CSC2457 3D & Geometric Deep Learning

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: 16 Feb. 2021

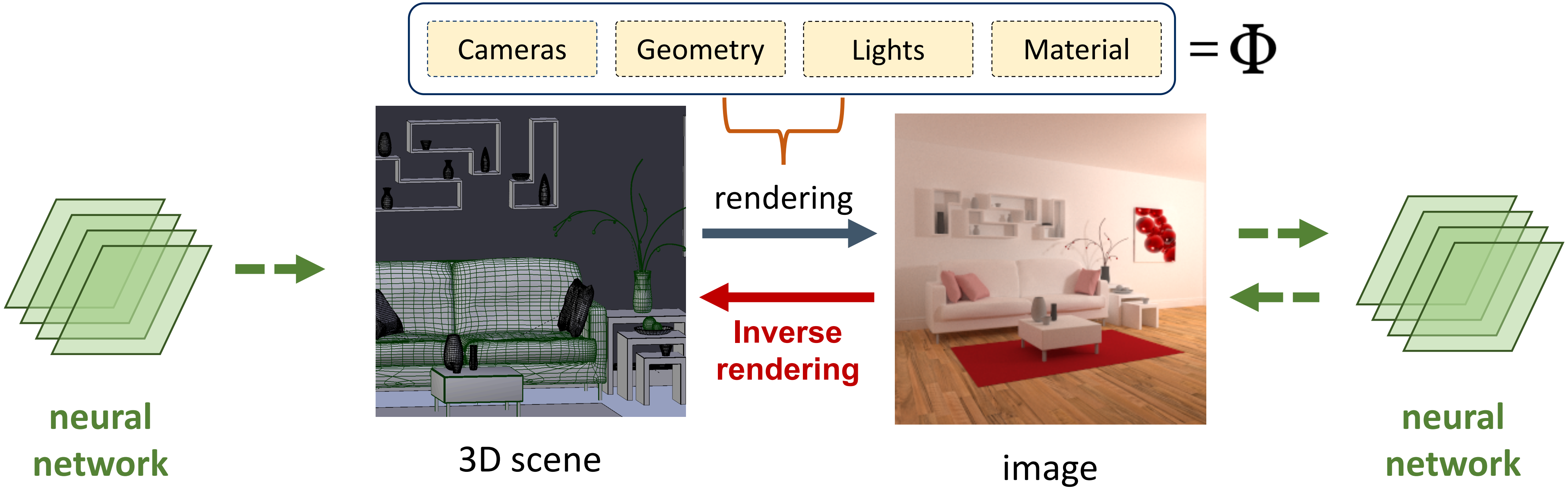
Presenter: Jingkang Wang

Instructor: Animesh Garg

UNIVERSITY OF TORONTO

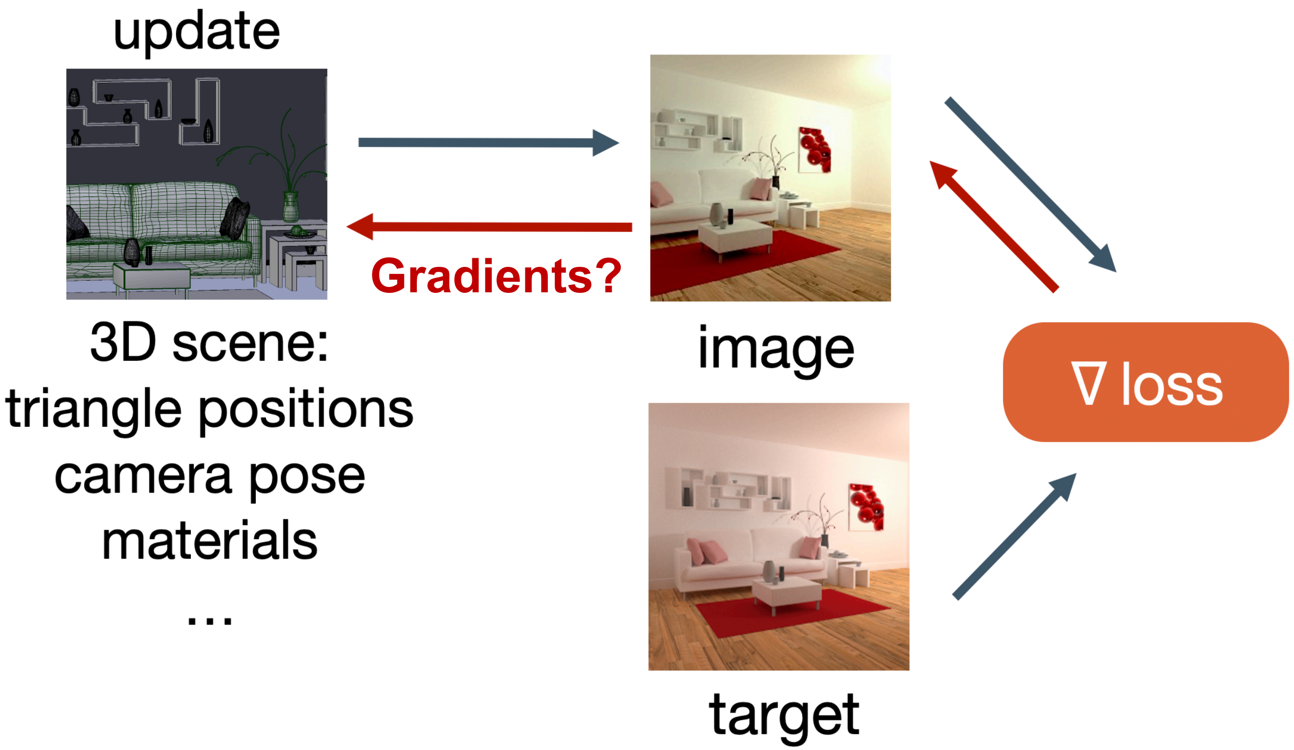
Differentiable Rendering is Important!

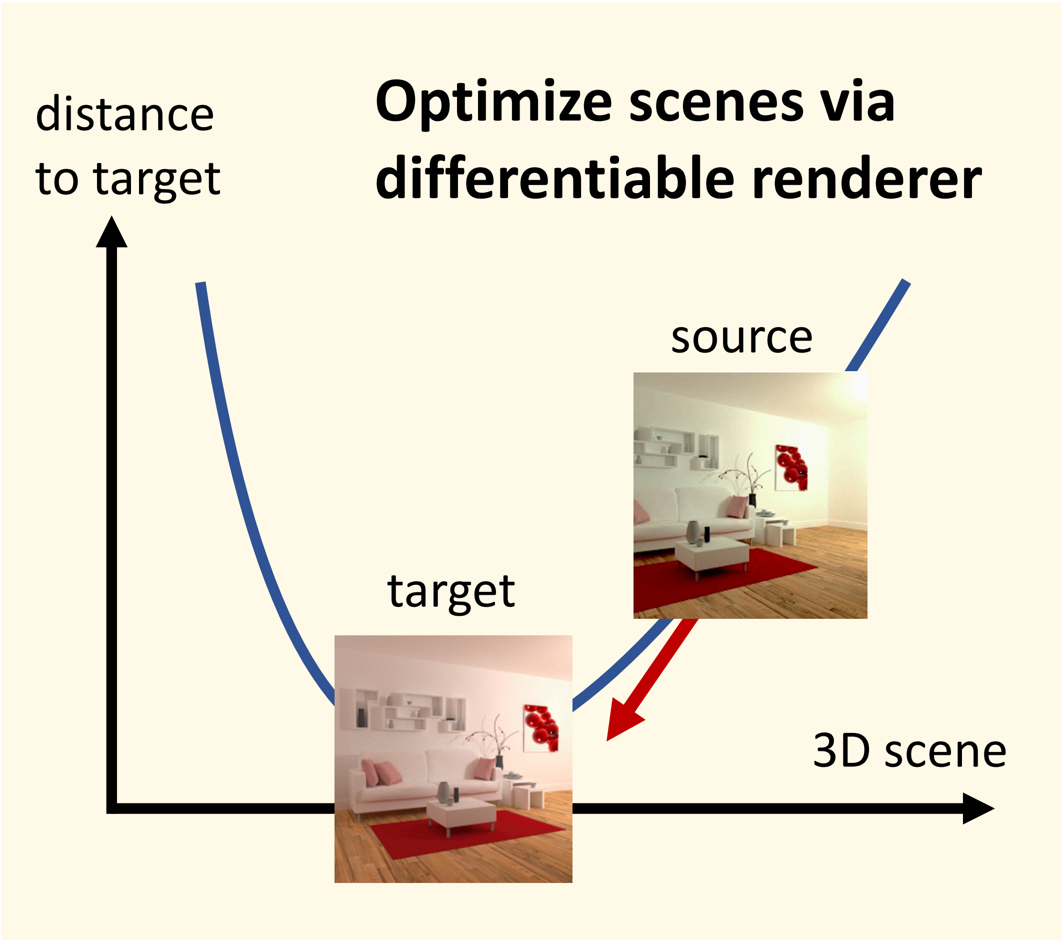
- The ability of calculating gradients are crucial to optimization
  - (a) inverse problems, (b) deep learning



Differentiable Rendering is Important!

- Render and compare approach





Differentiable Rendering is Important!

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

rendered image

Rendering integral includes visibility terms that are not differentiable

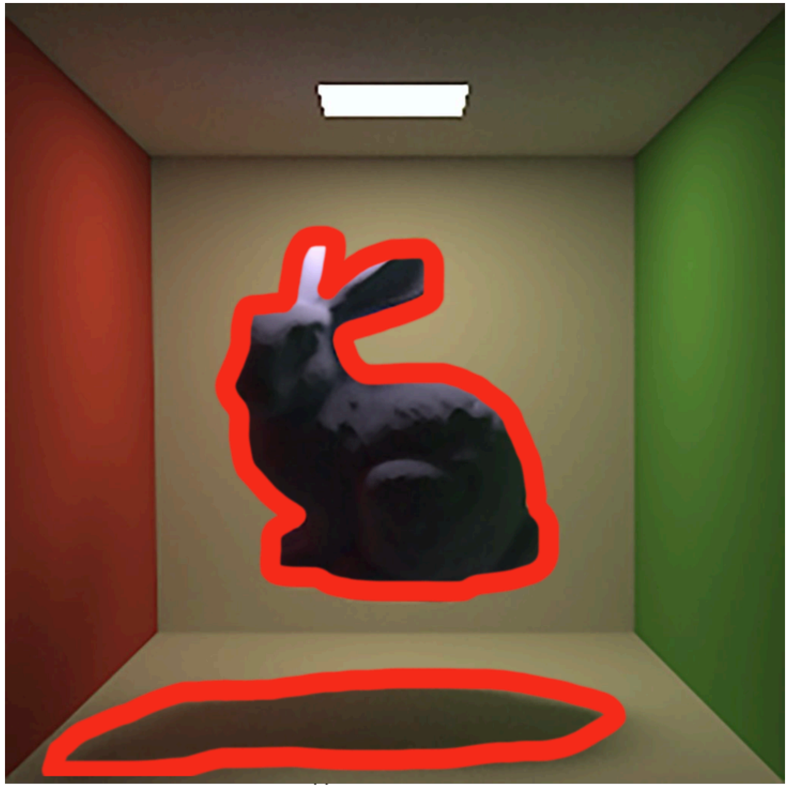
$$I = \iint \underbrace{k(x,y)}_{\text{Pixel filter}} \underbrace{L(x,y)}_{\text{Radiance (another integral)}} dxdy$$

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

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

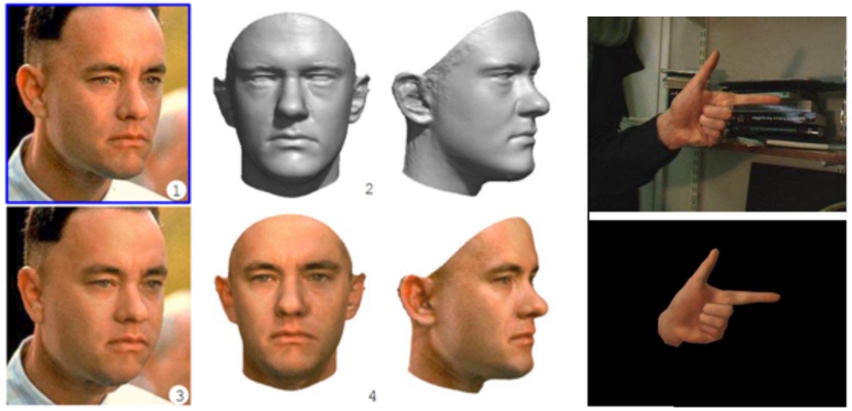
Differentiable Rendering is Challenging!

- **Challenge**: both primary and secondary visibility matter

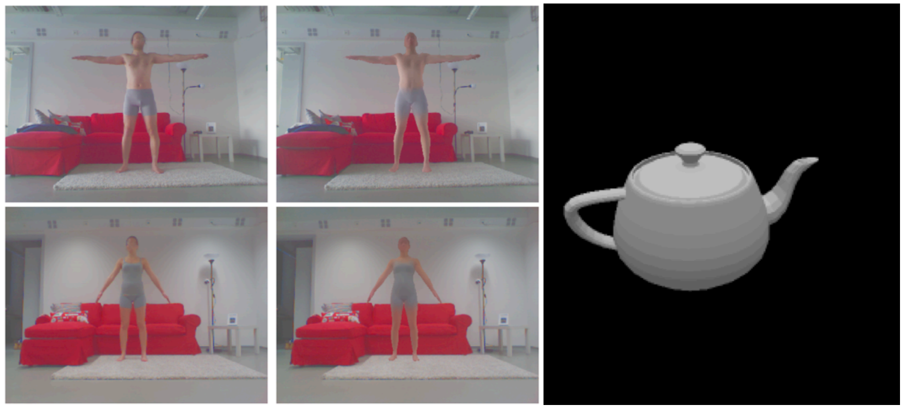


Contributions

- 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



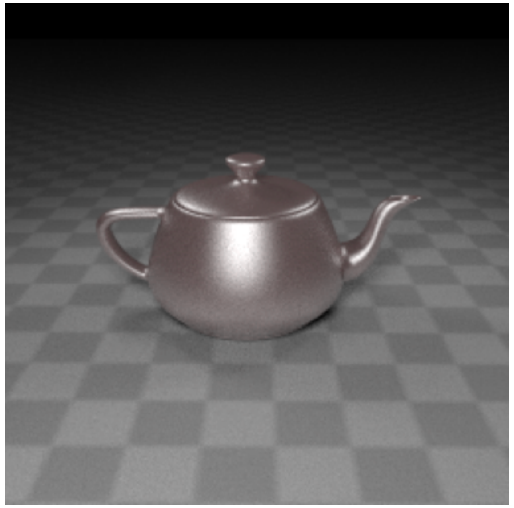
Specific cases
(Blanz et al. 1999, Gorce et al. 2008,
Gkioulekas et al., 2013)



Limited general renders
(Loper and Black 2014, Kato et al. 2018)

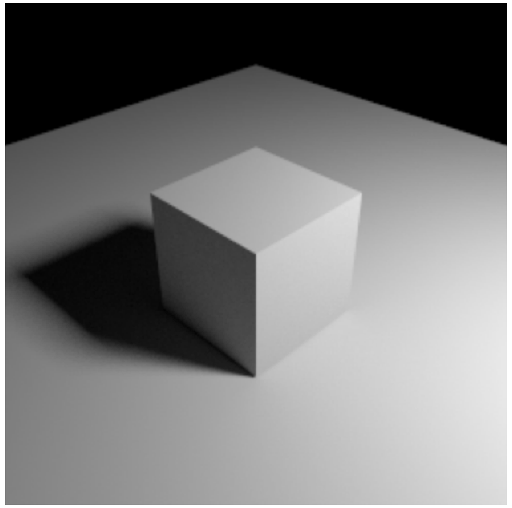
Contributions

- This paper proposed a general physically-based differentiable render



glossy reflection

mirror reflection



shadow

global illumination

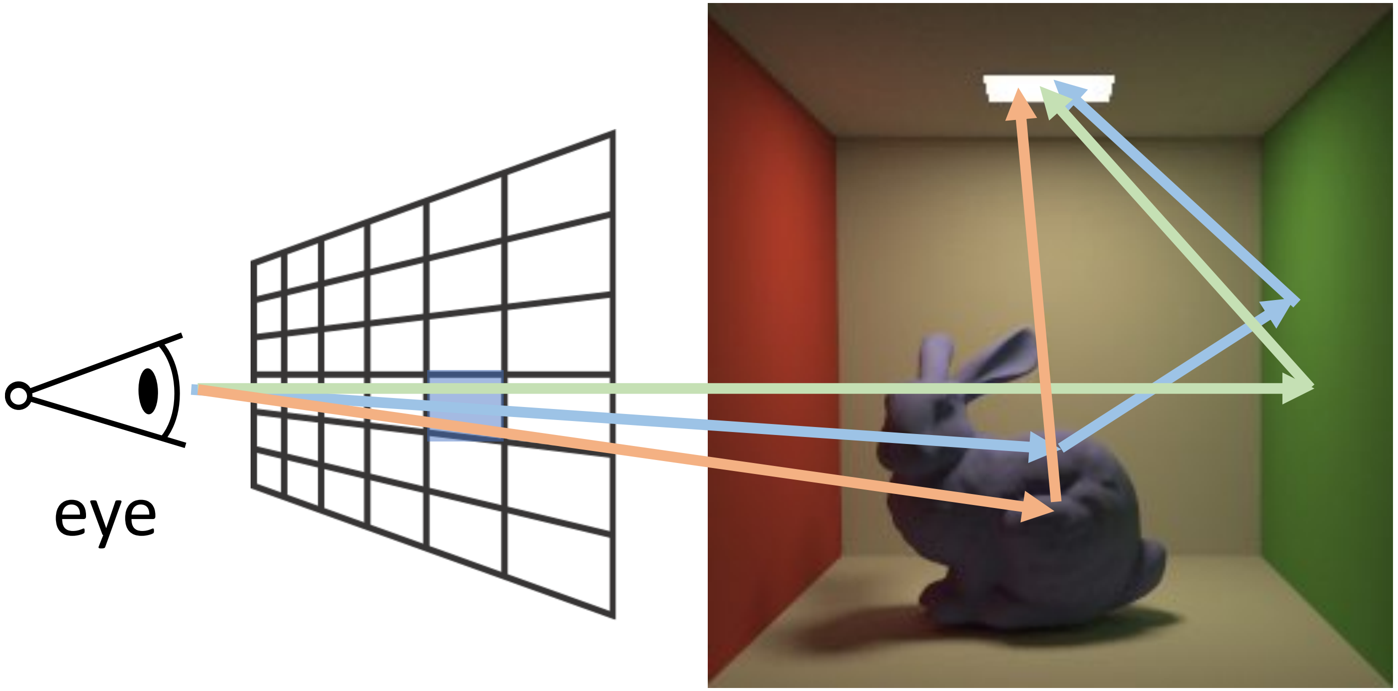
Contributions

- 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

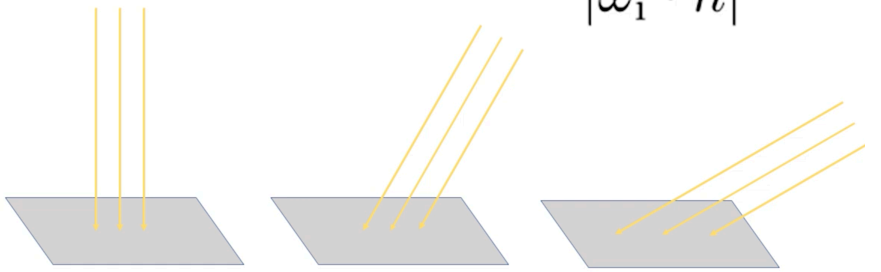
Physically-based Rendering

- The Rendering Equation



The Rendering Equation

$|\hat{\omega}_\mathrm{i} \cdot \hat{n}|$



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

<u>All</u> incoming directions
(a sphere)

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

The Rendering Equation

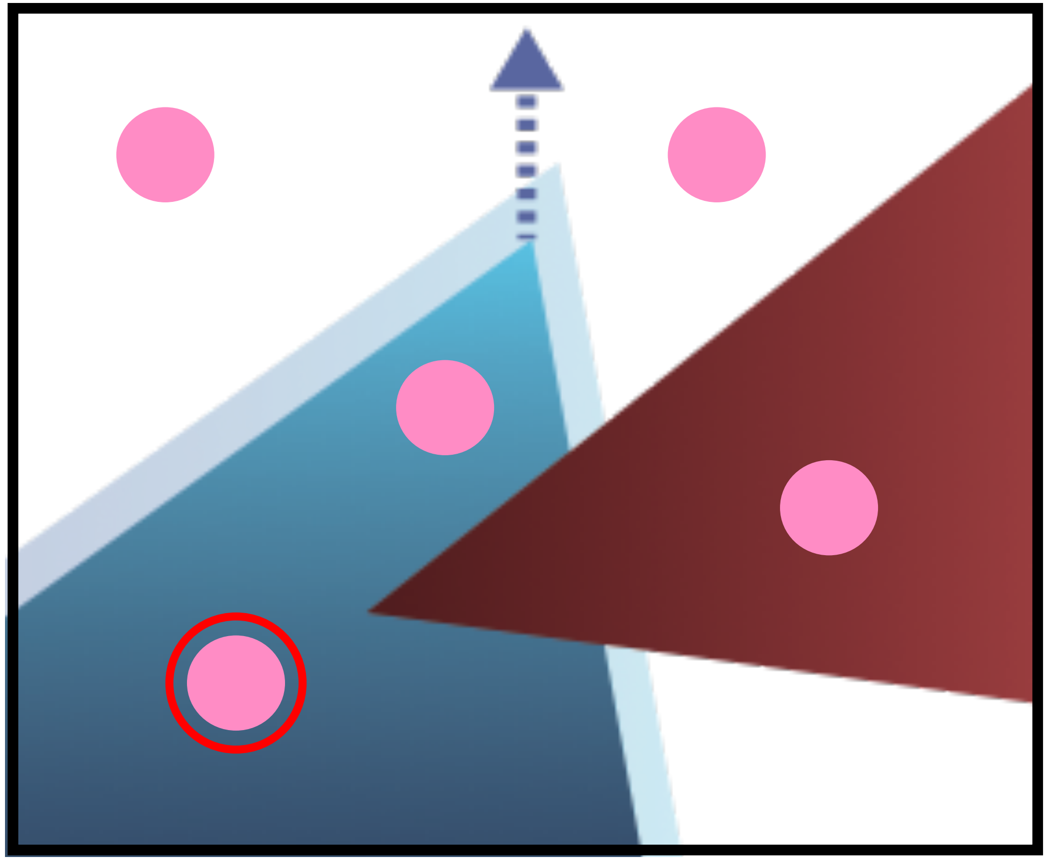
$$\boxed{L_o(X, \hat{\omega}_o)} = \boxed{L_e(X, \hat{\omega}_o)} + \int_{\mathbf{S}^2} \boxed{L_i(X, \hat{\omega}_i)} \; \boxed{f_X(\hat{\omega}_i, \hat{\omega}_o)} \; \boxed{|\hat{\omega}_i \cdot \hat{n}|} \; d\hat{\omega}_i$$

Outgoing light Emitted light Incoming light Material Lambert

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

Rendering = Sampling

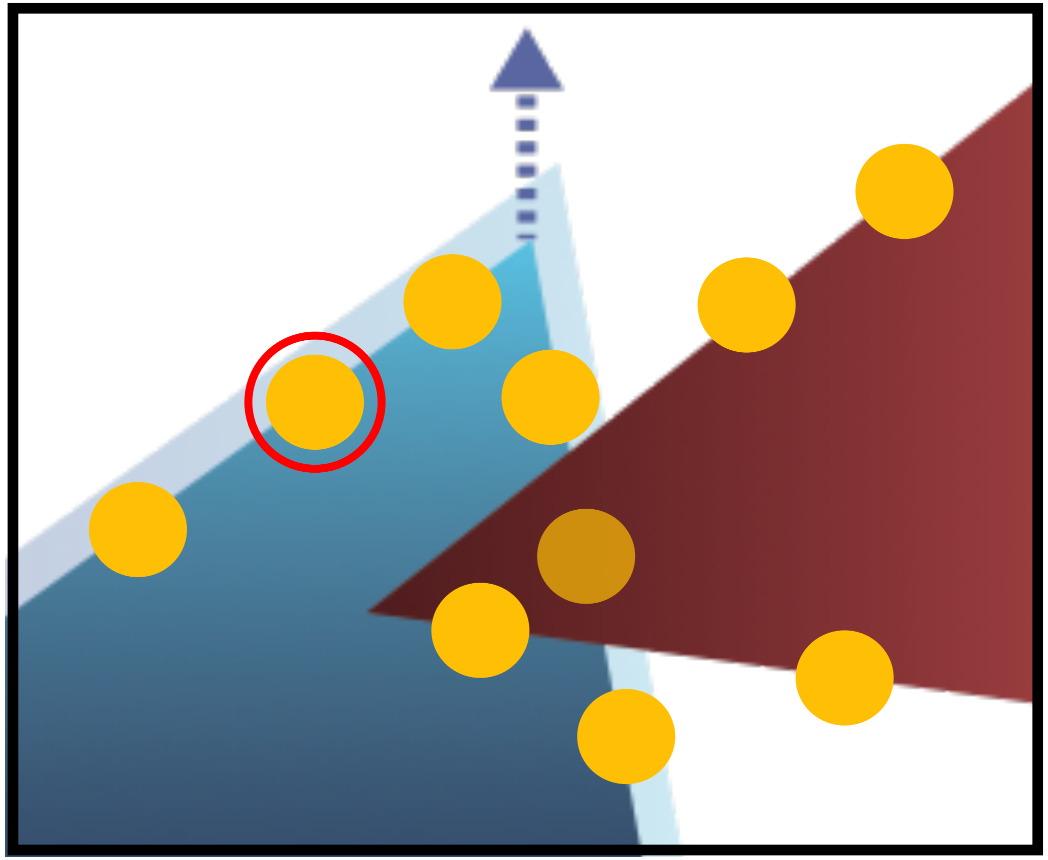
color change
when blue triangle
moves up?



pixel

Key idea: Edge sampling

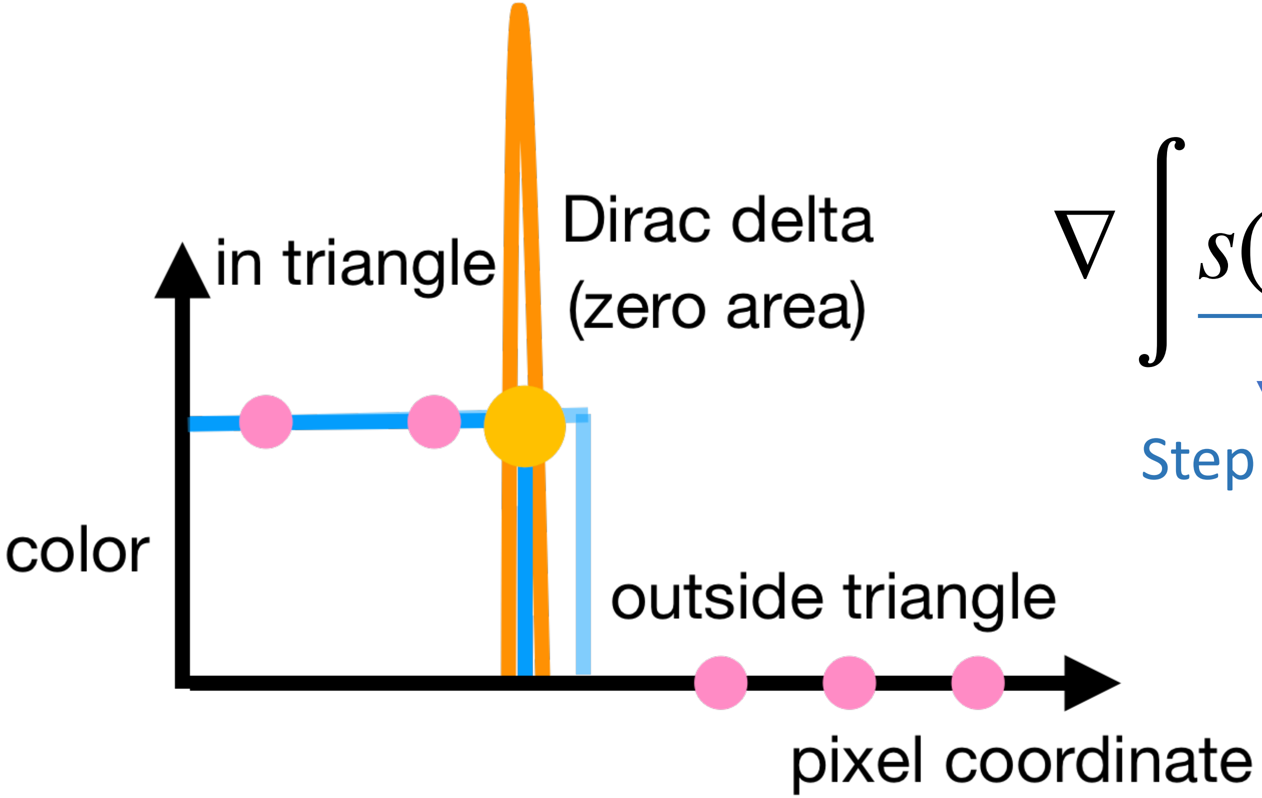
color change
when blue triangle
moves up?



pixel

Mathematical formulation

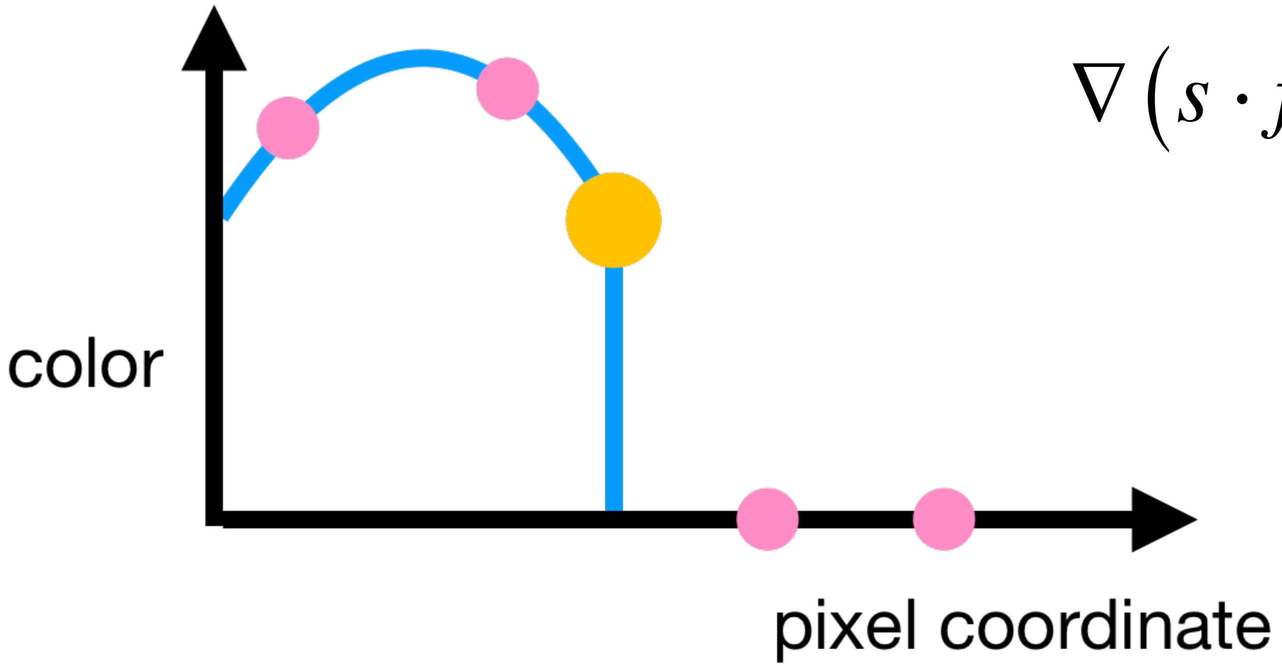
- Model each pixel is an integral over the step function
- Each pixel is an integral over the step functions



$$\nabla \int \underbrace{s(x)}_{\text{Step function}} dx = \int \nabla s(x) dx = \underbrace{\delta(x)}_{\text{Dirac delta}}$$

Mathematical formulation

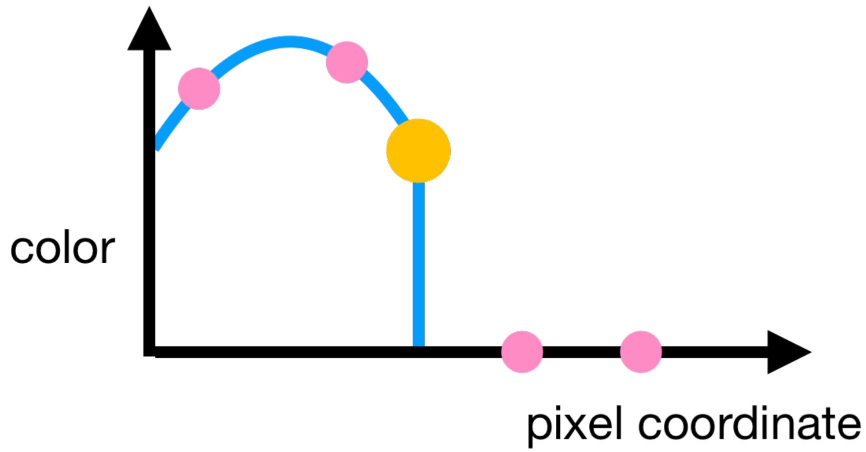
- A smooth shading function f multiples to the step function s



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

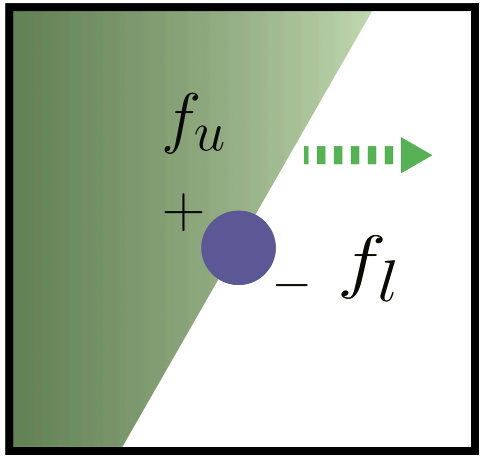
Mathematical formulation

- 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

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



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

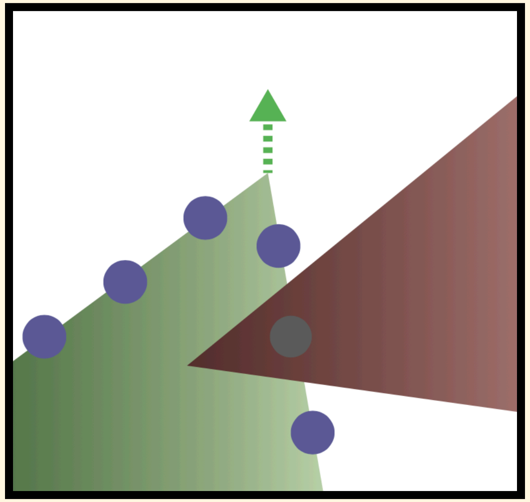
$$I = \iint f(x, y; \Phi) dxdy = \sum_i \iint \theta\left(\alpha_i(x, y)\right) f_i(x, y; \Phi) dxdy$$

Mathematical formulation

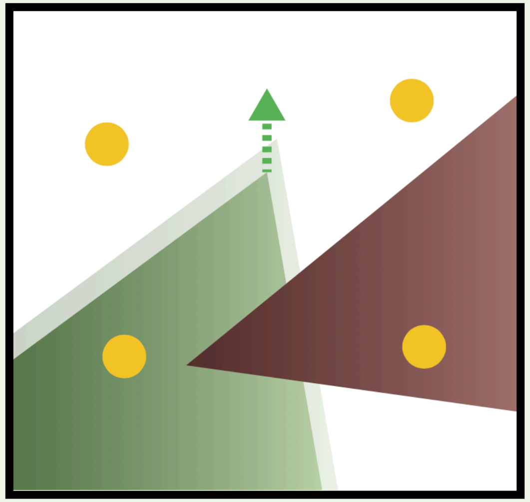
- Using the Chain rule

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

Edge sampling



Area sampling

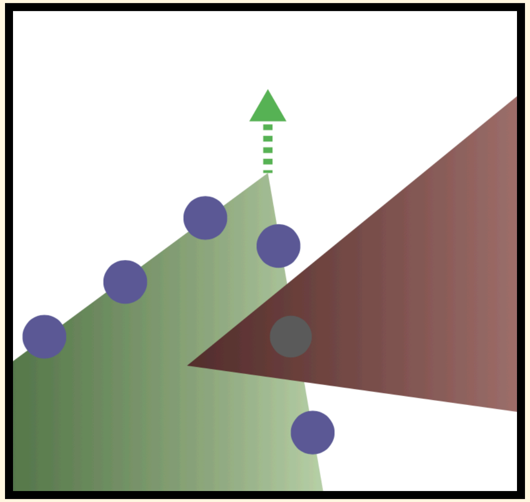


Mathematical formulation

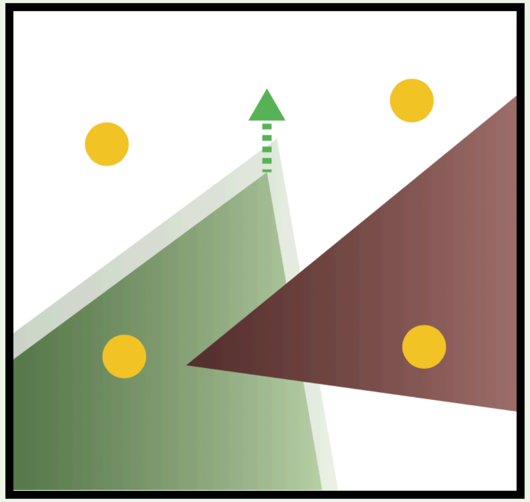
- Using the Chain rule

$$\nabla \iint \theta(\alpha(x,y)) f(x,y;\Phi)\,dxdy = \int_{\alpha_i(x,y)=0} \frac{\nabla \alpha_i(x,y)}{\|\nabla_{x,y}\alpha_i(x,y)\|} f_i(x,y)\,d\sigma(x,y) + \iint \nabla f(x,y;\Phi)\theta(\alpha(x,y))\,dxdy$$

Edge sampling

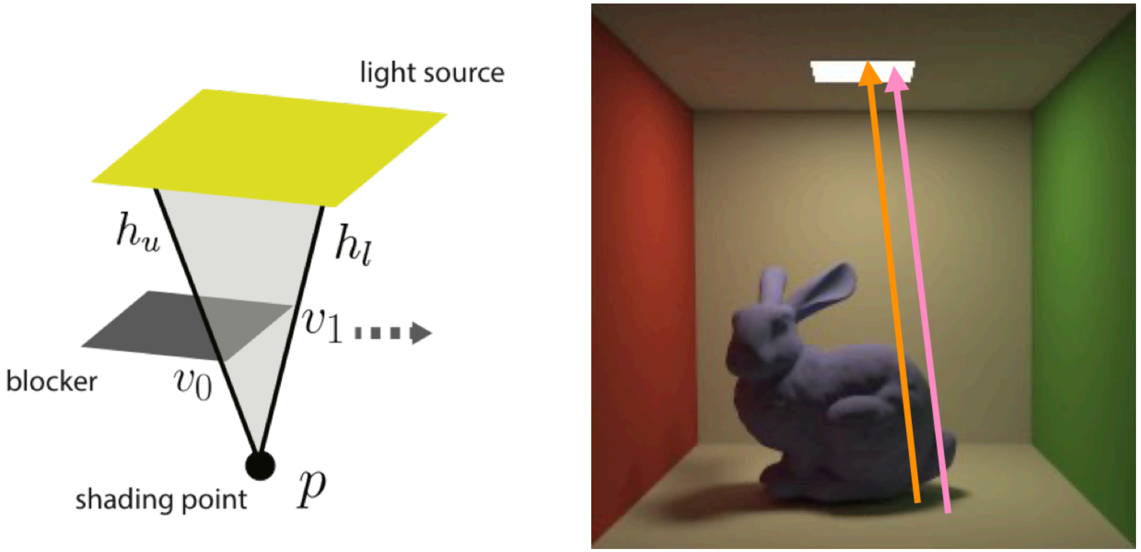


Area sampling

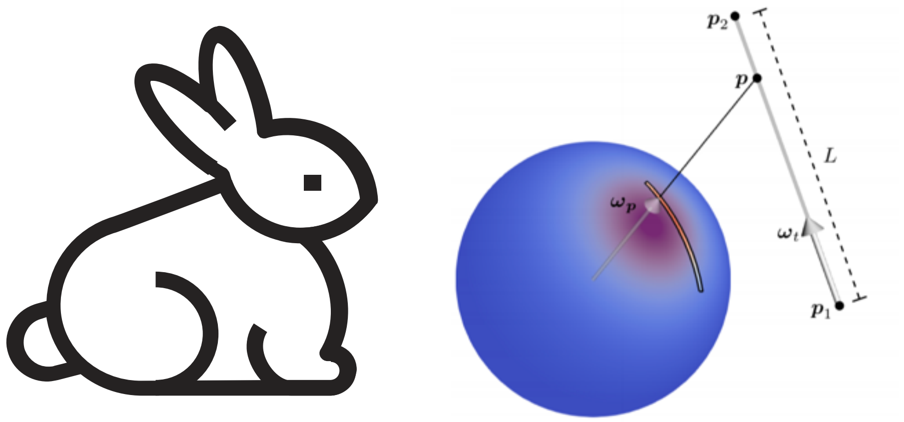


Generalization & Scalability

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



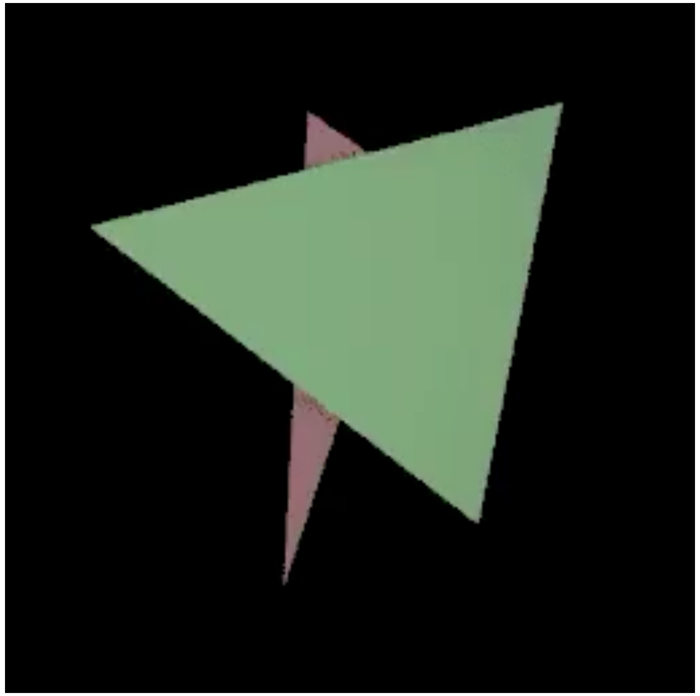
area of a light source



select an edge & pick a point

Experiments – Synthetic examples

- Optimizing 6 triangle vertices

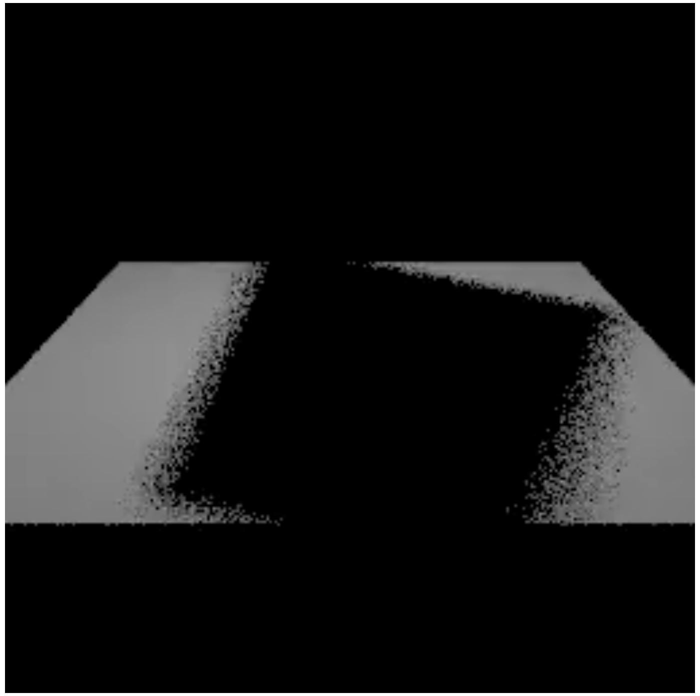


Source

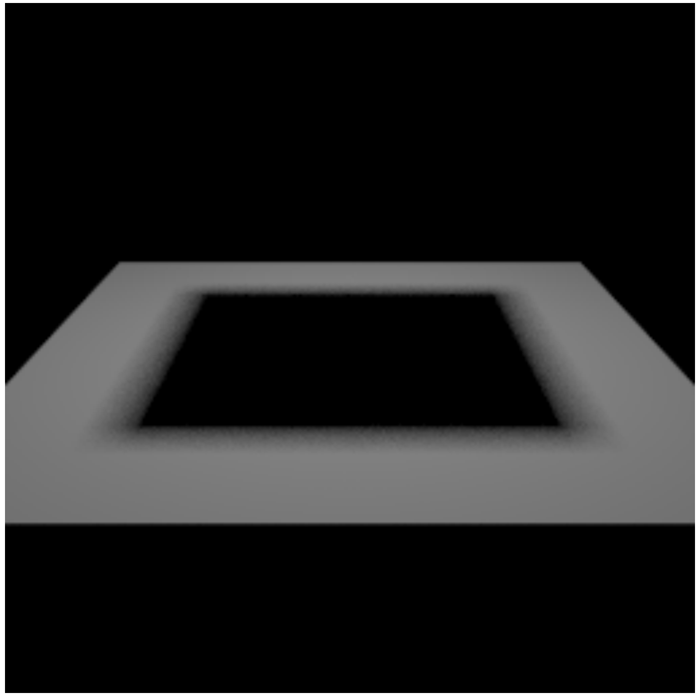
Target

Experiments – Synthetic examples

- Optimizing blocker vertices



Source



Target

Experiments – Synthetic examples

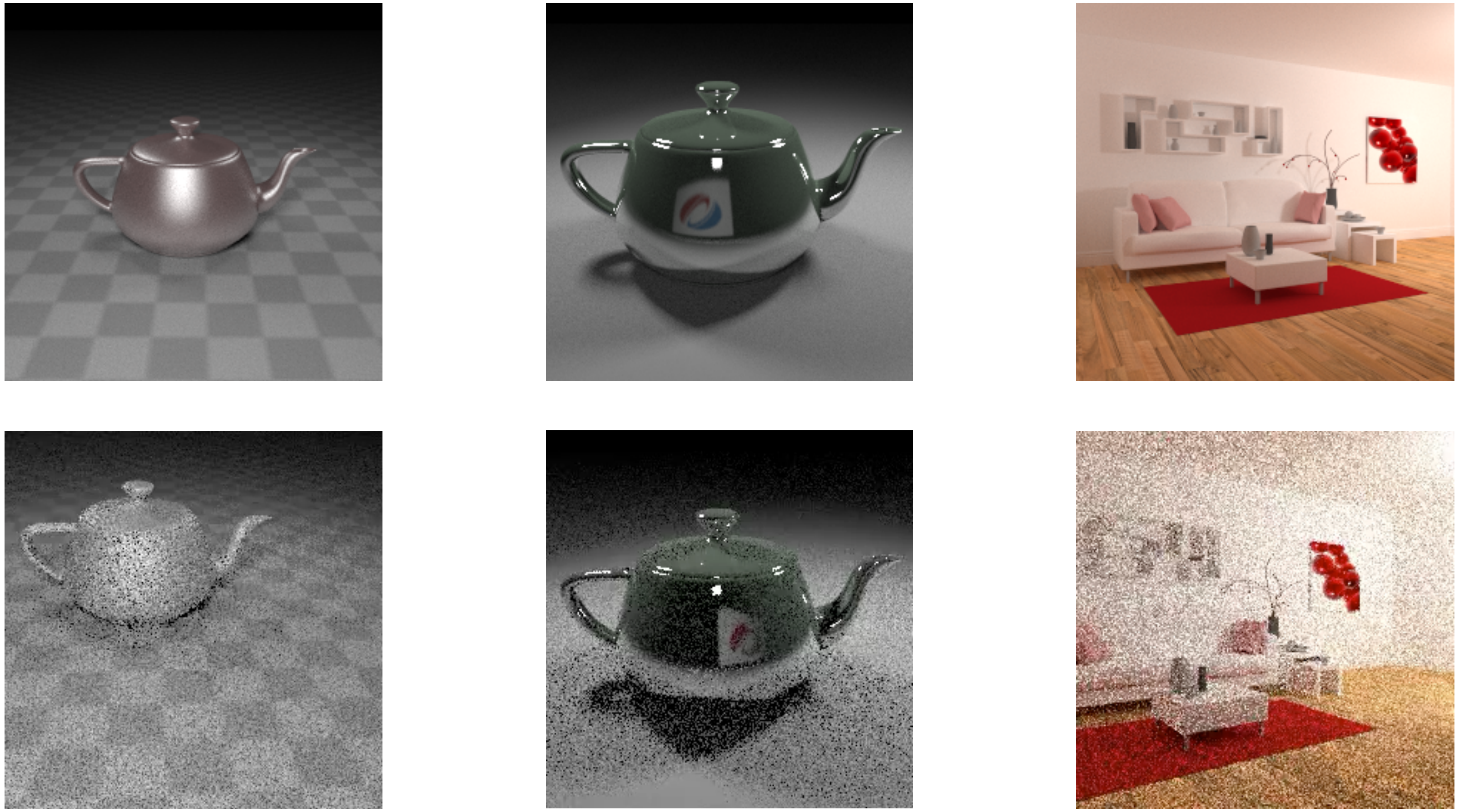
camera & teapot material

logo translation

camera

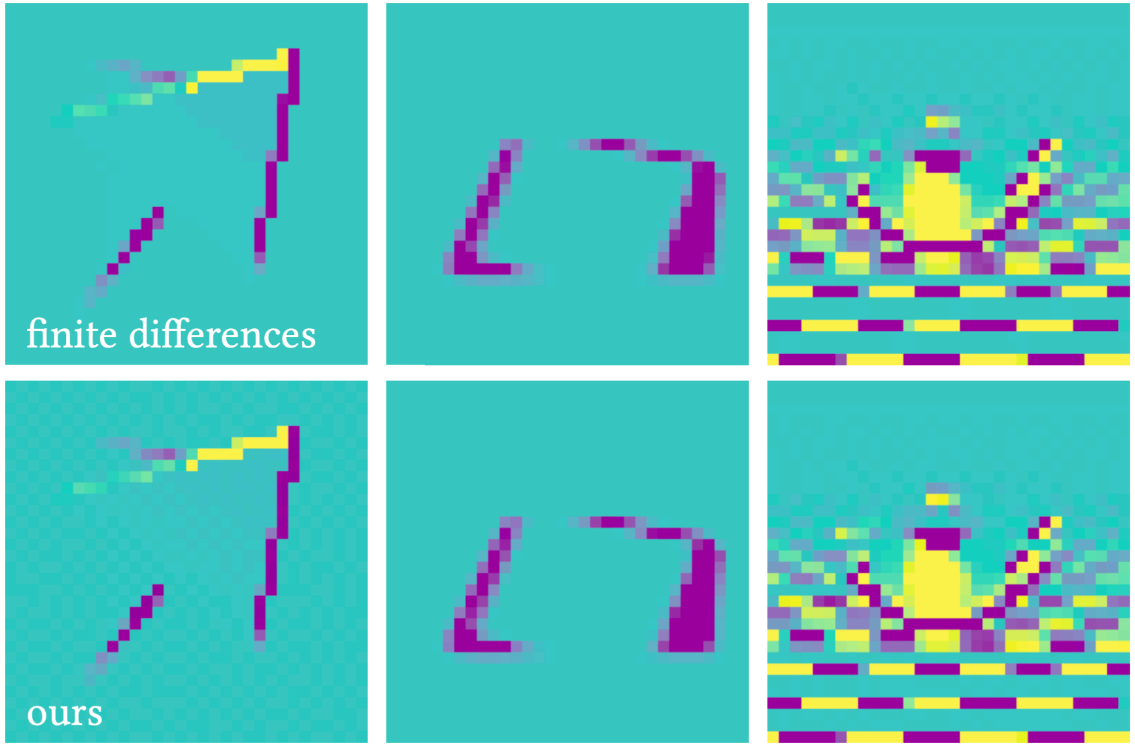
Target

Source



Experiments – Synthetic examples

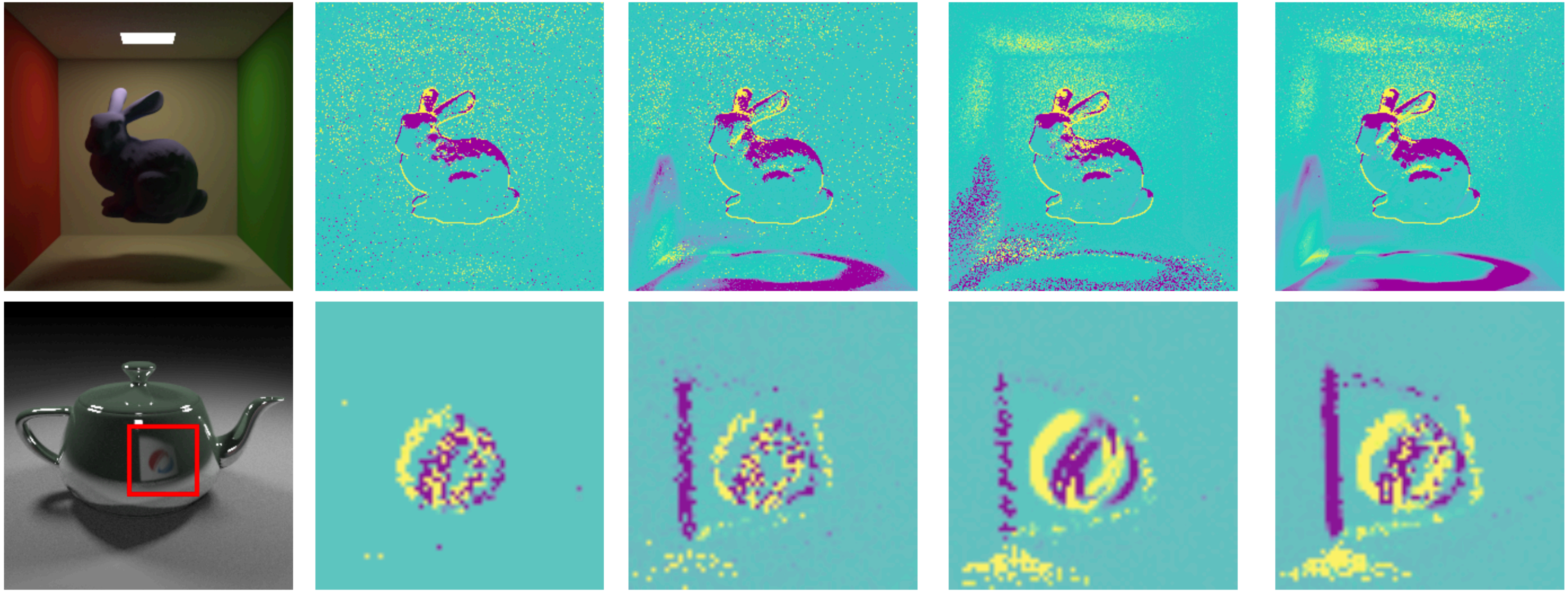
- Compare with central finite differences (32 x 32 scenes)



(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

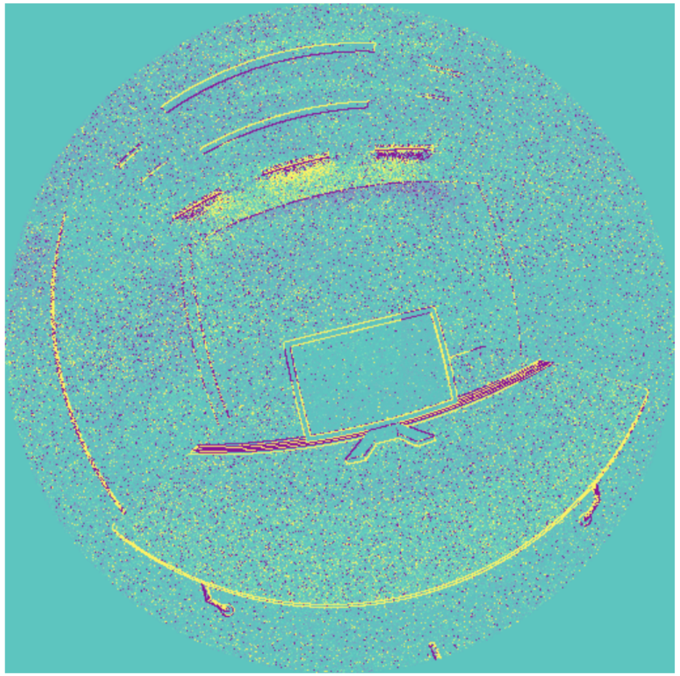
- Optimizing camera pose, light emission and materials

optimization

target

Experiments – Inverse rendering

- Optimizing camera pose, light emission and materials



camera gradient

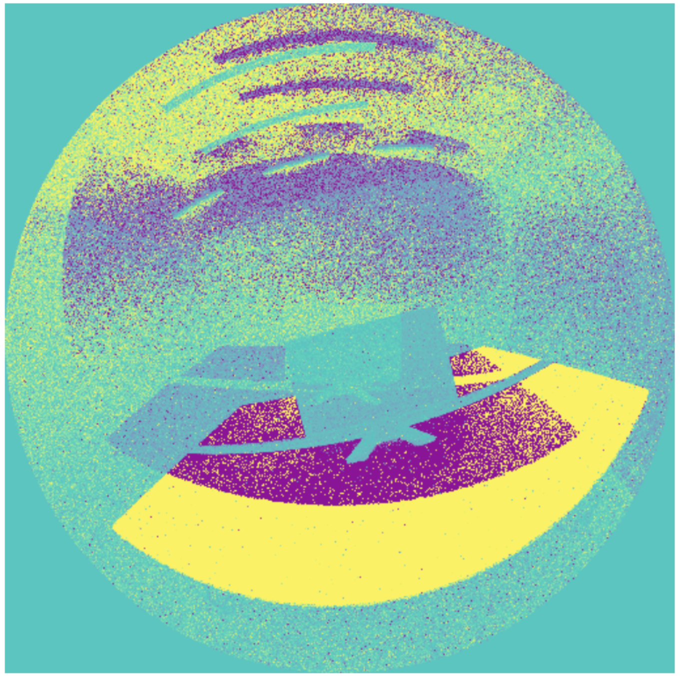
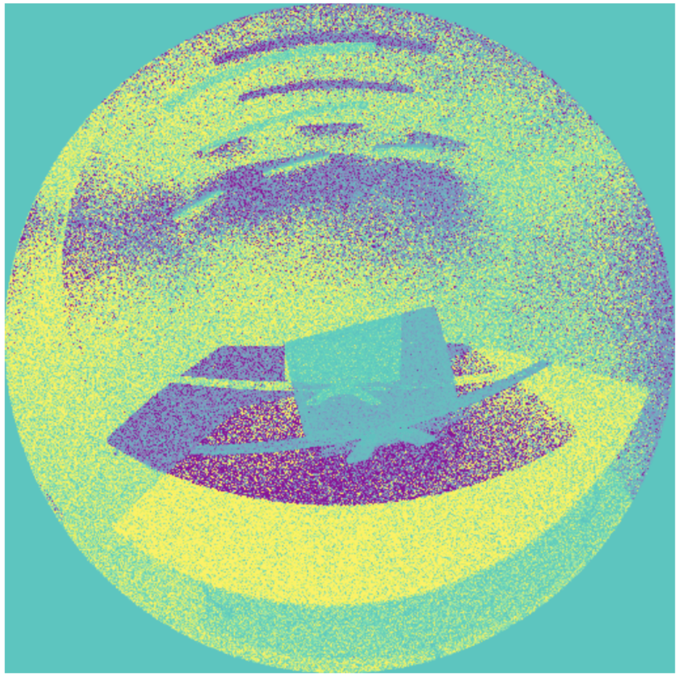


table albedo gradient



light gradient

Experiments – 3D adversarial examples

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

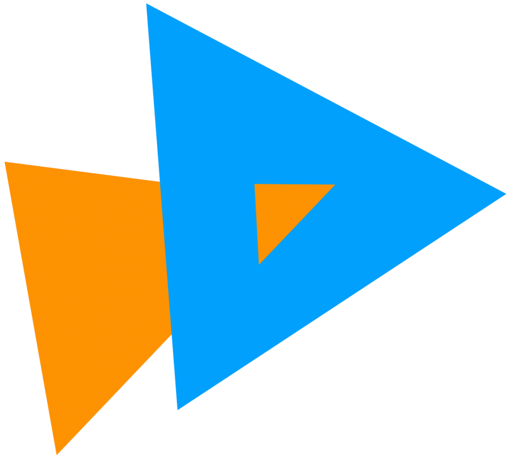
VGG 16:
53% street sign
6.7% handrail

5 iterations:
26.8% handrail
20.2% street sign

25 iterations:
23.3% handrail
3.4% street sign

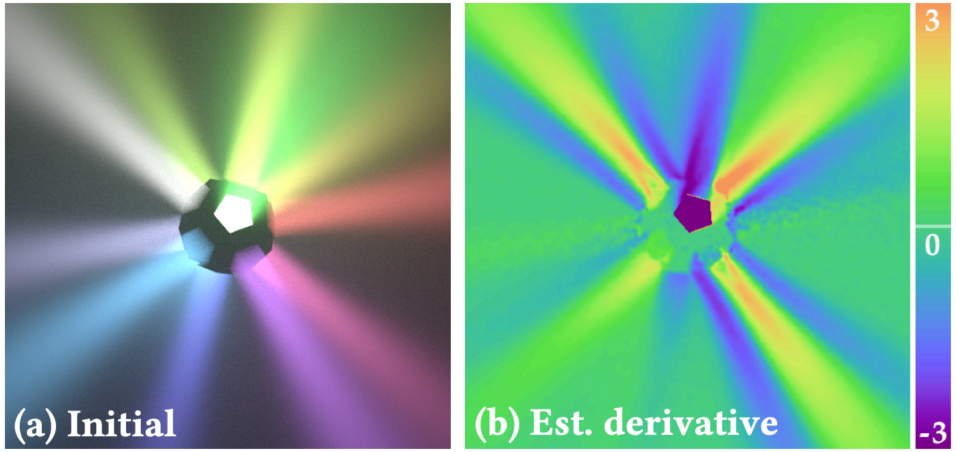
Limitations

- **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
  - A few hundreds of milliseconds to generate a small image (256x256) with a small number of samples (4)
- **Assumptions:**
  - **Mesh**
  - Interpenetrating geometries and parallel edges
  - Shader discontinuities
  - Motion blur

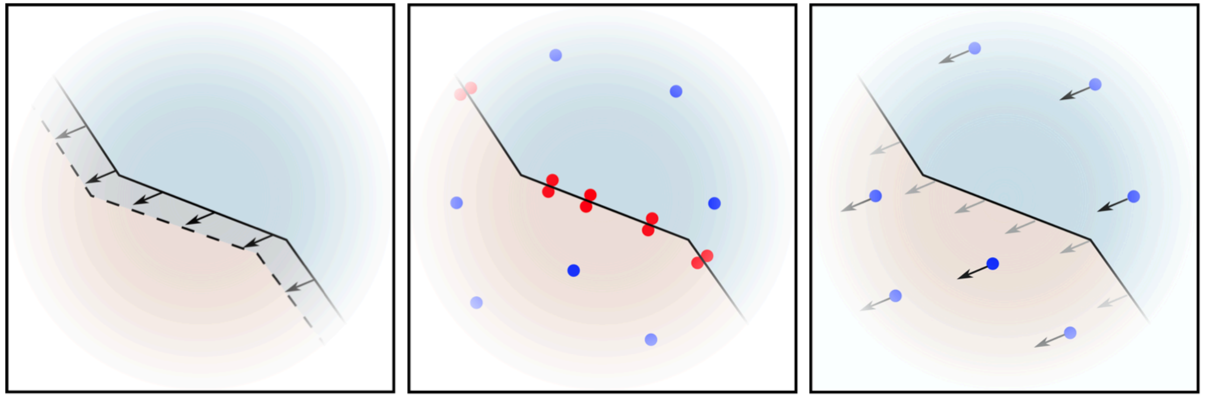


Follow-up works

- Addressing the discontinuity problem in the rendering equation



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



(a) Integrand with discontinuity

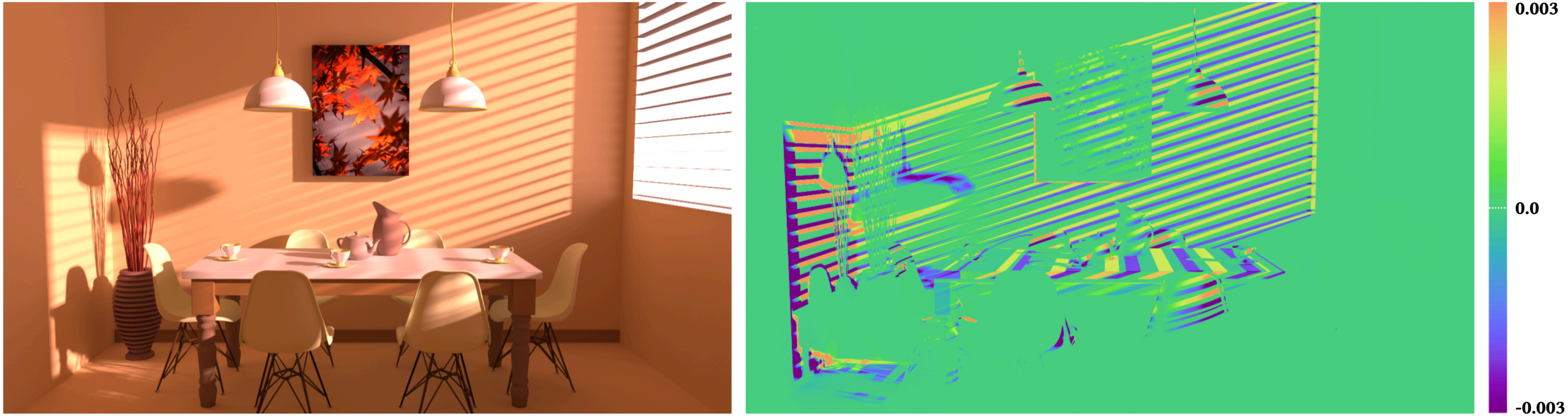
(b) Edge sampling [Li et al. 2018]

(c) Using changes of variables (ours)

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

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)

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

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/