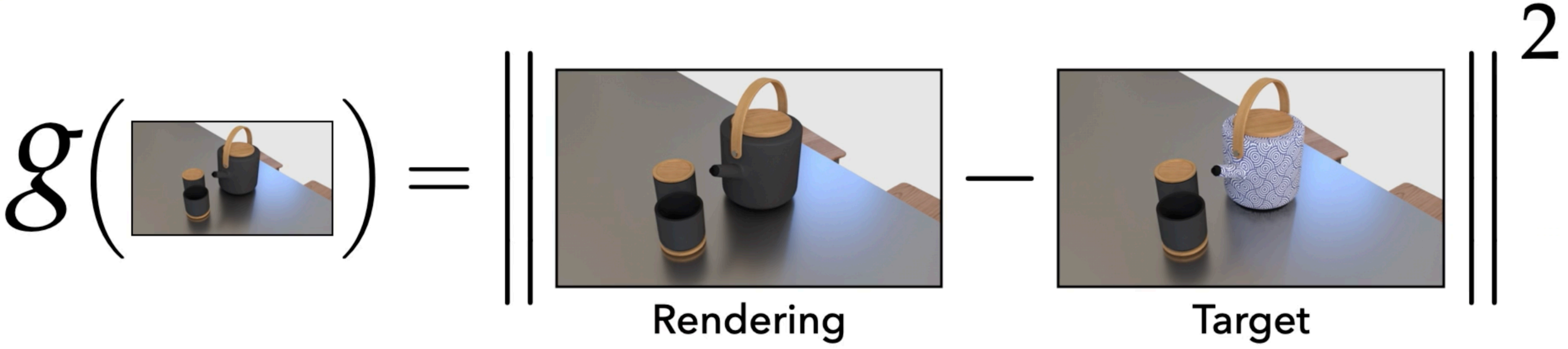
- Transporting from sensor/light may yield lower variance.

Differentiable rendering

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



Render and Compare



Scene parameters

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

Scene parameters

$$z = g(f(\mathbf{x}))$$

Objective

Rendering algorithm

We need: $$\frac{\partial z}{\partial \mathbf{x}}$$

Render and Compare

Objective function

Rendering

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

Scene parameters

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

Sensitivity gradients

Chain Rule

Objective function

Radiative Backpropagation

$$\frac{\partial z}{\partial \mathbf{x}} = \frac{\partial z}{\partial \mathbf{y}} \cdot (\cdots) \cdot \frac{\partial f_s(\mathbf{x}, \omega, \omega')}{\partial \mathbf{x}}$$

Scene parameters

Chain Rule

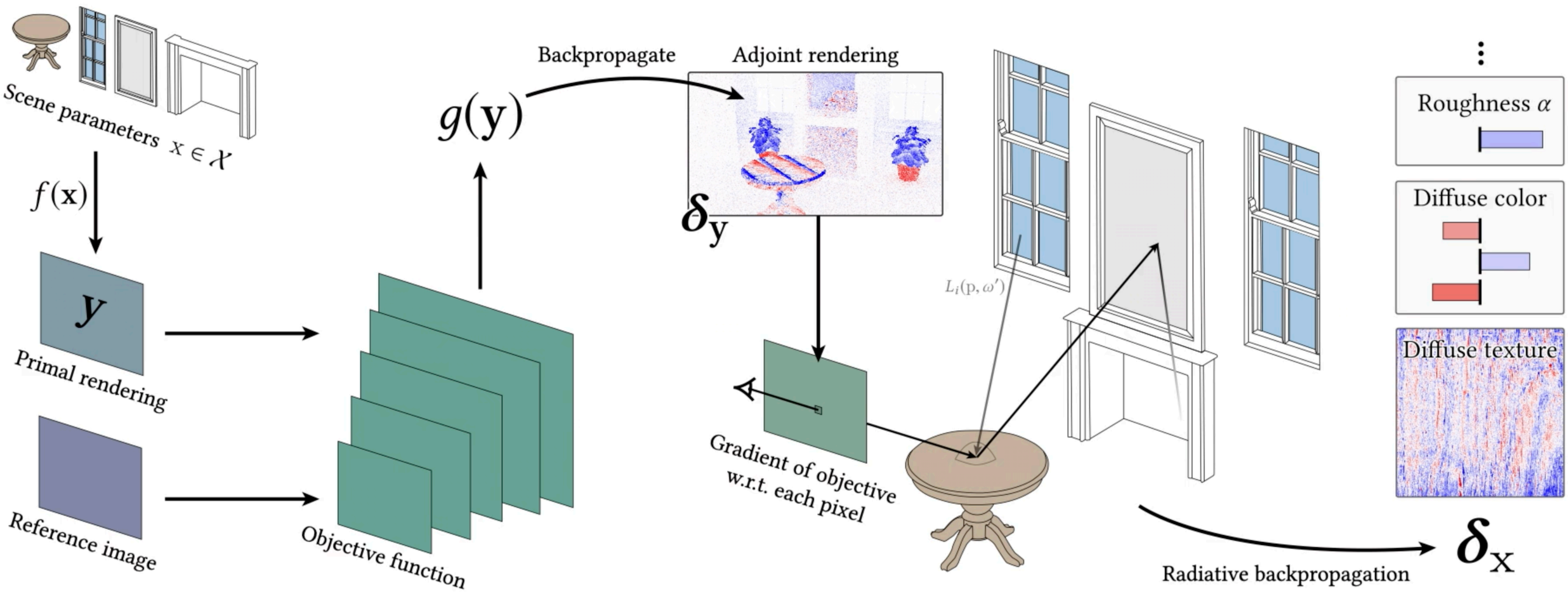
Radiative Backpropagation

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

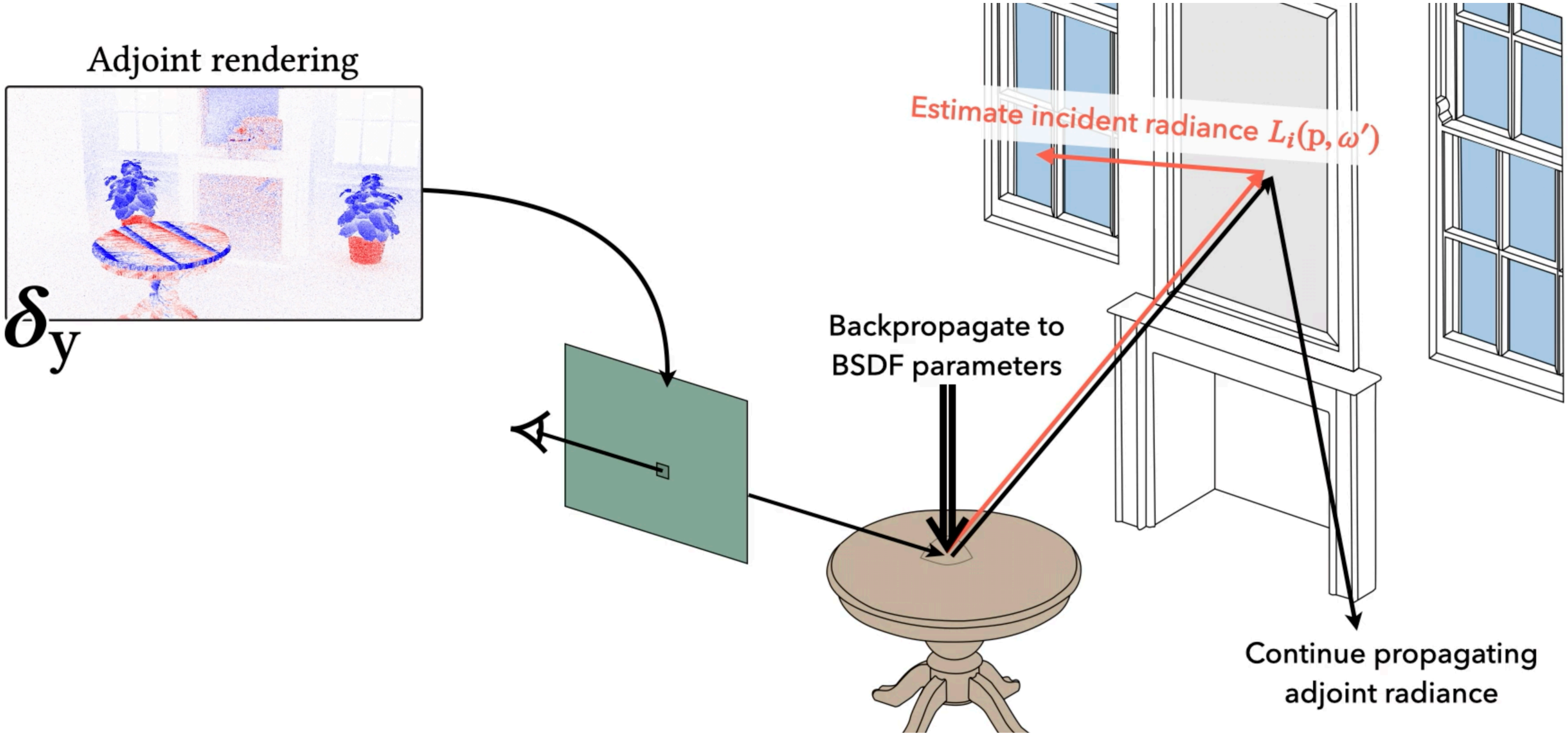
$$\cdot\,(\cdots)\,\cdot\,\frac{\partial f_s(\mathbf{x}, \omega, \omega')}{\partial \mathbf{x}}$$

"Derivative shader"
Easy & self-contained

Pipeline Overview

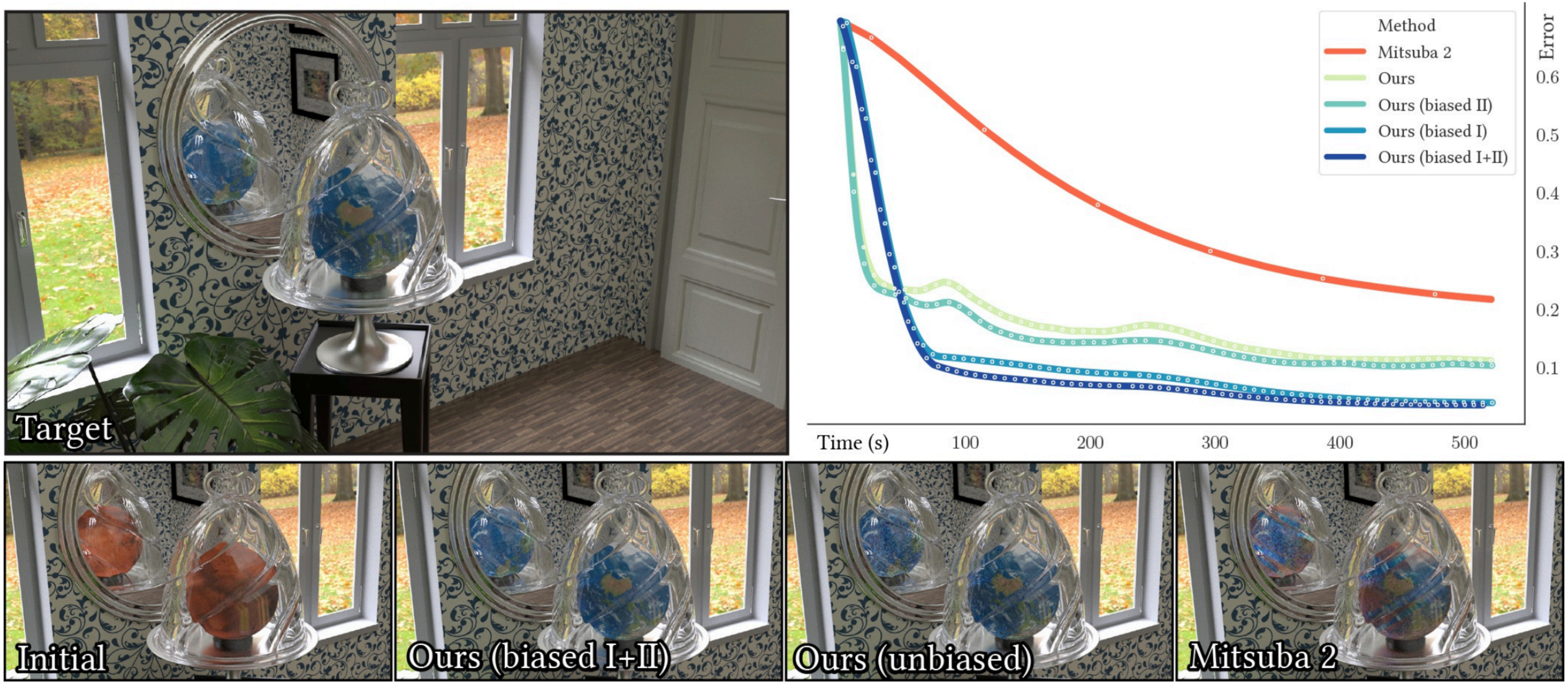


Radiative Backpropagation

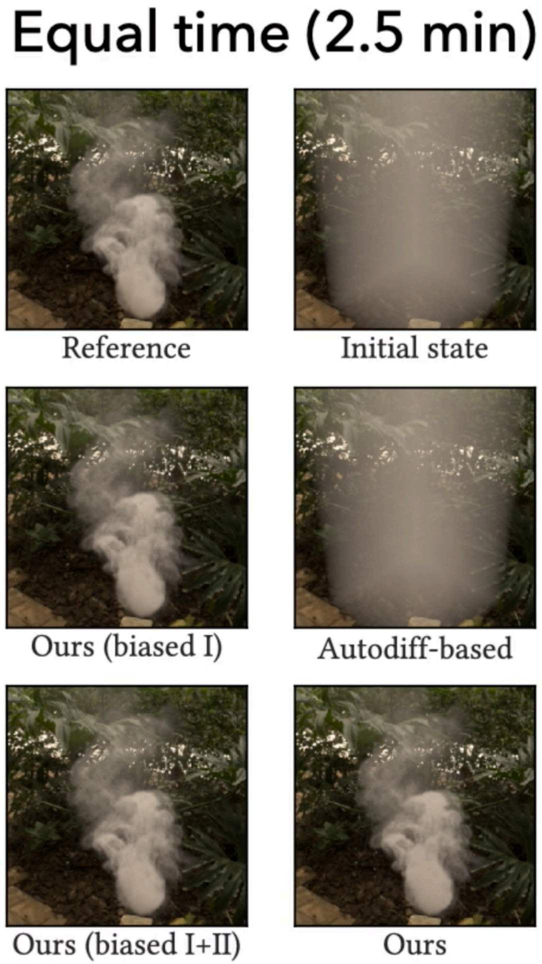


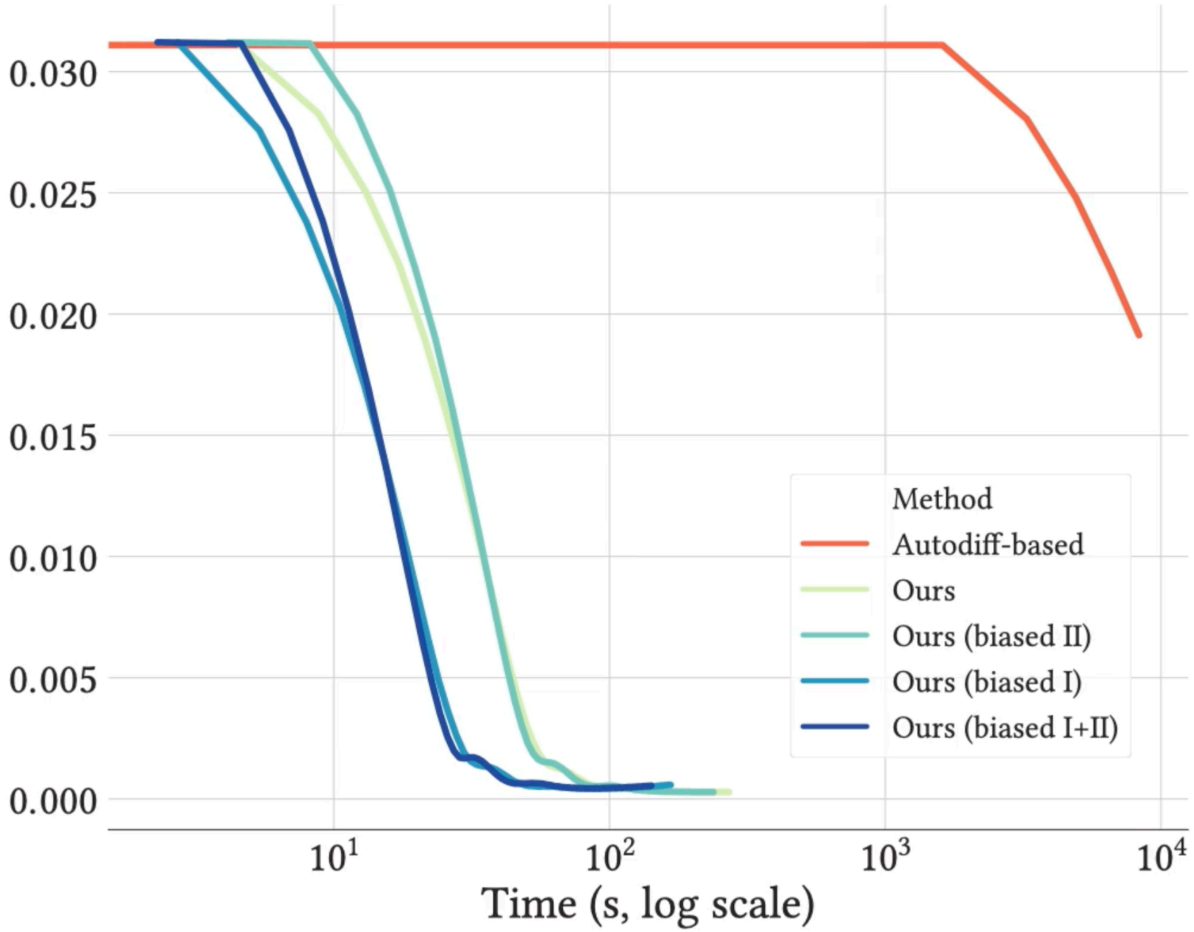
Surface Texture Optimization

Surface Texture Optimization

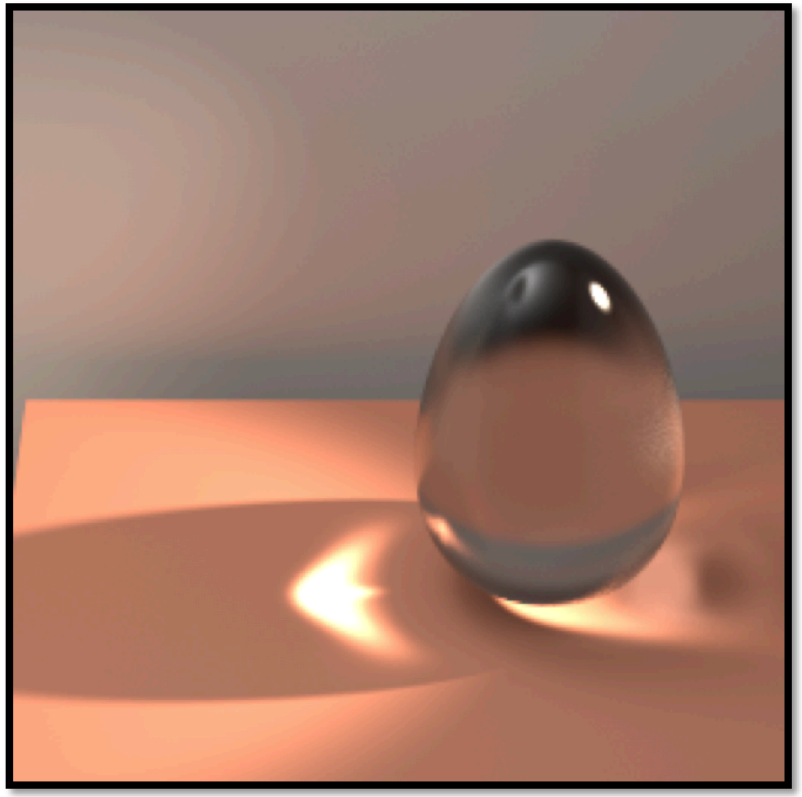


Volume Density Optimization





Challenges Remain



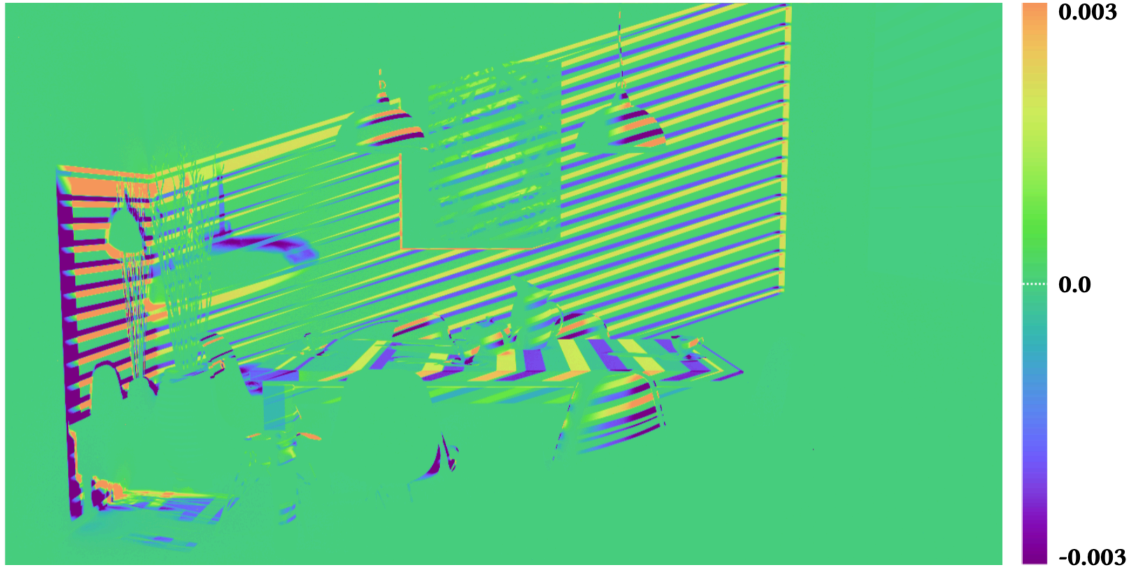
Complex light transport

Complex geometry & motion

Follow-up works

- Estimate the derivatives of the path integral formulation

Original



Derivative with respect to sun location

Path space differentiable rendering (Zhang et al., 2020)

Path Integral for Forward Rendering

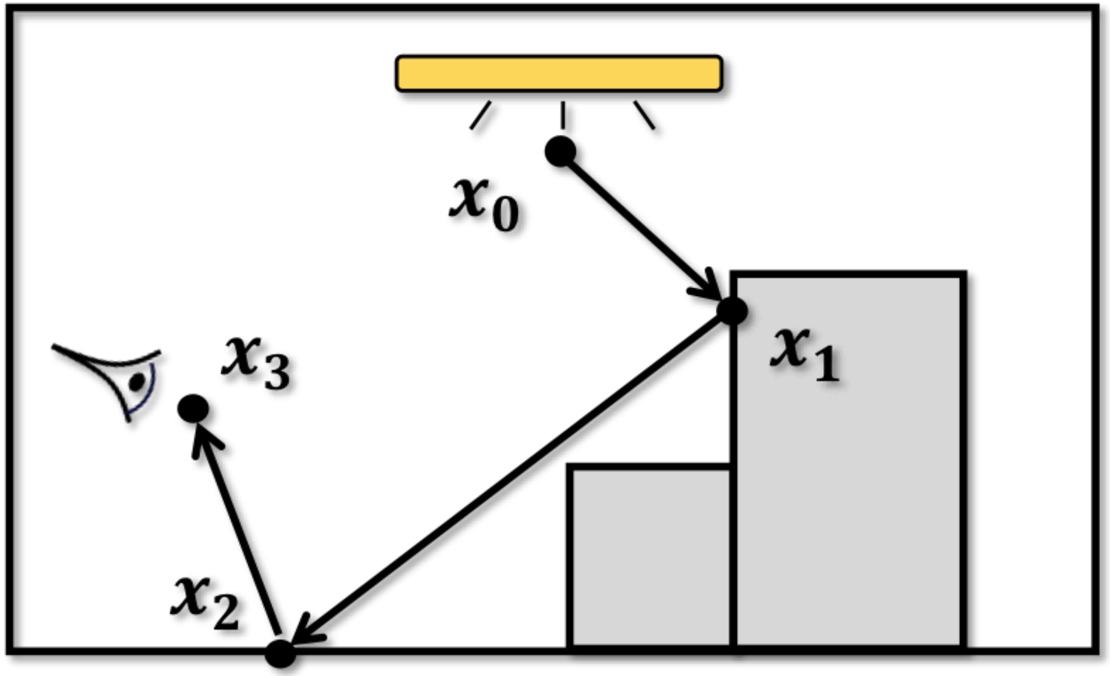
Measurement contribution function

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

Path space

Area-product measure

- Introduced by Veach [1997]
- Foundation of sophisticated Monte Carlo algorithms (e.g., BDPT, MCMC rendering)



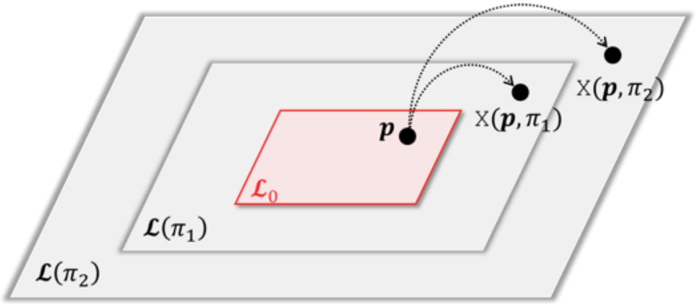
Light path $\bar{\boldsymbol{x}} = (x_0, x_1, x_2, x_3)$

- **Differential path integral**
  - Separated interior and boundary components

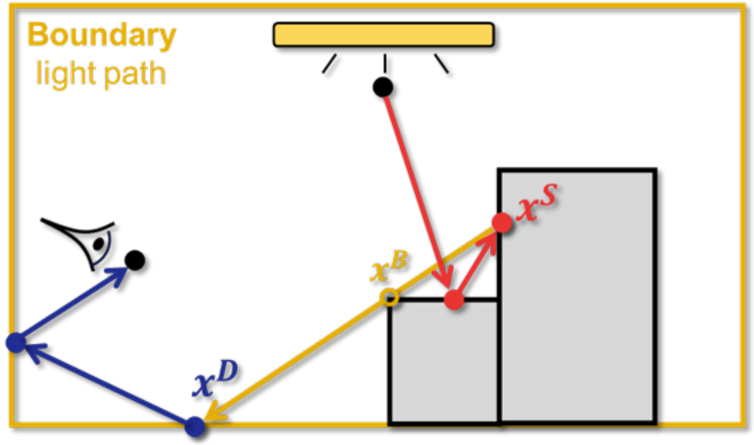
$$\frac{\mathrm{d}}{\mathrm{d}\pi}\int_{\Omega} f\,\mathrm{d}\mu = \int_{\Omega}\frac{\mathrm{d}f}{\mathrm{d}\pi}\,\mathrm{d}\mu + \int_{\partial\Omega} g\,\mathrm{d}\mu'$$

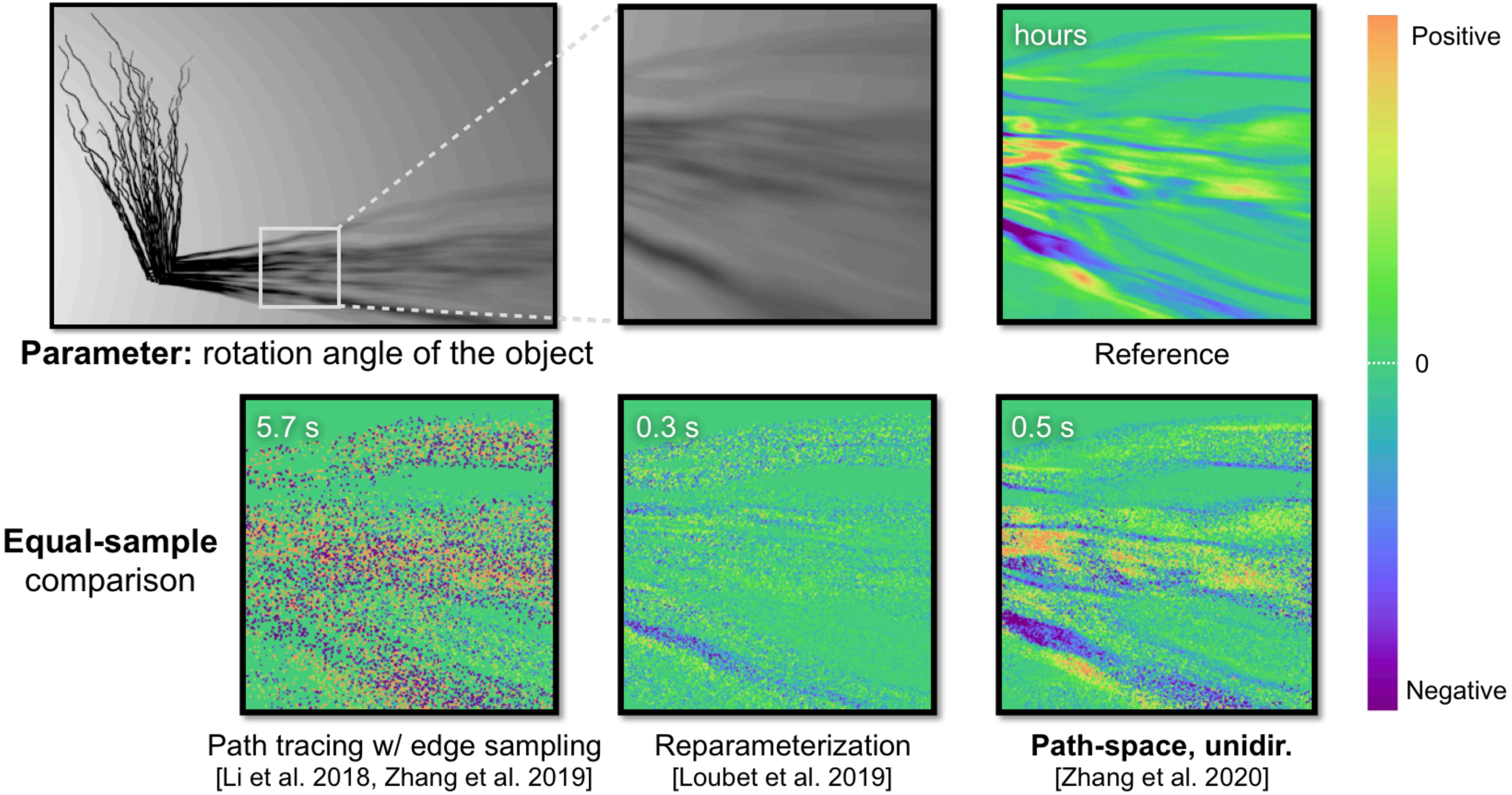
Interior　Boundary

- **Reparameterization**
  - Only need to consider silhouette edges



- **Unbiased Monte Carlo methods**
  - Unidirectional and bidirectional algorithms
  - No silhouette detection is needed





Parameter: rotation angle of the object
hours
Reference
Positive
0
Negative
Equal-sample comparison
5.7 s
0.3 s
0.5 s
Path tracing w/ edge sampling
[Li et al. 2018, Zhang et al. 2019]
Reparameterization
[Loubet et al. 2019]
Path-space, unidir.
[Zhang et al. 2020]

Summary

- **Differentiable rendering is challenging**
  - Discontinuities are everywhere
  - Automatic-differentiation is time & space consuming

- **Physics-based differentiable rendering**
  - Dealing with the discontinuities:
    - Edge sampling (Li et al. 2018, Zhang et al. 2019)
    - Reparameterization (Loubet et al. 2019)
    - Path integral formulation (Zhang et al. 2020)

  - Dealing with memory issue:
    - Radiative Backprop (Nimier-David et al. 2020)

References

- Differentiable Monte Carlo Ray Tracing through Edge Sampling. Li et al., 2018.
- Slides for “Differentiable Monte Carlo Ray Tracing through Edge Sampling”. Li et al., 2018.
- Differentiable Ray Tracing. Novello. https://sites.google.com/site/tiagonovellodebrito/diffrt.
- Differentiable Rendering: A Survey. Kato et al., 2020.
- Ray Tracing Essential Part 6: The Rendering Equation. https://news.developer.nvidia.com/ray-tracing-essentials-part-6-the-rendering-equation/
- https://rgl.epfl.ch/publications/Loubet2019Reparameterizing
- https://shuangz.com/projects/psdr-sg20/
- https://shuangz.com/courses/pbdr-course-sg20/