CVPR Tutorial June 20, 2021

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

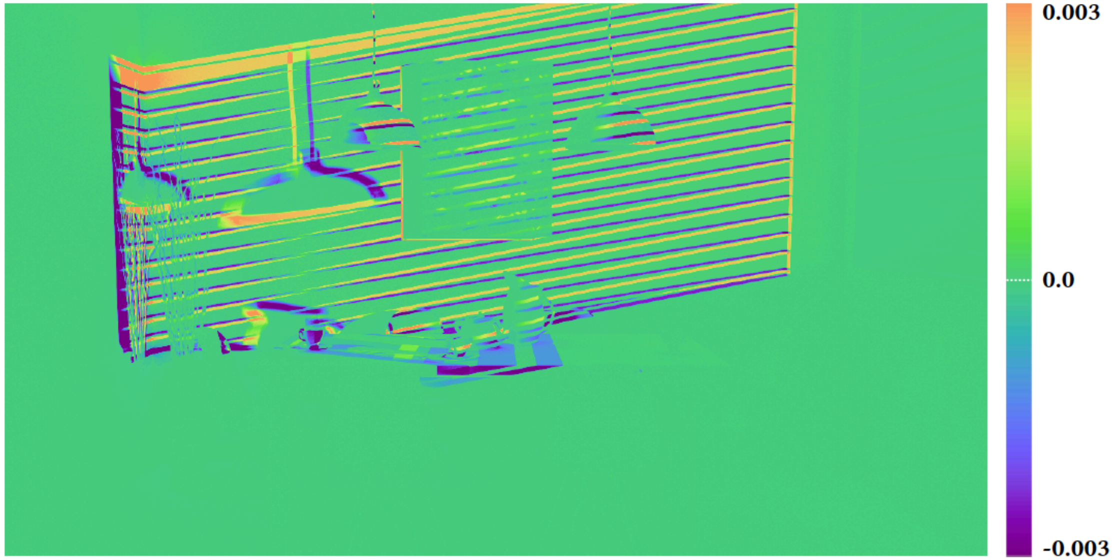
Introduction

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

$$\boldsymbol{I} = \mathscr{R}(\boldsymbol{\theta})$$

Inverse rendering

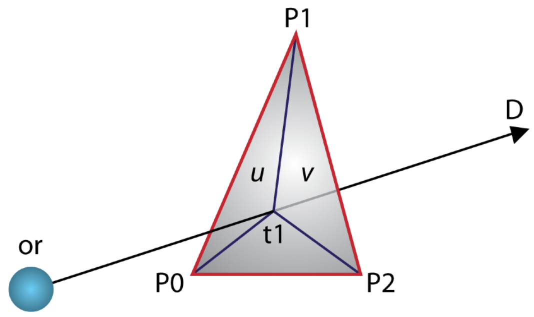
$$\boldsymbol{\theta} = \mathscr{R}^{-1}(\boldsymbol{I})?$$

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

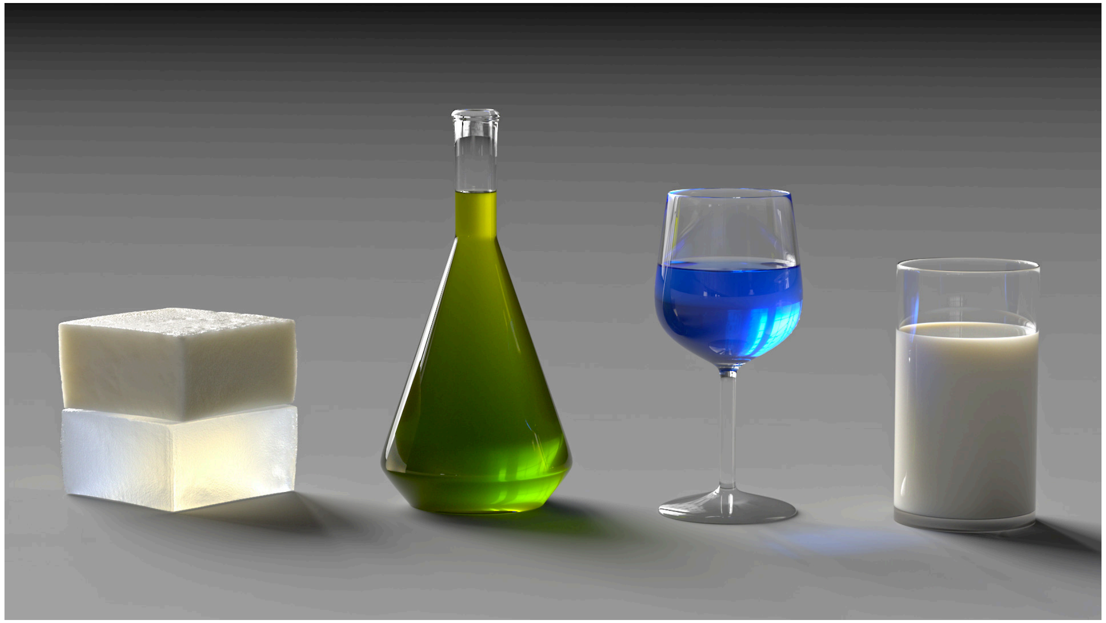


- *Basic building block* for most (if not all) physics-based rendering algorithms
  - e.g., path tracing, bidirectional path tracing, …

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

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

- 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)

(A) CAPTURE (B) PHOTOS (C) INIT. MODEL (D) OPT. MODEL (E) RENDERING

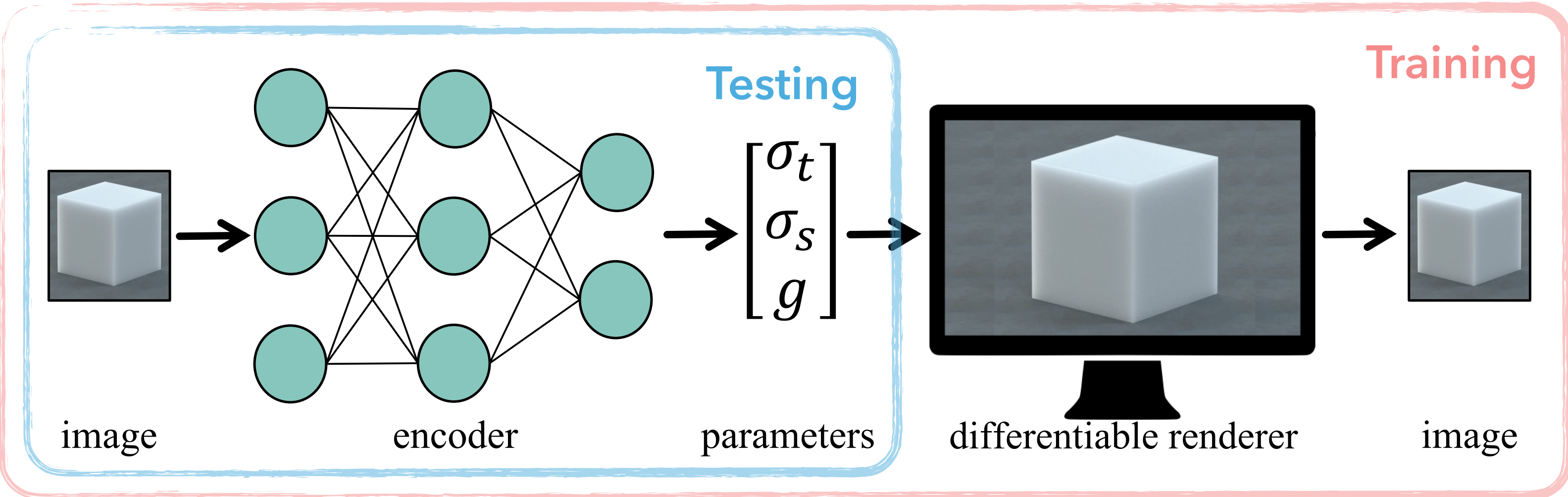
Computational Fabrication

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: $L(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o}) = \int_{\mathbb{S}^2} f_\mathrm{s}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o})\, \mathrm{d}\boldsymbol{\omega}_\mathrm{i} + L_\mathrm{e}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{o})$
  - 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: $\boldsymbol{I} = \mathscr{R}(\boldsymbol{\theta})$
  - $\boldsymbol{\theta} \in \mathbb{R}^n$ (n: number of parameters)
  - $\boldsymbol{I} \in \mathbb{R}^m$ (m: number of pixels)
- Gradient matrix: $\dfrac{\mathrm{d}\mathscr{R}}{\mathrm{d}\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^{m \times n}$
- Challenges:
  - m and n can both be large ($\sim 10^6$)
  - $(\mathrm{d}\mathscr{R}/\mathrm{d}\boldsymbol{\theta})$ 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

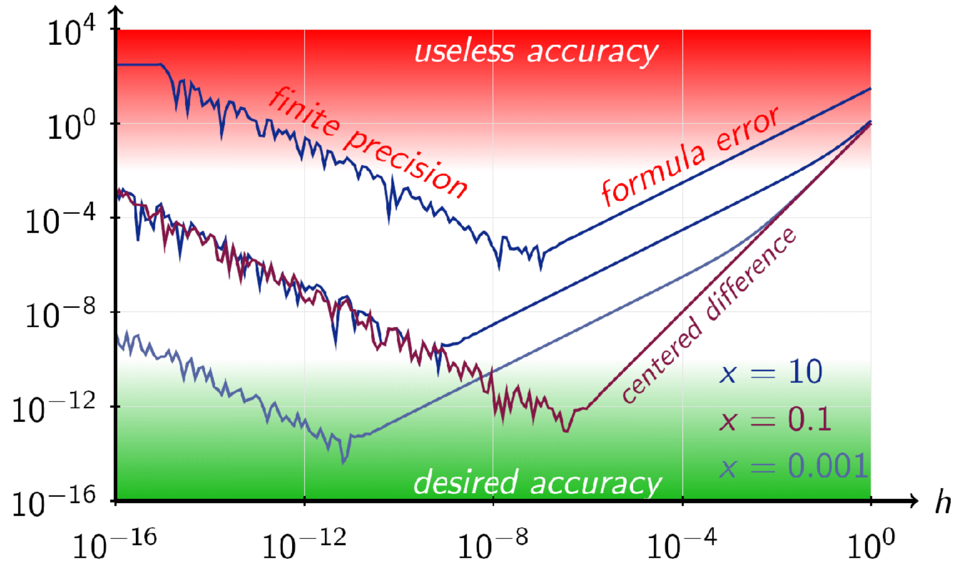
- Should the PDF (used by a Monte Carlo estimator) be differentiated?
  - Can go either way…
    (More on this later.)

- Discontinuities
  - Differentiating only the integrand is insufficient
    (More on this later.)

Why Not Simply Use Finite Differences?

Finite difference:

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



[Wikipedia]

Potential problems:

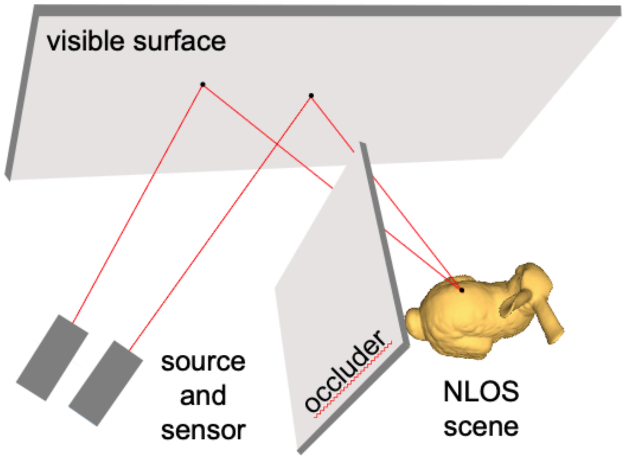
- High bias (large ε), rounding error (small ε)
- Need to correlate Monte Carlo samples
- Scales poorly with the number of parameters

Global Illumination

- Can be simulated with modern differentiable renderers

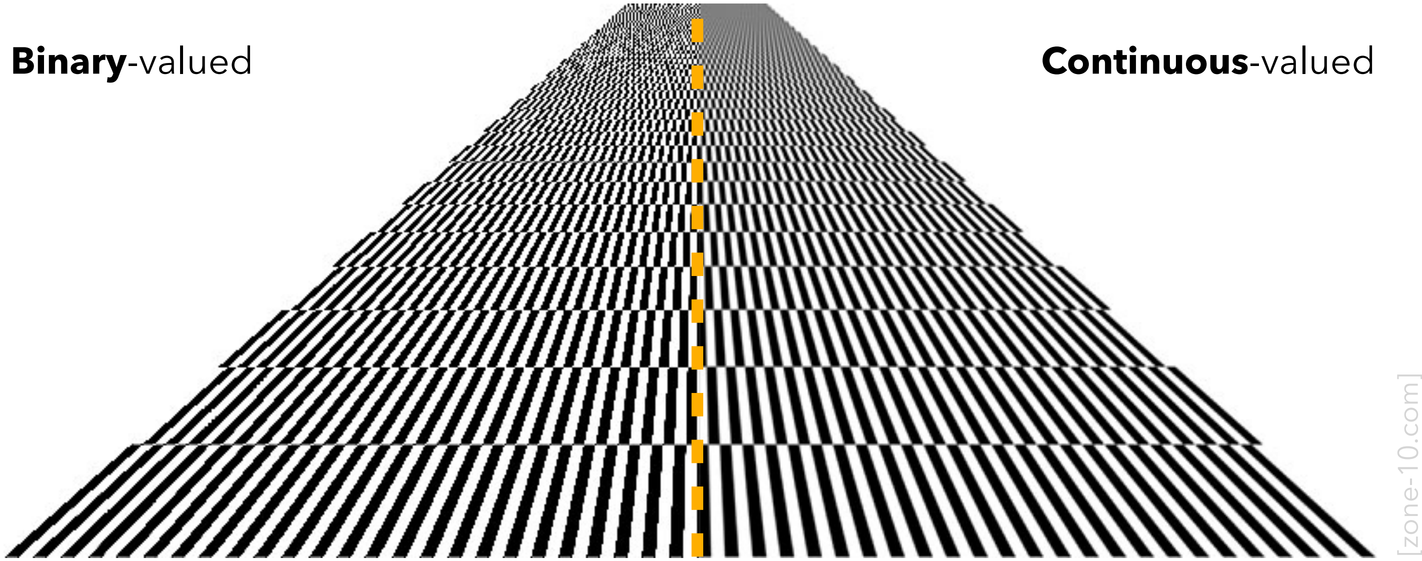
- Required when solving many inverse-rendering problems

Computational fabrication



Non-line-of-sight imaging

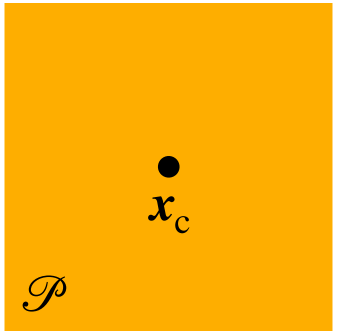
Pixel-Level Antialiasing Matters



No antialiasing

Perfect antialiasing

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



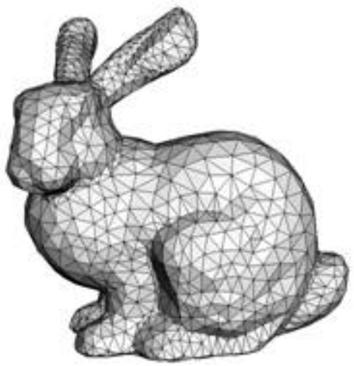
One pixel

$$\boxed{\text{Pixel value} = \frac{1}{|\mathscr{P}|}\int_{\mathscr{P}} I(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}$$

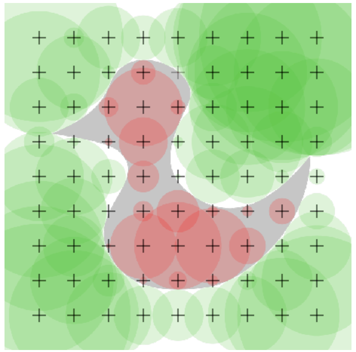
More information, more differentiable!

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)

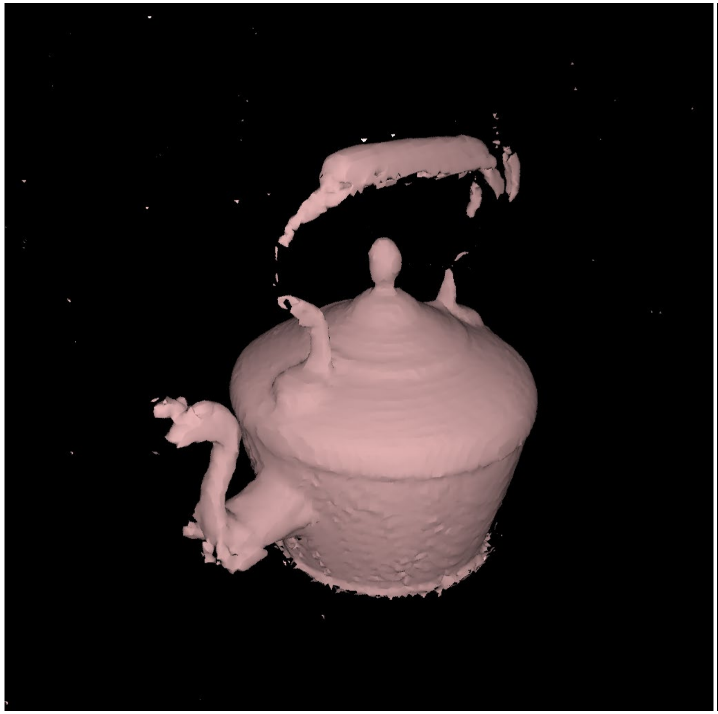
Why you should use ray-tracing-based differentiable rendering

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:

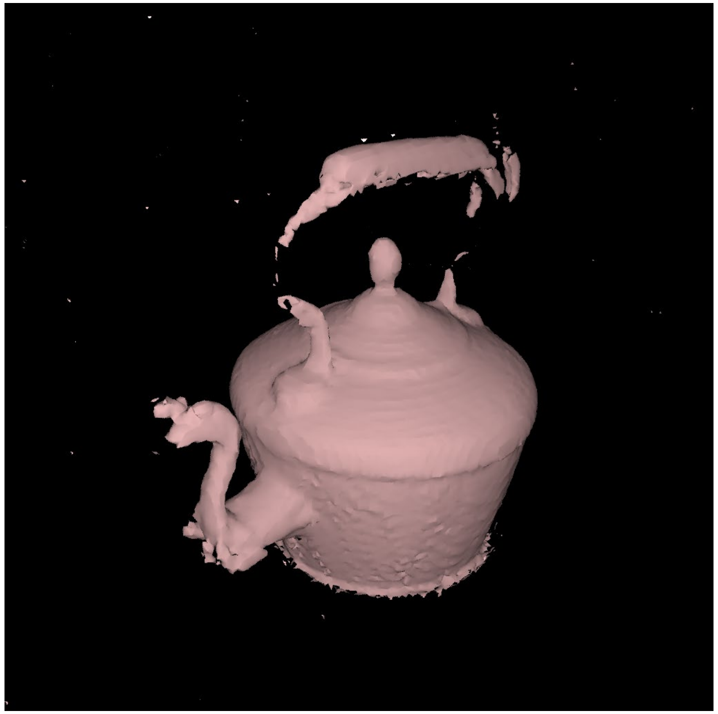
- **psdr-cuda** (ray-tracing-based)*: 2.8 msec
- **PyTorch3D** (soft rasterizer): 52.5 msec

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)

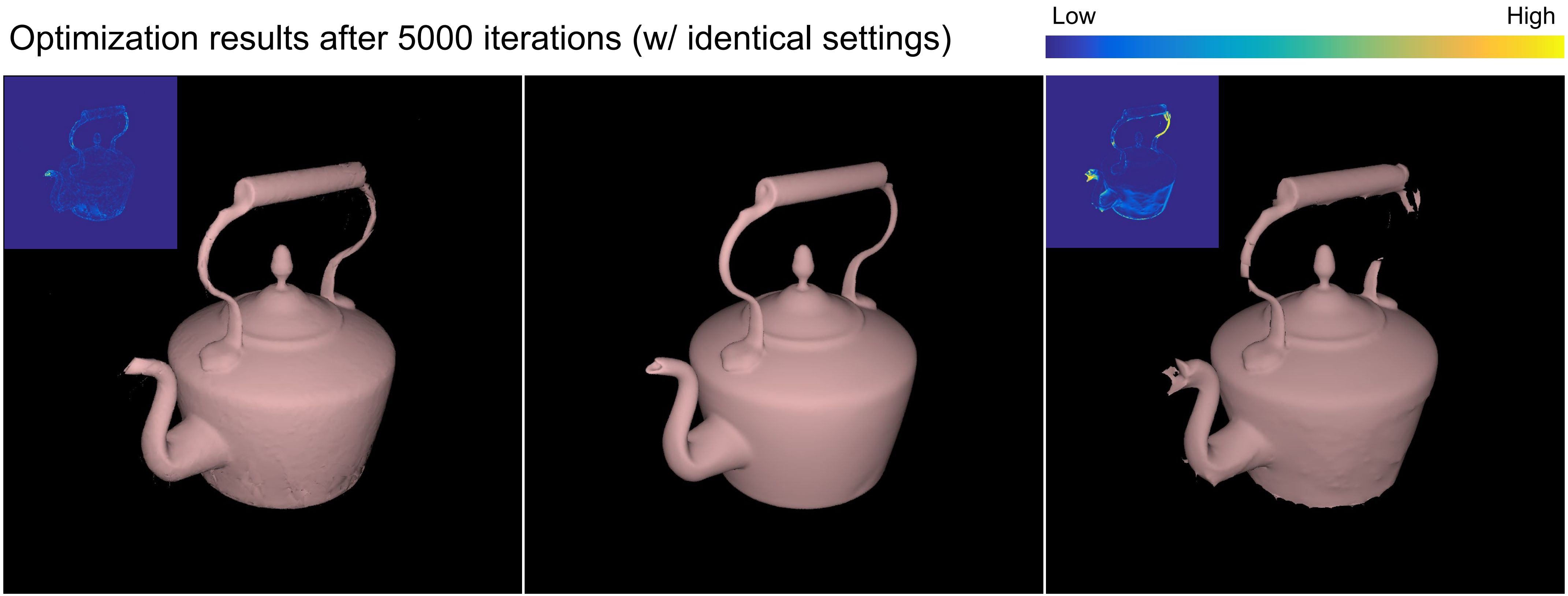
Absolute error

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
- Ray-tracing-based methods can compute **correct** (i.e., unbiased) gradients
  - Correct gradients matter in optimization!

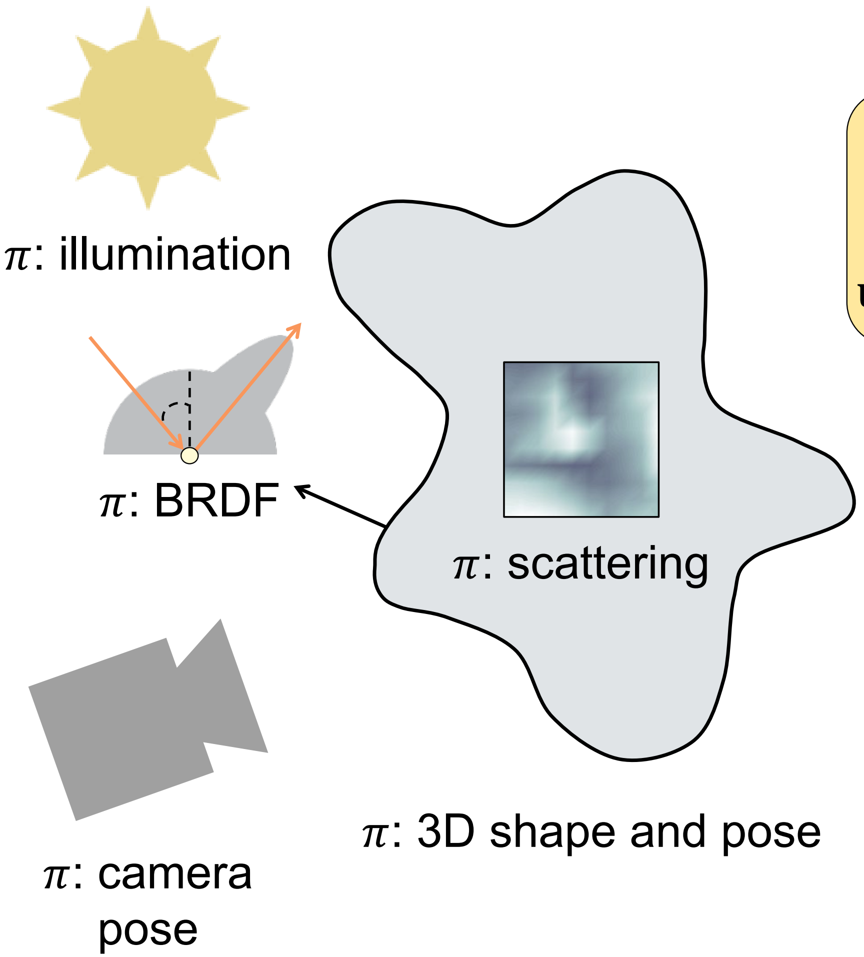
Second part of this tutorial

- Ray-tracing-based methods can handle **complex** light-transport effects
  - Soft shadows, environmental illumination
  - Inter-reflections, radiative transfer (e.g., subsurface scattering), caustics
- 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, ...)

Third part of this tutorial

What differentiable rendering does not give us

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



Analysis-by-synthesis optimization:

$$\min_{\substack{\text{scene}\\ \text{unknowns } \pi}} \ \text{loss}\Bigg[$$

$$,\ \text{render}\left(\begin{matrix}\text{scene}\\ \text{unknowns } \pi\end{matrix}\right)\Bigg]$$

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

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

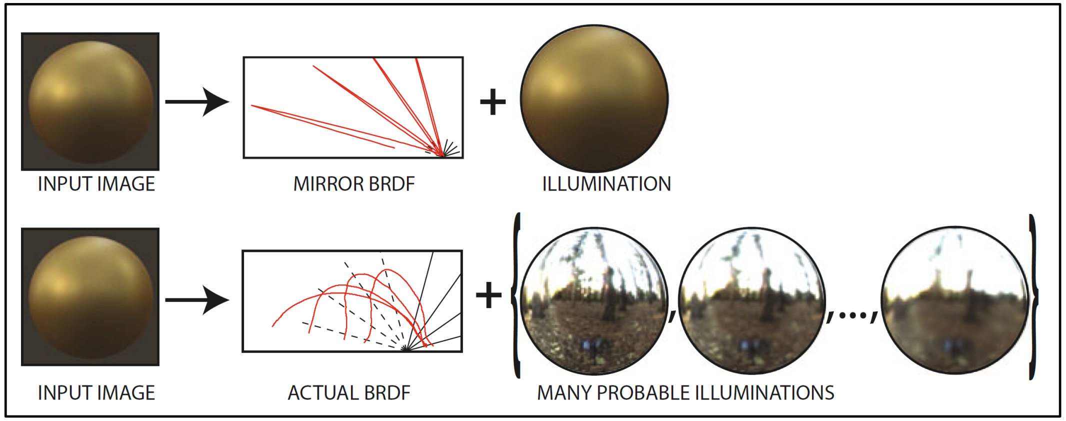
while (not converged)

update $\pi \leftarrow \pi + \eta \cdot \boxed{\dfrac{\mathrm{dloss}(\pi)}{\mathrm{d}\pi}}$

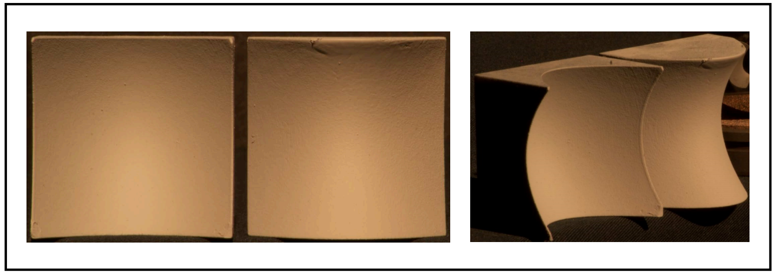
Differentiable rendering

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 BRDF and lighting [Romeiro and Zickler 2010]



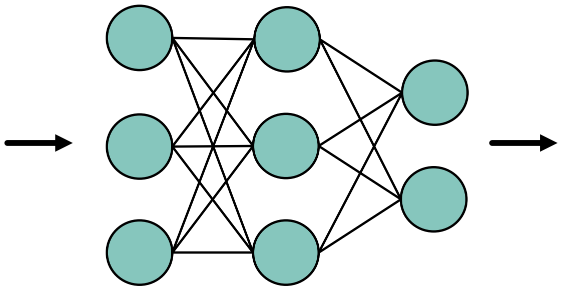
Ambiguities between shape and lighting [Xiong et al. 2015]

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:

$$\min_{\substack{\text{scene}\\ \text{unknowns } \pi}} \boxed{\text{loss}} \Big[$$

$$, \text{render}\Big(\substack{\text{scene}\\ \text{unknowns } \pi}\Big)\Big]$$

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

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

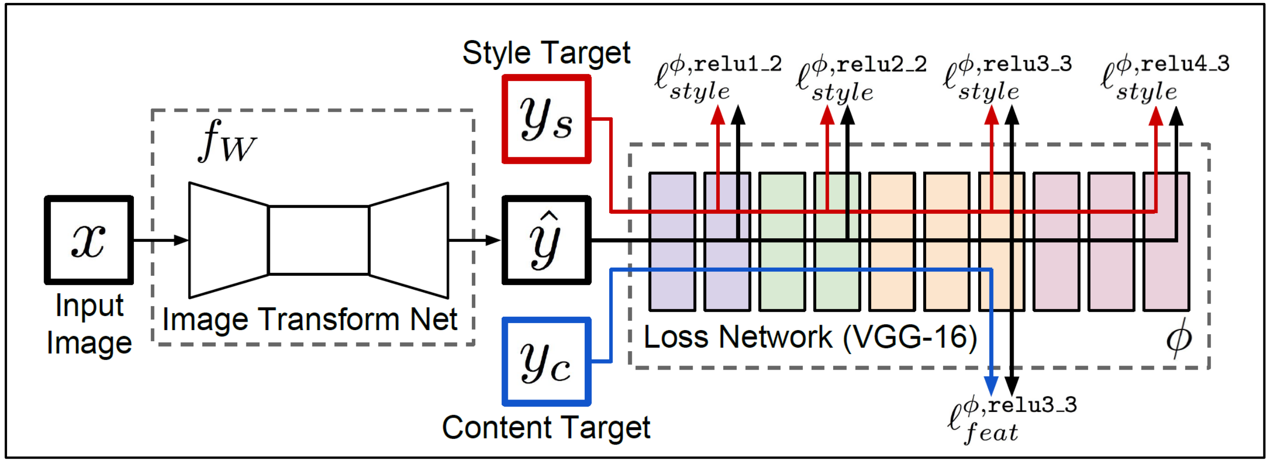
while (not converged)

update $\pi \leftarrow \pi + \eta \cdot \boxed{\dfrac{\mathrm{dloss}(\pi)}{\mathrm{d}\pi}}$

Differentiable rendering

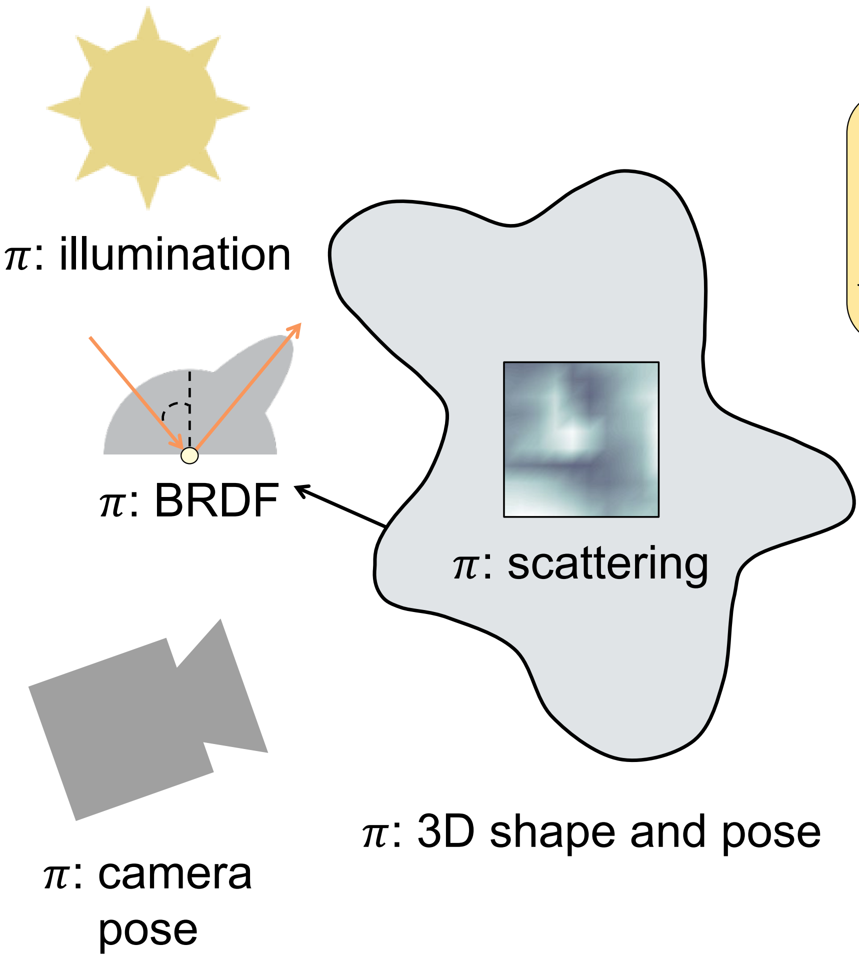
Why we need discriminative loss functions

- 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:

$$\min_{\substack{\text{scene}\\ \text{unknowns } \pi}} \boxed{\text{loss}} \Big[$$

$$, \text{render}\left(\substack{\text{scene}\\ \text{unknowns } \pi}\right)\Big]$$

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

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

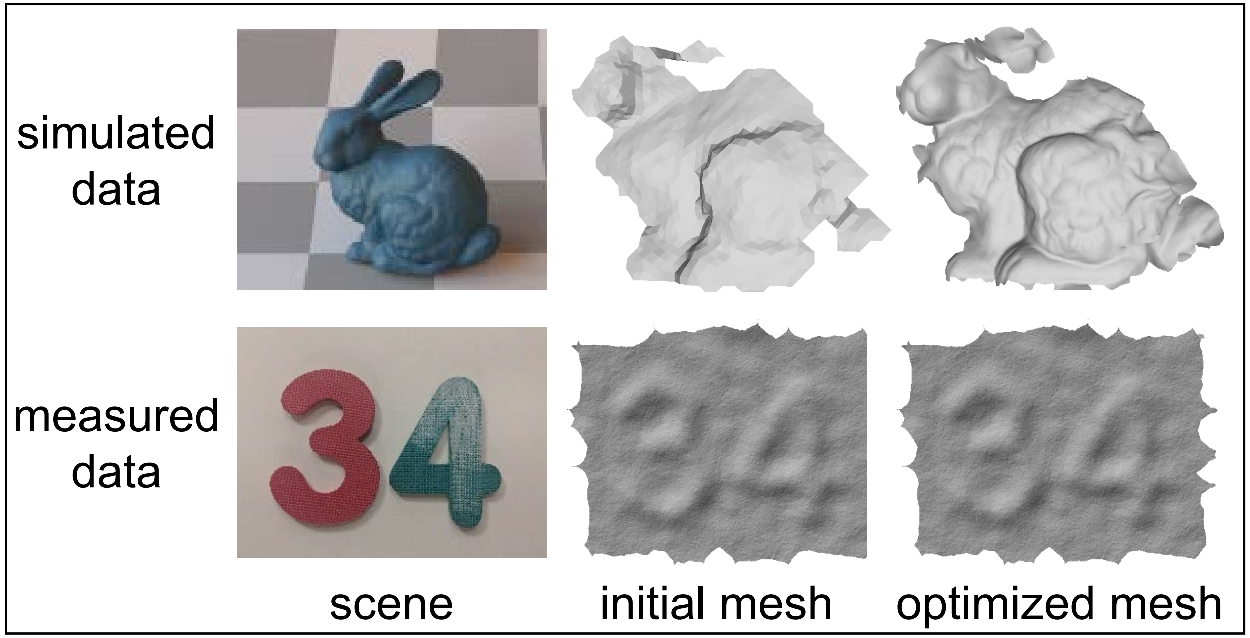
while (not converged)

update $\pi \leftarrow \pi + \eta \cdot \boxed{\frac{\mathrm{dloss}(\pi)}{\mathrm{d}\pi}}$

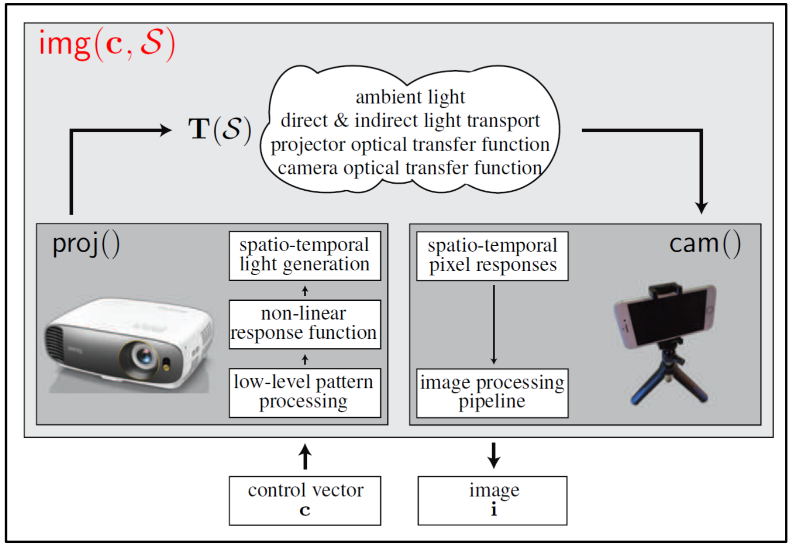
Differentiable rendering

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

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

Move derivative inside integral

Account for changes in integration limits

$$+\ f(b(\pi),\pi)\frac{\mathrm{d}b(\pi)}{\mathrm{d}\pi} - f(\alpha(\pi);\pi)\frac{\mathrm{d}a(\pi)}{\mathrm{d}\pi}$$

Account for discontinuities of integrand that depend on π

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

A simple example

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

$$\frac{\mathrm{d}}{\mathrm{d}\pi}\int_{0}^{4\pi} f(x,\pi)\mathrm{d}x = \int_0^{2\pi}\frac{\mathrm{d}}{\mathrm{d}\pi}0\mathrm{d}x + \int_{\pi}^{4\pi}\frac{\mathrm{d}}{\mathrm{d}\pi}1\mathrm{d}x$$

Move derivative inside integral

Account for changes in integration limits

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

Account for discontinuities of integrand that depend on π

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

Leibniz integral rule

Differentiation under the integral sign
Also known as the Leibniz integral rule

Interior integral

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

Move derivative inside integral

Boundary terms

Account for changes in integration limits

$$+\ f(b(\pi),\pi)\frac{\mathrm{d}b(\pi)}{\mathrm{d}\pi} - f(\alpha(\pi);\pi)\frac{\mathrm{d}a(\pi)}{\mathrm{d}\pi}$$

Account for discontinuities of integrand that depend on π

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

Simplified Leibniz integral rule

Differentiation under the integral sign
Also known as the Leibniz integral rule

Interior integral

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

Move derivative inside integral

Differentiation wrt $\boldsymbol{\pi}$ simplifies to just moving derivative inside integral when:

- Integration limits are independent of $\boldsymbol{\pi}$.
- Integrand discontinuities are independent of $\boldsymbol{\pi}$.

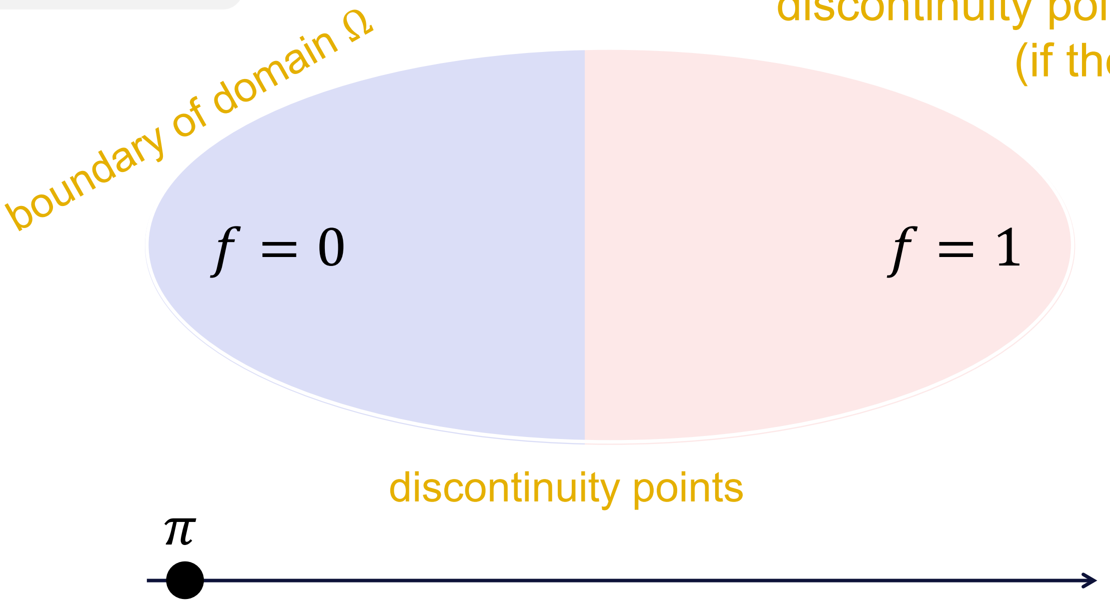
Reynolds transport theorem

Interior integral · Boundary integral

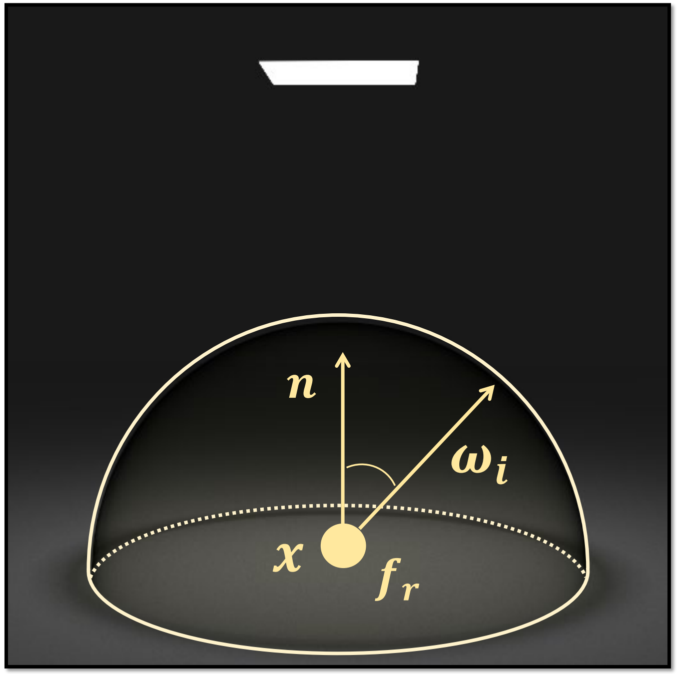
$$\frac{\mathrm{d}}{\mathrm{d}\pi}\int_{\Omega(\pi)} f(x,\pi)\,\mathrm{d}A(x) \overset{?}{=} \underbrace{\int_{\Omega(\pi)} \frac{\mathrm{d}f(x,\pi)}{\mathrm{d}\pi}\,\mathrm{d}A(x)}_{\text{Interior integral}} + \underbrace{\int_{\partial\Omega(\pi)} g(x,\pi)\,\mathrm{d}l(x)}_{\text{Boundary integral}}$$

Reynolds transport theorem [1903]
Generalization of the Leibniz rule

Boundary domain
||
discontinuity points ∪ boundary of domain Ω
(if they depend on π)



Direct illumination integral



Radiance from x:

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

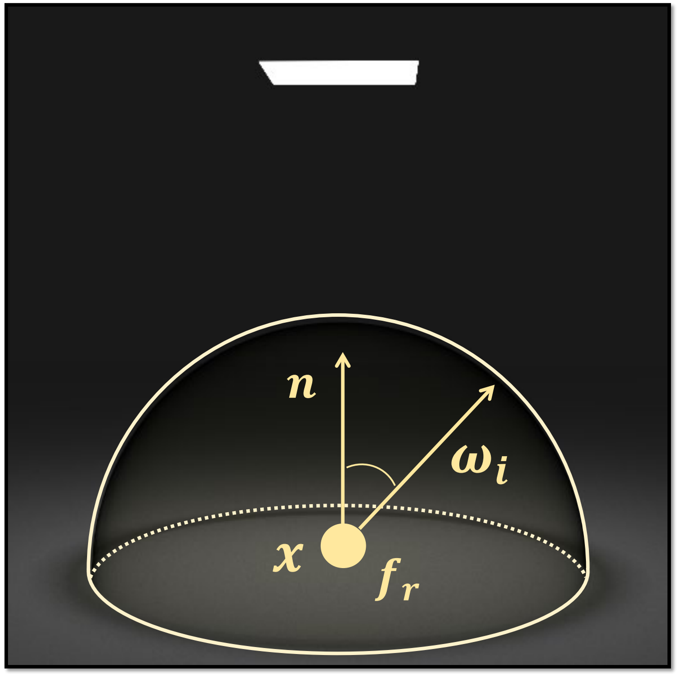
Unit hemisphere ($\mathbb{H}^2$)

Monte Carlo rendering:

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

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

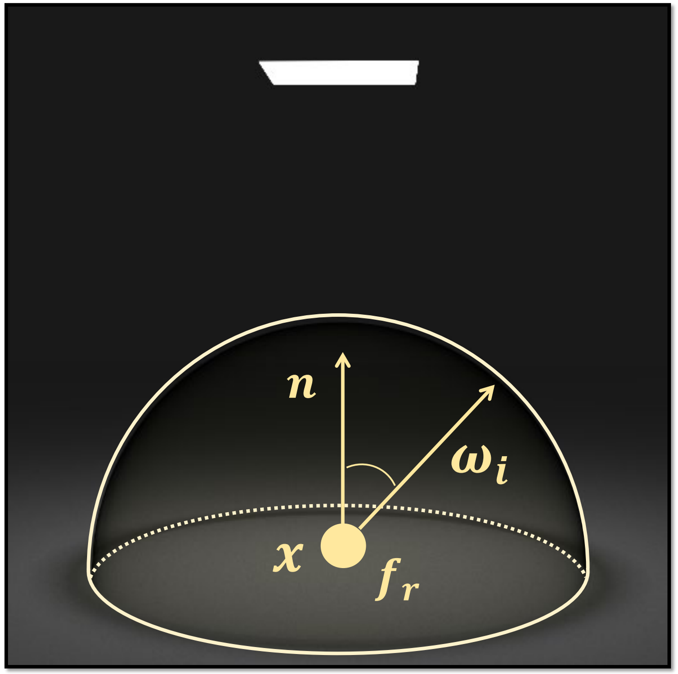
Differential direct illumination



Differential radiance from x:

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

Differential direct illumination: local parameters



π: *local* parameters
- BRDF parameters
- *shading* normal
- illumination brightness

Differential radiance from x:

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

Just move derivative inside integral

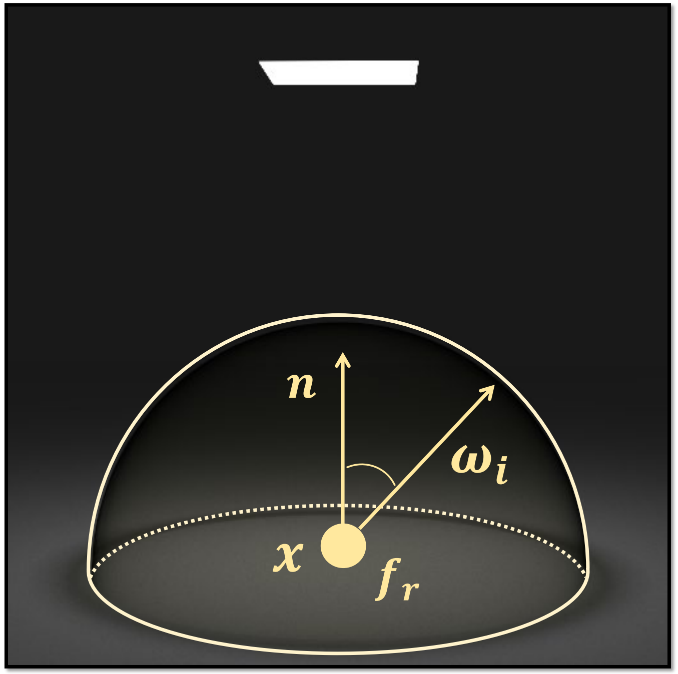
Monte Carlo differentiable rendering:

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

Just differentiate numerator
[Khungurn et al. 2015, Gkioulekas et al. 2015]

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

Alternative estimator



π: *local* parameters

- BRDF parameters

Differential radiance from x:

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

Just move derivative inside integral

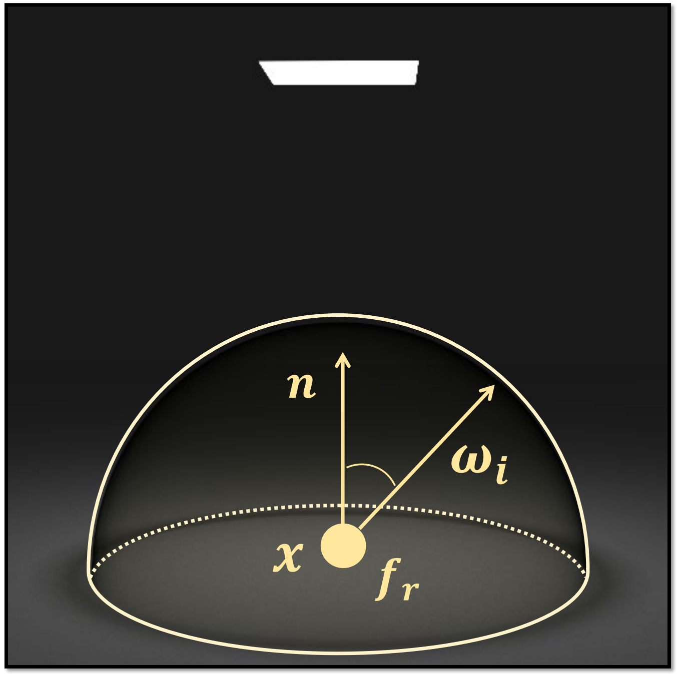
Monte Carlo estimation:

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

Differentiate entire contribution [Zeltner et al. 2021]

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

Differential direct illumination: global parameters



π: *global* parameters

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

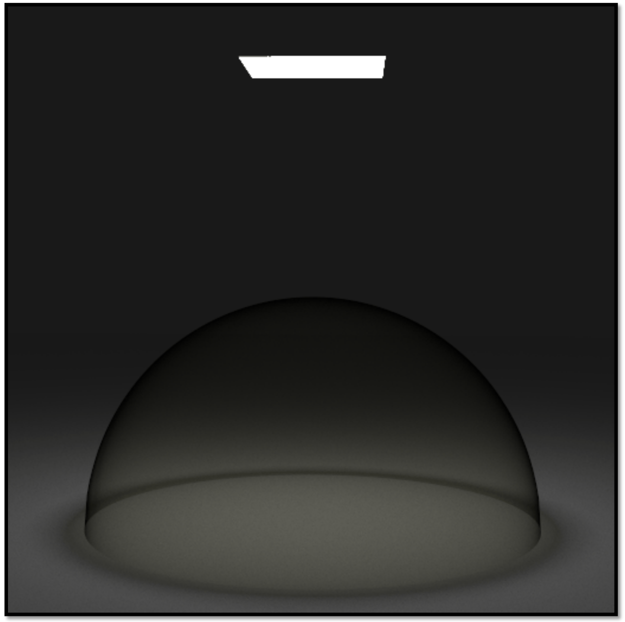
Differential radiance from x:

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

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

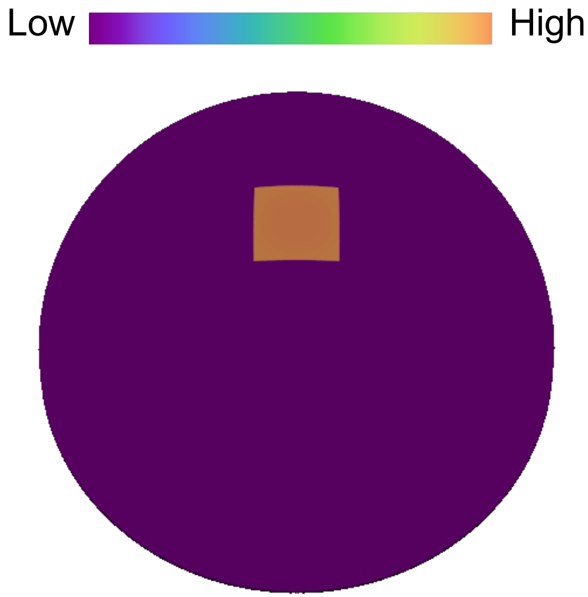
Need to use full Reynolds transport theorem

Discontinuities in the integrand

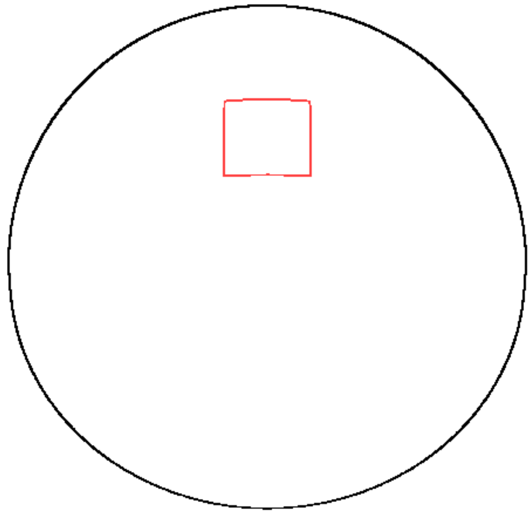


π: size of the emitter

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



Integrand
$f(\omega_i)$



Discontinuous points
(π-dependent)

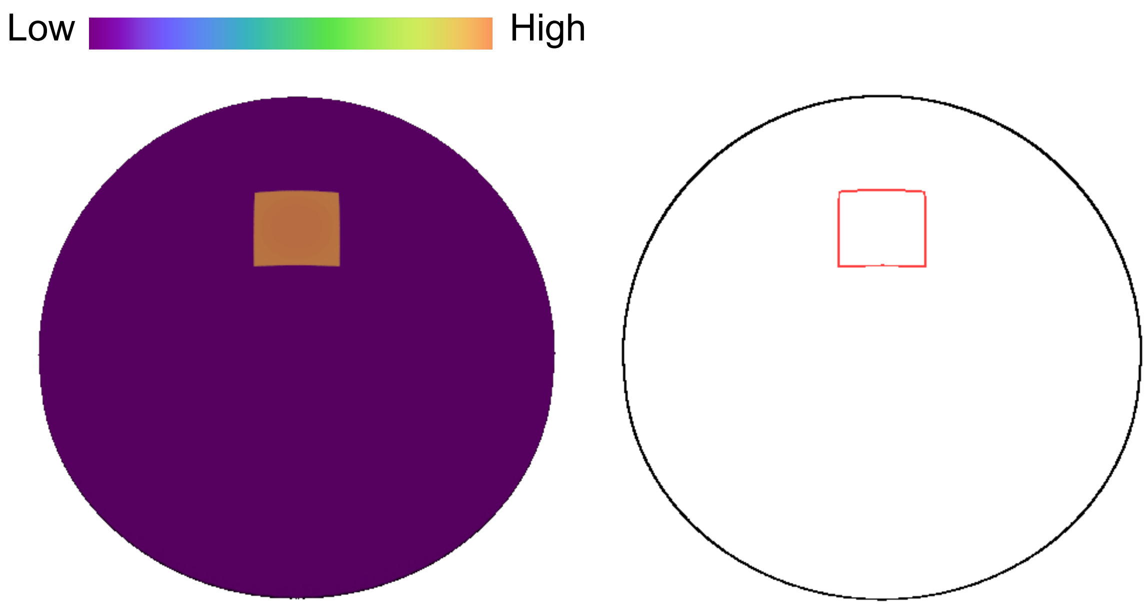
Applying the Reynolds transport theorem

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

$$\frac{\mathrm{d}I}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}f}{\mathrm{d}\pi} \mathrm{d}\sigma + \int_{\partial\mathbb{H}^2} g \, \mathrm{d}l$$

Interior integral (same as for local parameters)

Boundary integral



Integrand $f(\omega_i)$

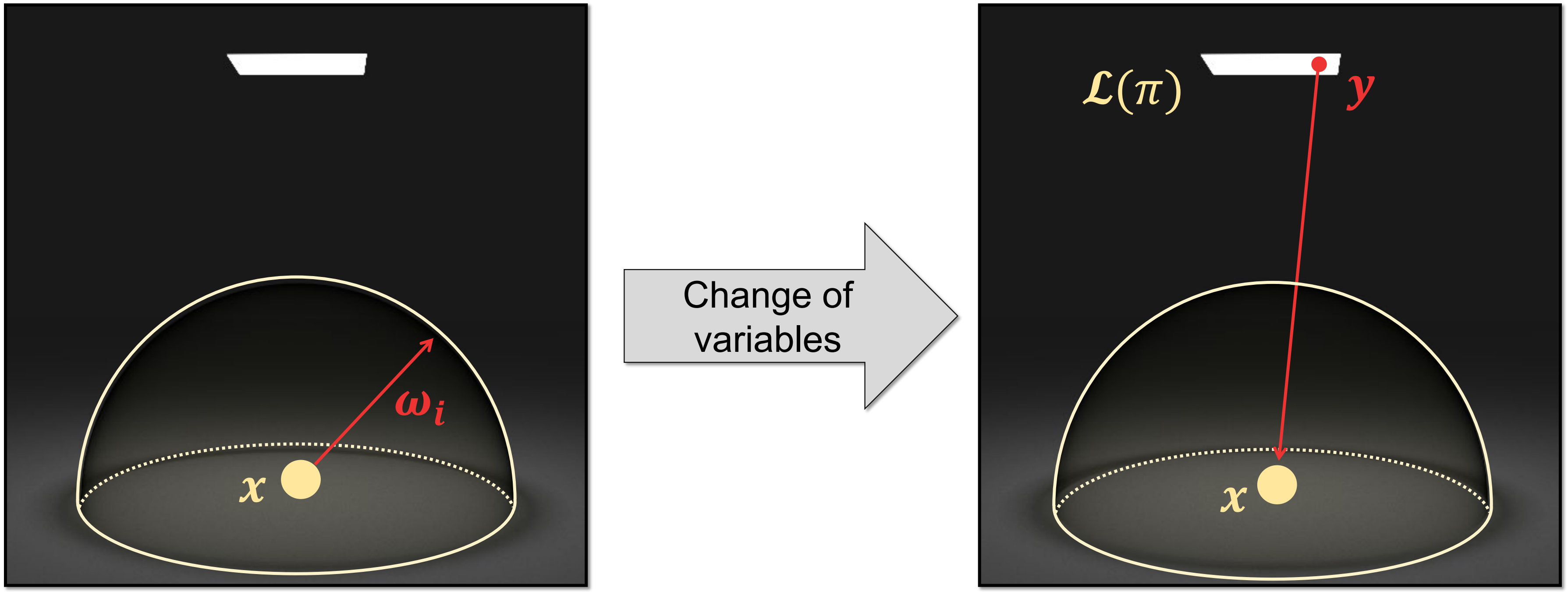
Discontinuous points (π-dependent)

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

Reparameterizing the direct illumination integral

Hemispherical integral

Surface integral



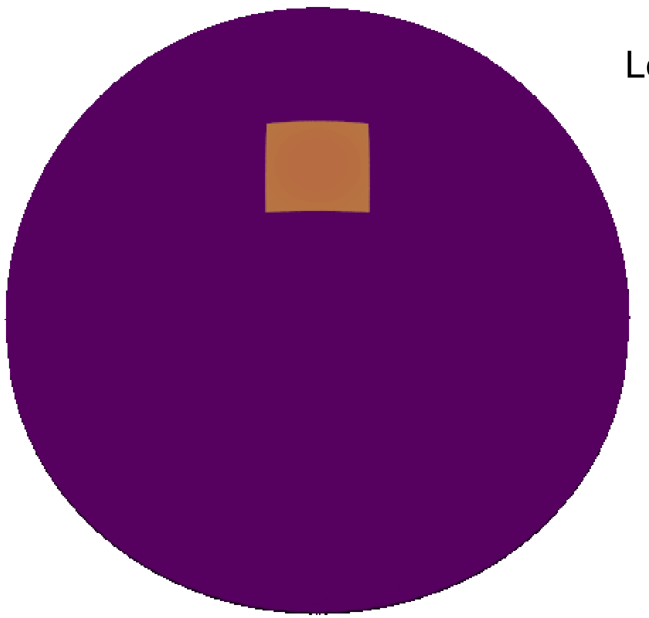
$$I = \int_{\mathbb{H}^2} f(\boldsymbol{\omega_i})\, \mathrm{d}\sigma(\boldsymbol{\omega_i})$$

$$I = \int_{\mathcal{L}(\pi)} f(\boldsymbol{y} \to \boldsymbol{x})\, \boxed{G(\boldsymbol{x}, \boldsymbol{y})}\, \mathrm{d}A(\boldsymbol{y})$$

Includes visibility, fall-off, and foreshortening terms

Reparameterizing the direct illumination integral

Hemispherical integral



Low  High

Surface integral

Change of variables

discontinuous

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

constant domain

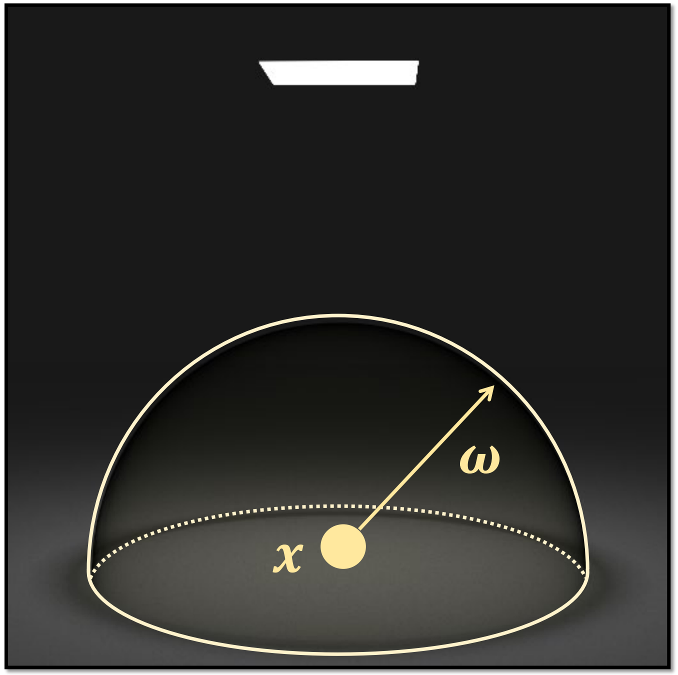
continuous

$$I = \int_{\mathcal{L}(\pi)} f(y \to x)\, G(x, y)\, \mathrm{d}A(y)$$

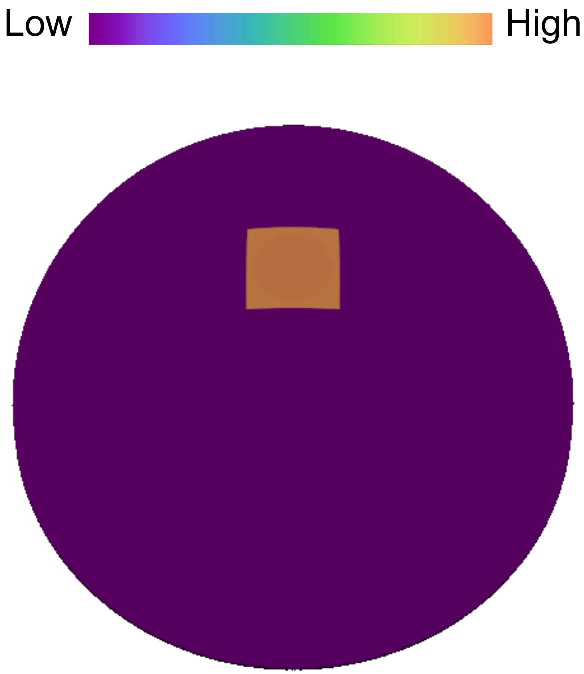
evolving domain

Differentiating the hemispherical integral

π: size of the emitter



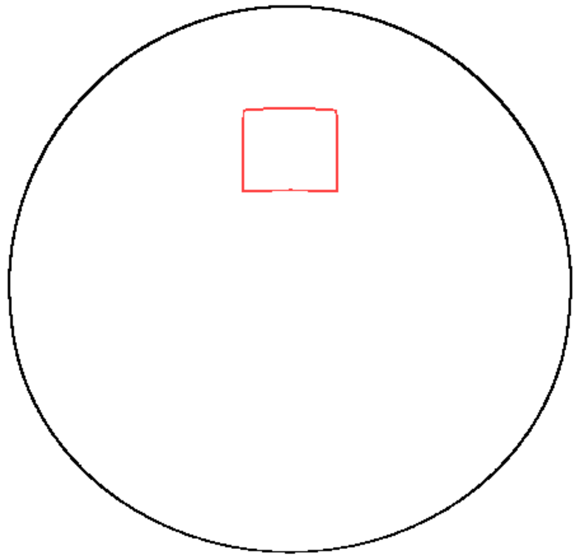
$$I = \int_{\mathbb{H}^2} f(\omega_i)\,\mathrm{d}\sigma(\omega_i)$$



Differentiation

Reynolds transport theorem

Discontinuities of f

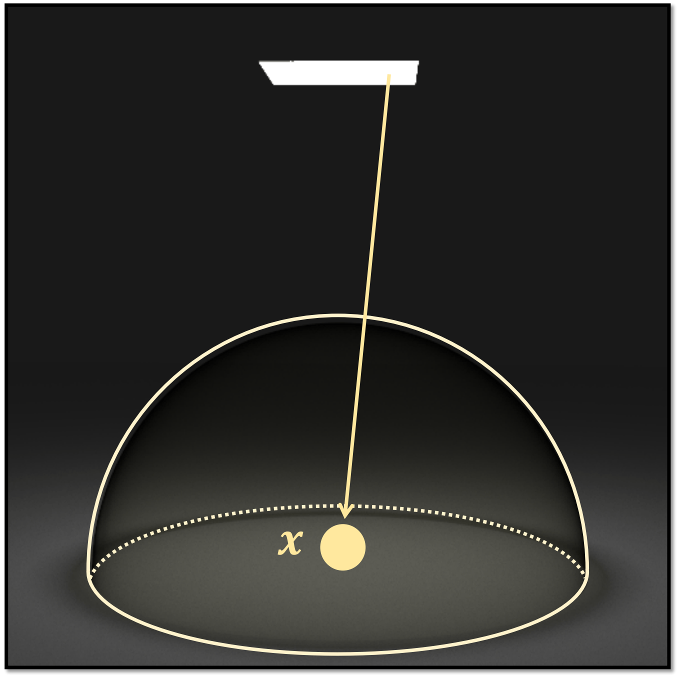


$$\frac{\mathrm{d}I}{\mathrm{d}\pi} = \int_{\mathbb{H}^2} \frac{\mathrm{d}(f)}{\mathrm{d}\pi}\,\mathrm{d}\sigma + \int_{\partial\mathbb{H}^2} g\,\mathrm{d}l$$

Interior Boundary

Differentiating the area integral

π: size of the emitter



$$I = \int_{\mathcal{L}(\pi)} f(y \to x) G(x, y) \mathrm{d}A(y)$$

Low High

Differentiation

Reynolds transport theorem

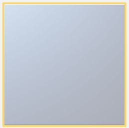
Boundary of $\mathcal{L}(\pi)$

$$\frac{\mathrm{d}I}{\mathrm{d}\pi} = \int_{\mathcal{L}(\pi)} \frac{\mathrm{d}(fG)}{\mathrm{d}\pi} \mathrm{d}A + \int_{\partial\mathcal{L}(\pi)} g \, \mathrm{d}l$$

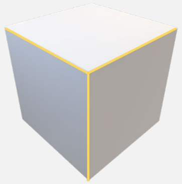
Interior Boundary

Sources of discontinuities

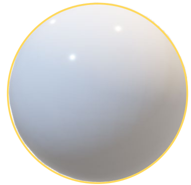
Boundary edge

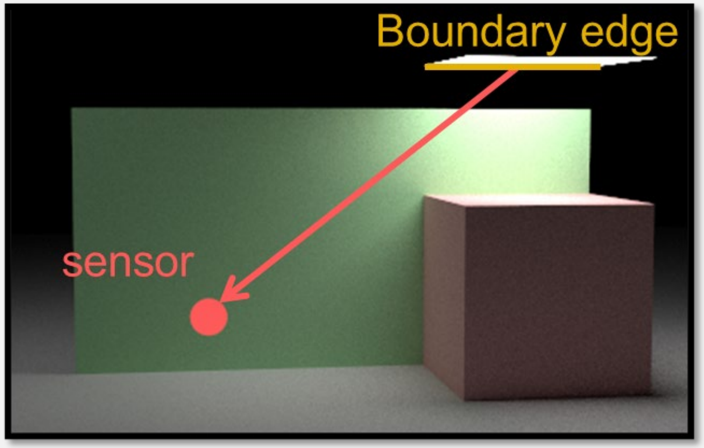


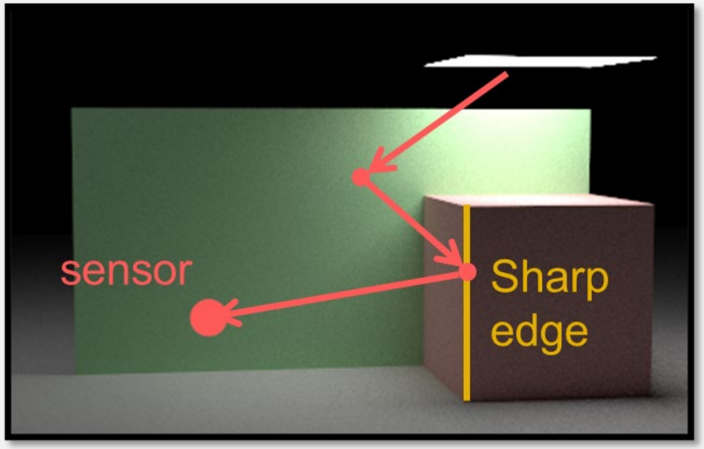
Sharp edge

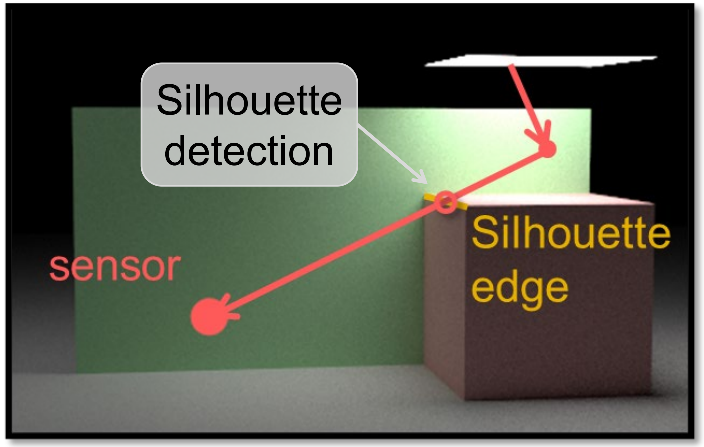


Silhouette edge





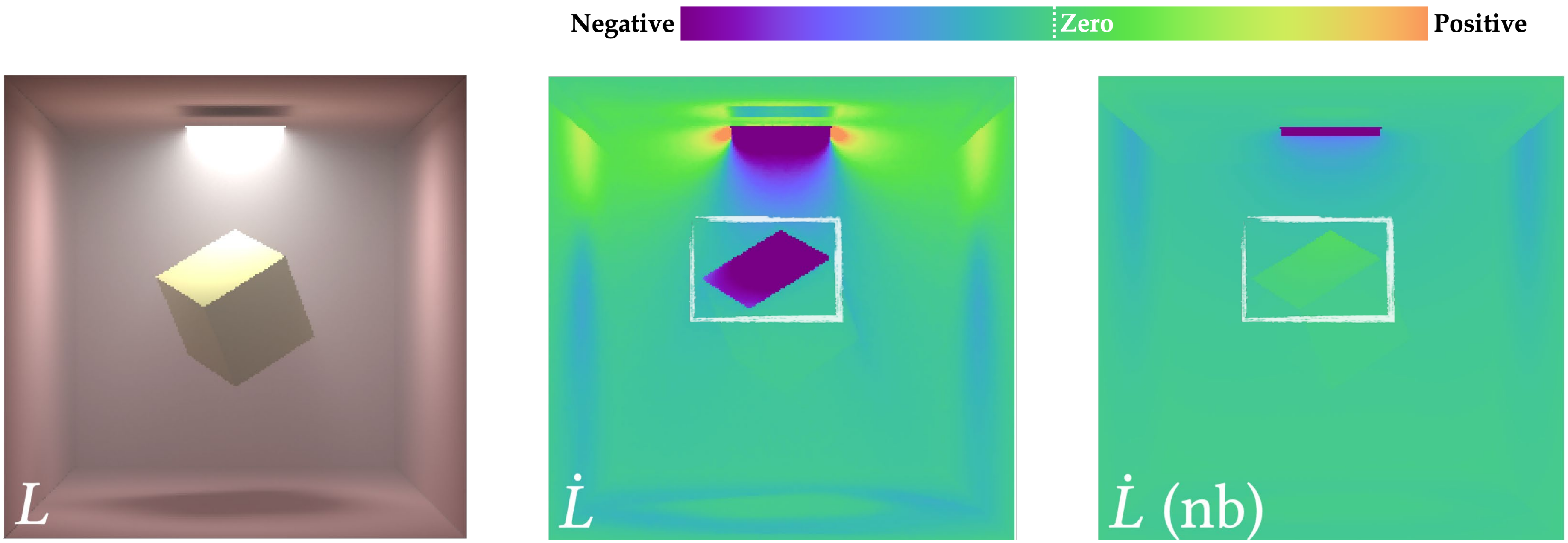




Topology-driven

Visibility-driven

Significance of the boundary integral



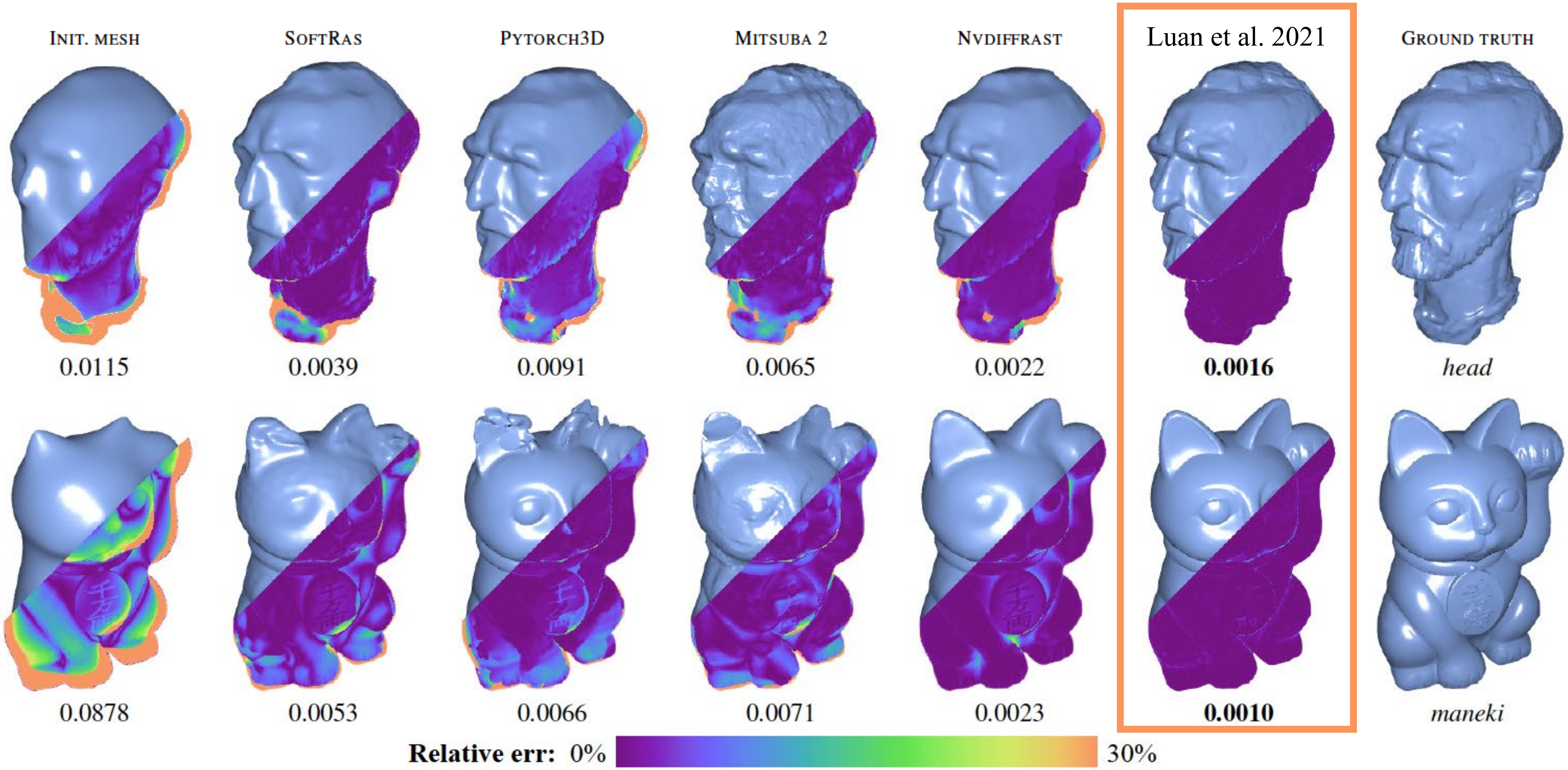
Original image

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

Derivative image
w/o boundary integral

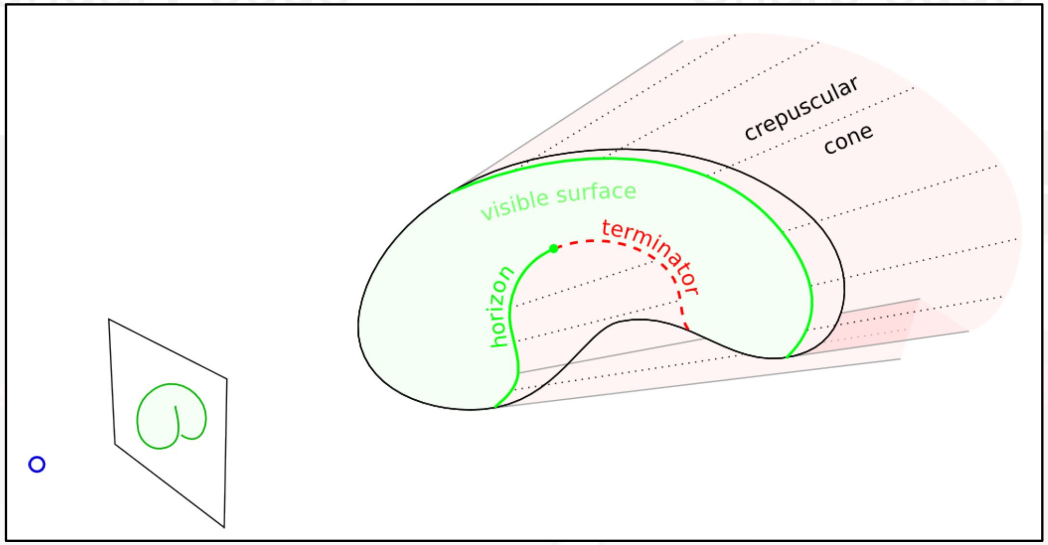
Gradient Accuracy Matters

Inverse-rendering results with *identical* optimization settings

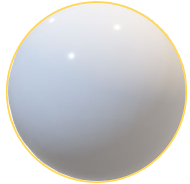


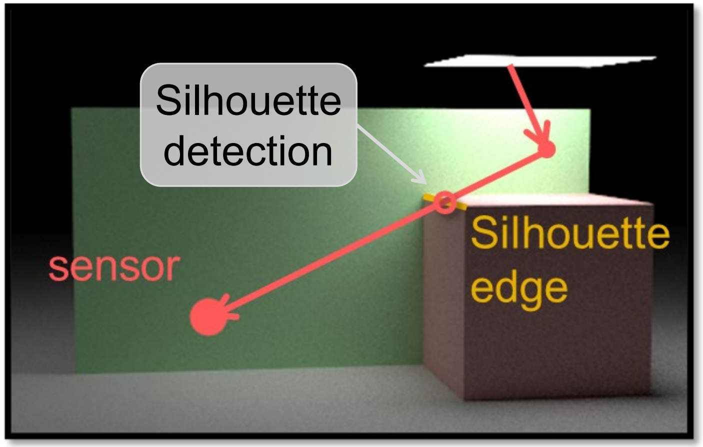
Sources of discontinuities

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



[Gargallo et al., ICCV 2007]

Silhouette edge 



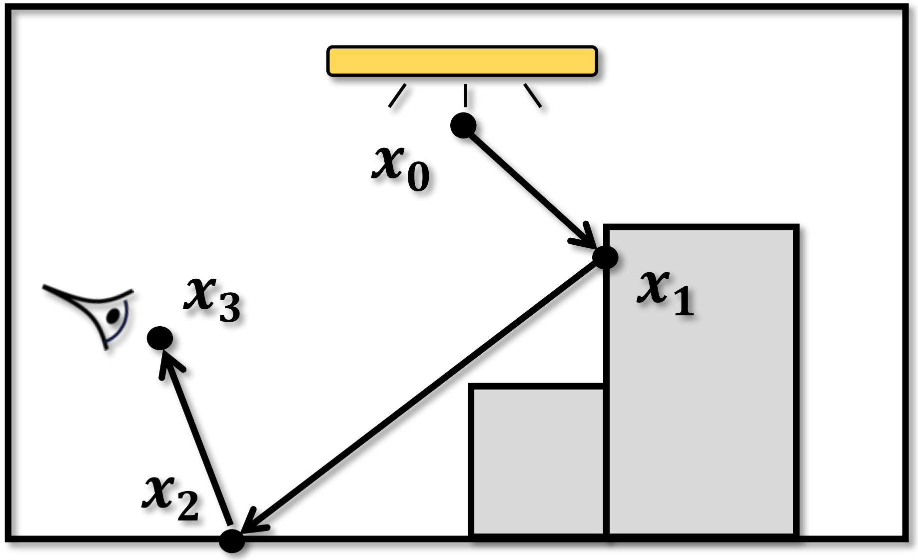
Visibility-driven

Handling Global Illumination

Background: Path Integral for Global Illumination

$$\underbrace{I}_{\text{Pixel value}} = \int_{\underbrace{\Omega}_{\text{Path space}}} \overbrace{f(\bar{\boldsymbol{x}})}^{\text{Measurement contribution}} \, \mathrm{d} \underbrace{\mu(\bar{\boldsymbol{x}})}_{\text{Area-product measure}}$$

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

Background: Estimating Path Integrals

Measurement contribution

Pixel value

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

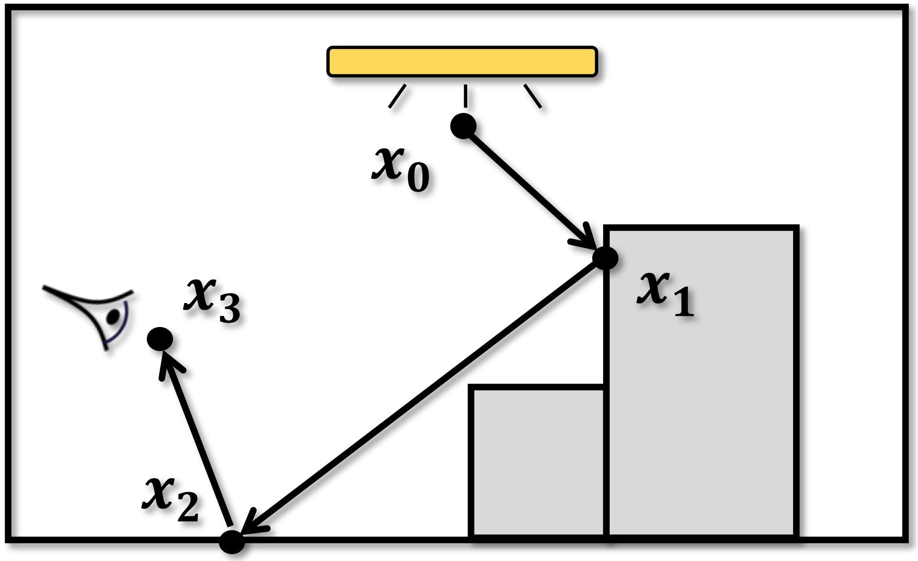
Path space

Area-product measure

Monte Carlo estimator:

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

Probability density for sampling path $\bar{\boldsymbol{x}}$



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

Differential Path Integral

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

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

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

$$h_n^{(0)} := \left[\prod_{n'=n+1}^{N} g(\boldsymbol{x}_{n'};\boldsymbol{x}_{n'-2},\boldsymbol{x}_{n'-1})\right] W_{\mathrm{e}}(\boldsymbol{x}_N \to \boldsymbol{x}_{N-1}), \tag{52}$$

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

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

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

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

$$h_n(\boldsymbol{x}_n;\boldsymbol{x}_{n-1}) = \int_{\mathcal{M}^{N-n}} h_n^{(0)} \prod_{n'=n+1}^{N} \mathrm{d}A(\boldsymbol{x}_{n'}), \tag{55}$$

and

$$\begin{aligned}\dot{h}_n(\boldsymbol{x}_n;\boldsymbol{x}_{n-1}) &= \int_{\mathcal{M}^{N-n}} \left[\left(h_n^{(0)}\right)^{\boldsymbol{\cdot}} - h_n^{(0)} h_n^{(1)}\right] \prod_{n'=n+1}^{N} \mathrm{d}A(\boldsymbol{x}_{n'}) \\ &+ \sum_{n'=n+1}^{N} \int \Delta h_{n,n'}^{(0)}\, V_{\overline{\partial\mathcal{M}}_{n'}}(\boldsymbol{x}_{n'})\,\mathrm{d}\ell(\boldsymbol{x}_{n'}) \prod_{\substack{n<i\le N\\ i\ne n'}} \mathrm{d}A(\boldsymbol{x}_i),\end{aligned} \tag{56}$$

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 $\boldsymbol{x}_i$ with $i \ne n'$ and $\overline{\partial\mathcal{M}}_{n'}(\pi)$, which depends on $\boldsymbol{x}_{n'-1}$, for $\boldsymbol{x}_{n'}$.

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 $g_{n-1} := g(\boldsymbol{x}_n;\boldsymbol{x}_{n-2},\boldsymbol{x}_{n-1})$ for all $0 < n \le N$. Then,

$$\begin{aligned}h_{n-1}(\boldsymbol{x}_{n-1};\boldsymbol{x}_{n-2}) &= \int_{\mathcal{M}} g_{n-1} \int_{\mathcal{M}^{N-n}} h_n^{(0)} \prod_{n'=n+1}^{N} \mathrm{d}A(\boldsymbol{x}_{n'})\,\mathrm{d}A(\boldsymbol{x}_n) \\ &= \int_{\mathcal{M}^{N-n+1}} h_{n-1}^{(0)} \prod_{n'=n}^{N} \mathrm{d}A(\boldsymbol{x}_{n'}),\end{aligned} \tag{57}$$

and

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

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

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

$$\begin{aligned}\dot{h}_0(\boldsymbol{x}_0) &= \int_{\mathcal{M}^N} \left[\dot{f}(\bar{\boldsymbol{x}}) - f(\bar{\boldsymbol{x}}) \sum_{n'=1}^{N} \kappa(\boldsymbol{x}_{n'})\,V(\boldsymbol{x}_{n'})\right] \prod_{n'=1}^{N} \mathrm{d}A(\boldsymbol{x}_{n'}) \\ &+ \sum_{n'=1}^{N} \int \Delta f_{n'}(\bar{\boldsymbol{x}})\,V_{\overline{\partial\mathcal{M}}_{n'}}\,\mathrm{d}\ell(\boldsymbol{x}_{n'}) \prod_{\substack{0<i\le N\\ i\ne n'}} \mathrm{d}A(\boldsymbol{x}_i).\end{aligned} \tag{59}$$

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

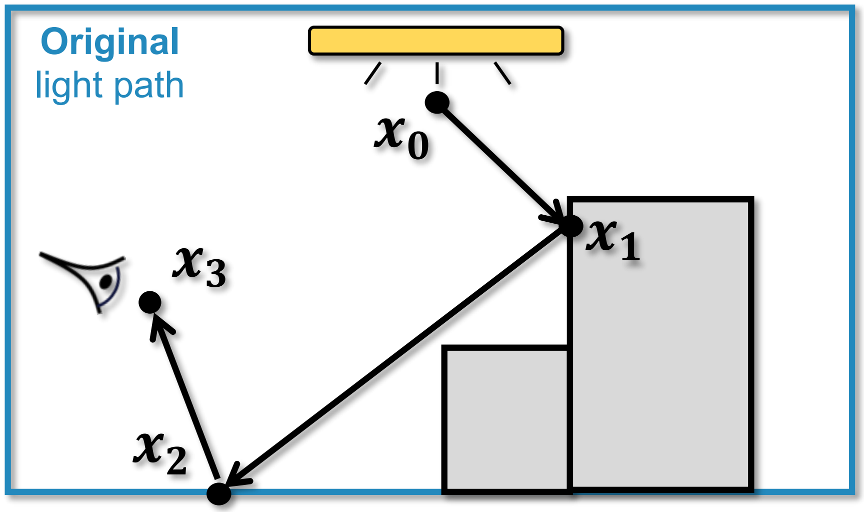
$$\begin{aligned}\frac{\partial I_N}{\partial\pi} &= \frac{\partial}{\partial\pi}\int_{\mathcal{M}} h_0(\boldsymbol{x}_0)\,\mathrm{d}A(\boldsymbol{x}_0) \\ &= \int_{\mathcal{M}} \left[\dot{h}_0(\boldsymbol{x}_0) - h_0(\boldsymbol{x}_0)\,\kappa(\boldsymbol{x}_0)\,V(\boldsymbol{x}_0)\right] \mathrm{d}A(\boldsymbol{x}_0)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

(The full derivation is quite involved…)

Differential Path Integral

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

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_\Omega f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right) = \underbrace{\int_\Omega \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})}_{\text{Interior integral}} + \int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})$$



Interior integral

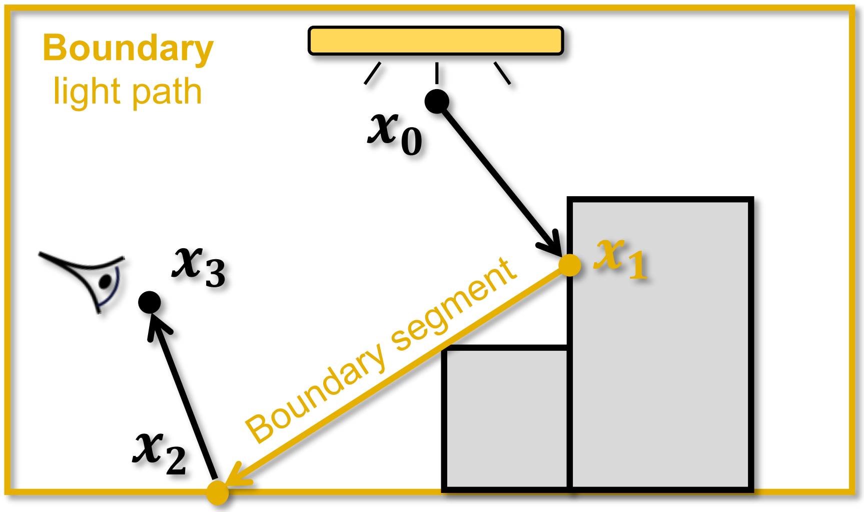
- Defined on the ordinary path space Ω
- The integrand $\dot{f}$ can be obtained by differentiating the ordinary *measurement contribution function* f

Differential Path Integral

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

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

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]

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_\Omega f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right) = \underbrace{\int_\Omega \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})}_{\text{Interior integral}} + \underbrace{\int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})}_{\text{Boundary integral}}$$

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]

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_\Omega f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right) = \underbrace{\int_\Omega \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})}_{\text{Interior integral}} + \int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})$$

- 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]

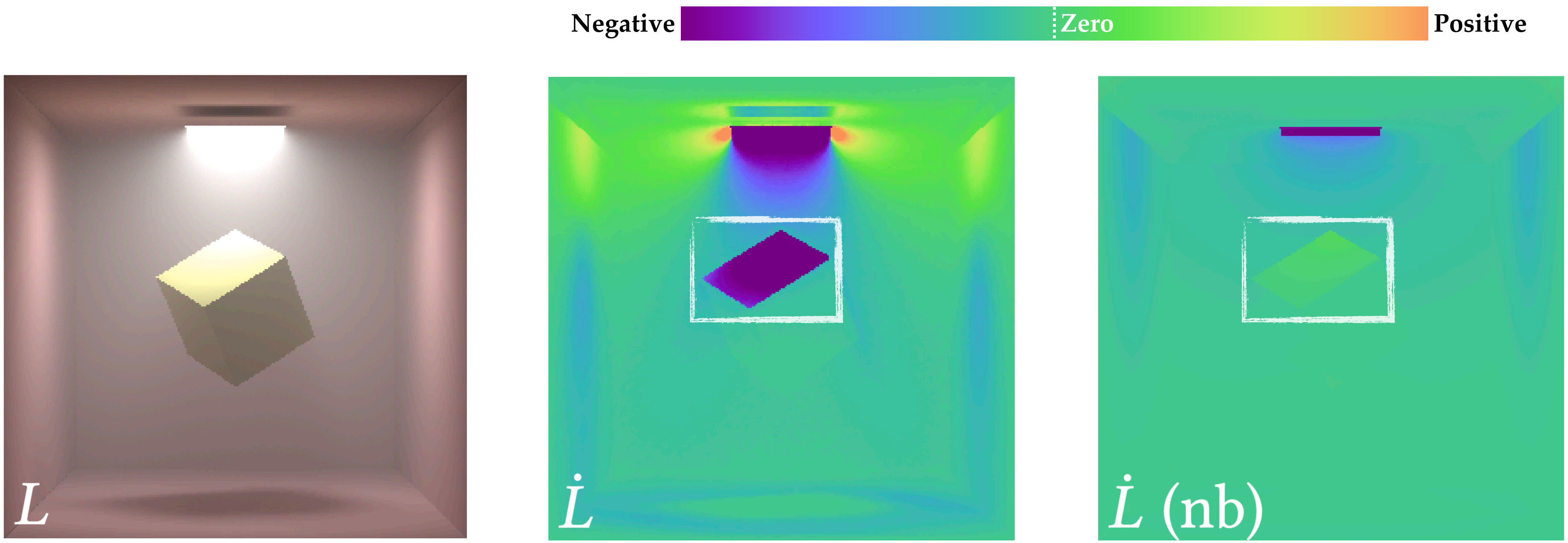
$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_{\Omega} f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right) = \underbrace{\int_{\Omega} \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})}_{\text{Interior integral}} + \underbrace{\int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})}_{\text{Boundary integral}}$$

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



Original image

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

Derivative image
w/o boundary integral

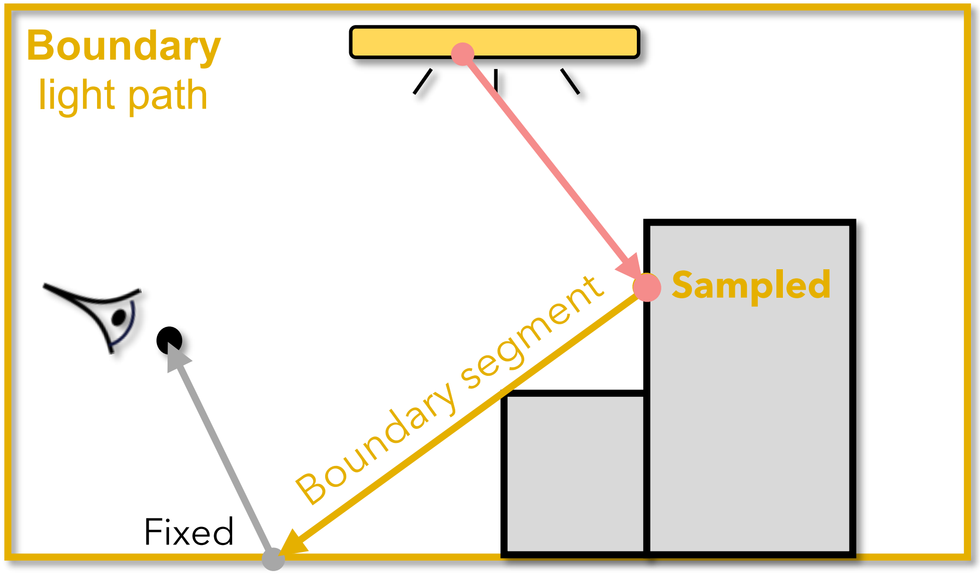
Handling Discontinuities

- **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]

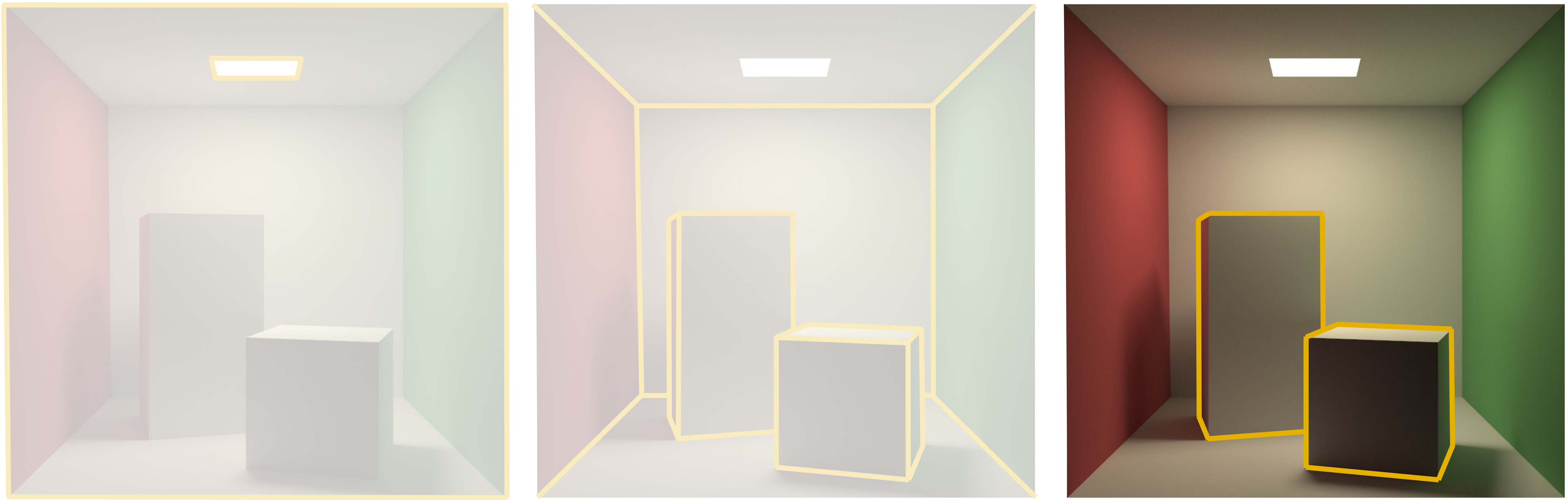
To sample a **boundary** segment:

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



Recap: Sources of Discontinuities

Boundary edges | **Sharp** edges | **Silhouette** edges



(Topological) boundary of an object

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

View-dependent object silhouettes

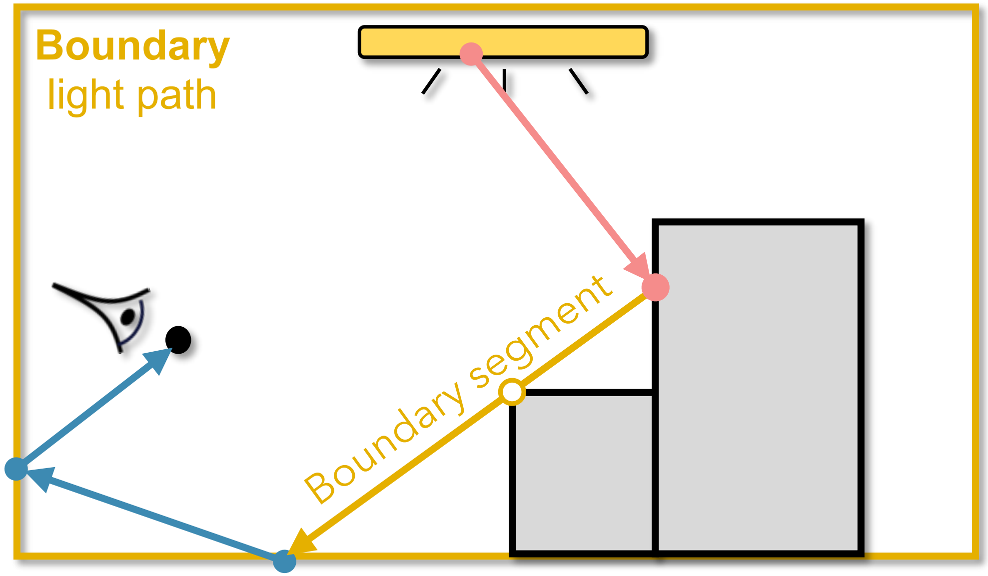
Handling Discontinuities

- **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)

To be discussed next

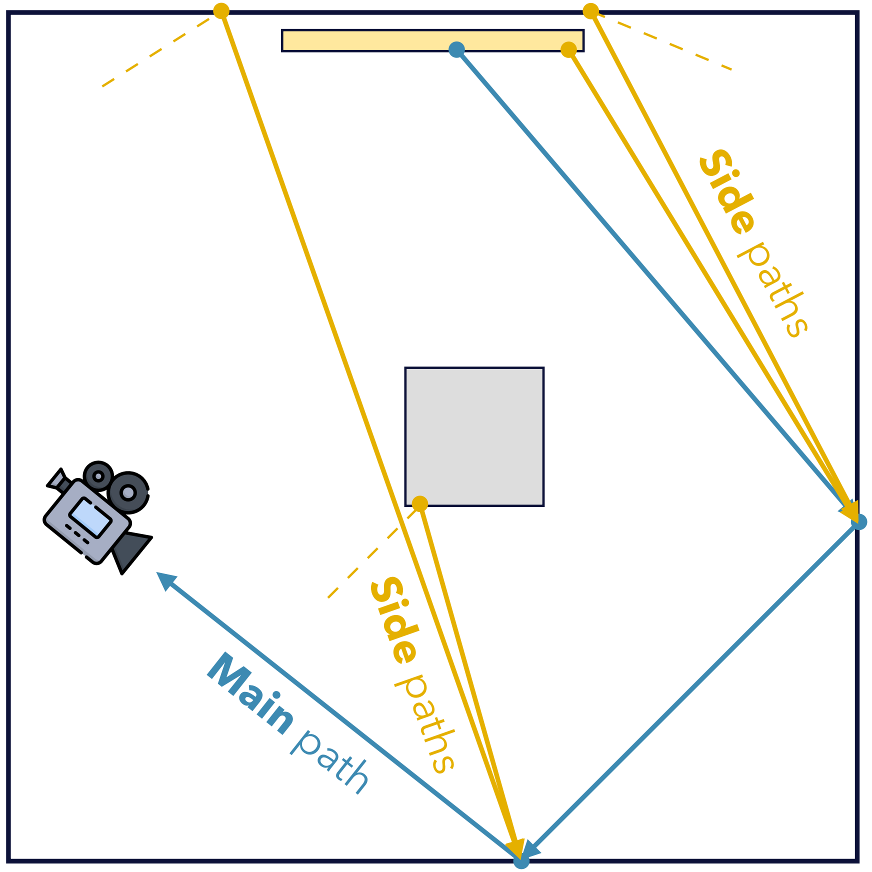
- **Area-sampling differentiable** rendering
  - Avoids boundary integrals altogether (Sai will cover this later)

Differentiable Path Tracing with Edge Sampling

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_{\Omega} f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right)$$

$$= \int_{\Omega} \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}}) + \int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})$$

Interior integral **Boundary** integral

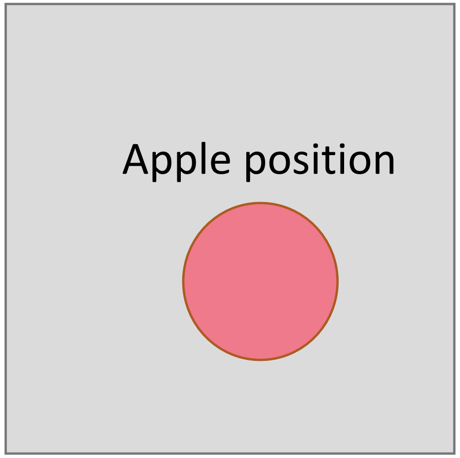


Differentiable path tracing with **edge sampling**

- 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)

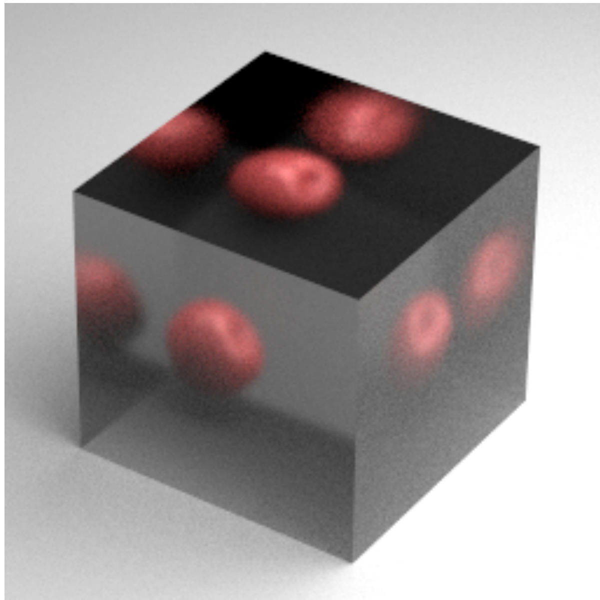
Inverse-Rendering Result [Zhang et al. 2019]

Parameters

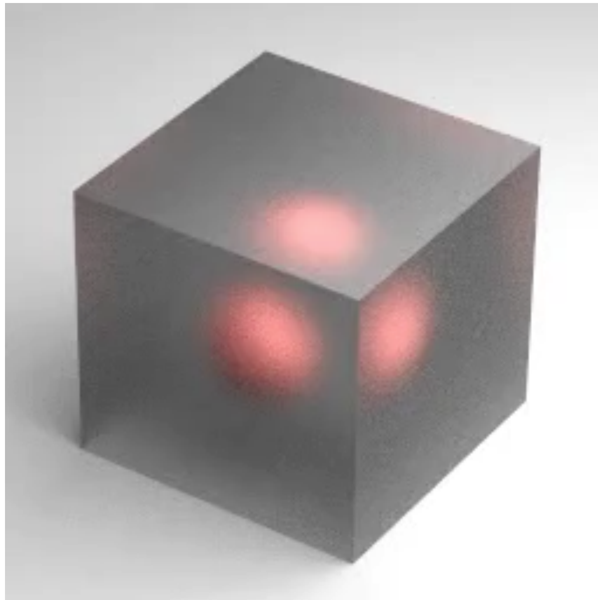


Cube roughness

Target



Optimization process

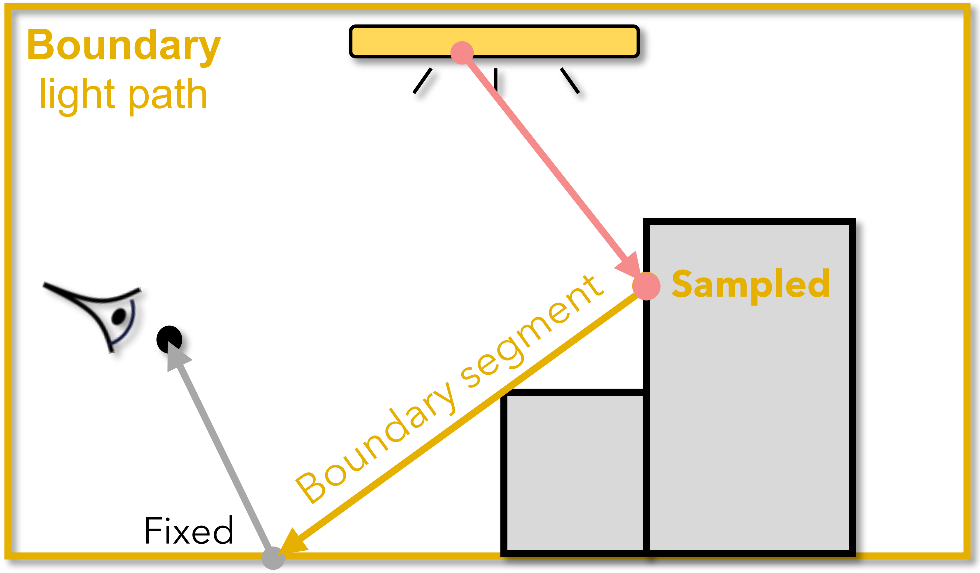


Light-transport phenomena: rough reflection and refraction
subsurface scattering

Differentiable Path Tracing with Edge Sampling

To sample a **boundary** segment:

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

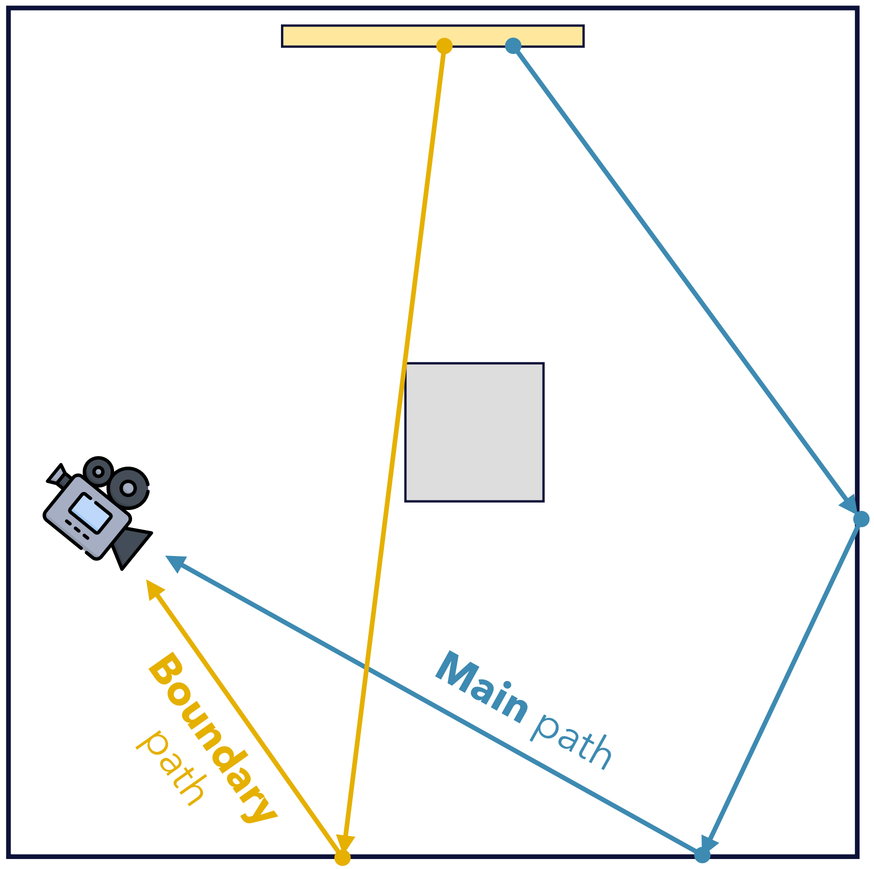


Requires **silhouette detection**, which can be **expensive**!

Path-Space Differentiable Path Tracing

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_{\Omega} f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})\right)$$

$$= \underbrace{\int_{\Omega} \dot{f}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}})}_{\text{Interior integral}} + \underbrace{\int_{\partial\Omega} g(\bar{\boldsymbol{x}})\,\mathrm{d}\mu'(\bar{\boldsymbol{x}})}_{\text{Boundary integral}}$$



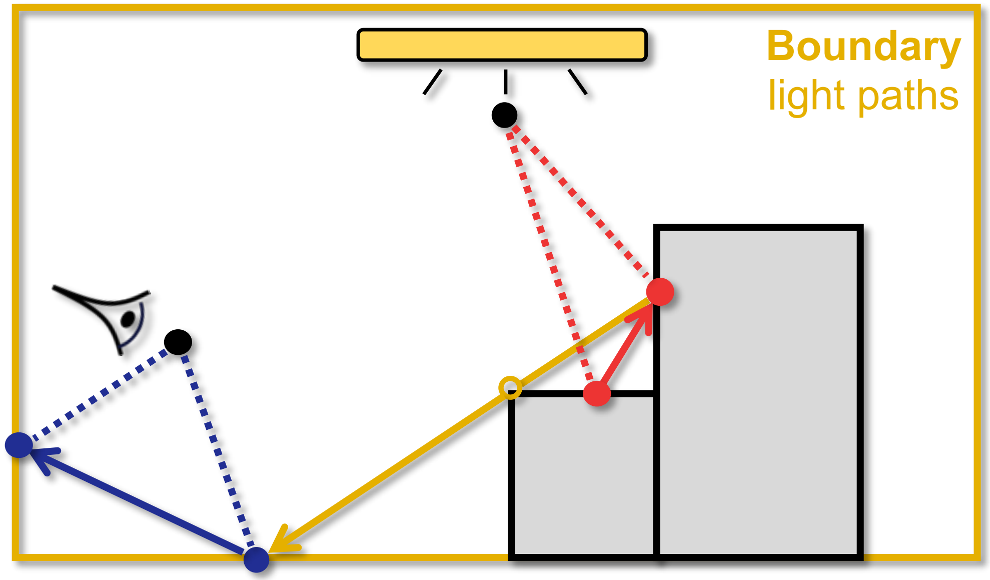
Path-space **differentiable** path tracing

- 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

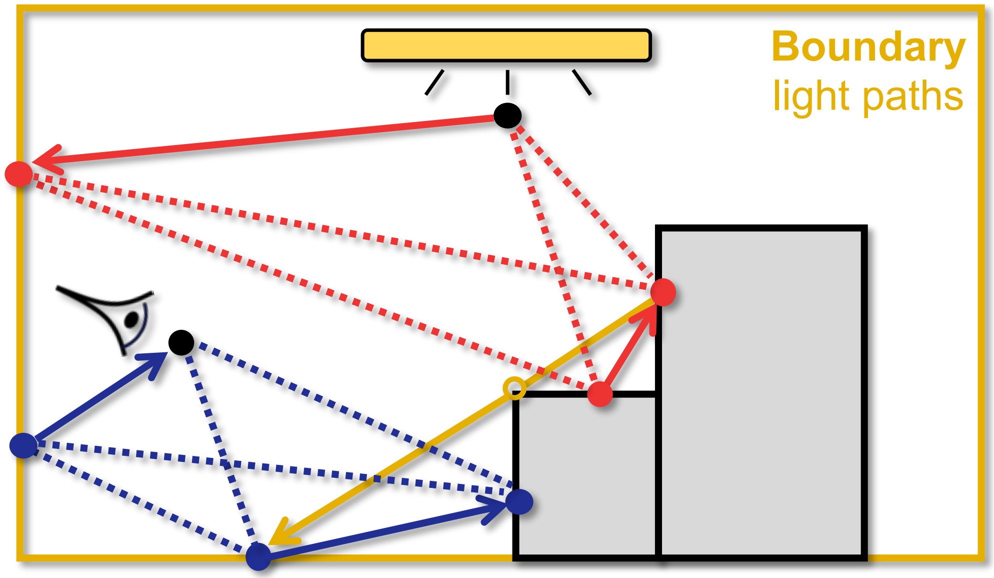
- **Interior**: *unidirectional* path tracing
- **Boundary**: *unidirectional* sampling of subpaths



Unidirectional path tracing + NEE

Bidirectional estimator

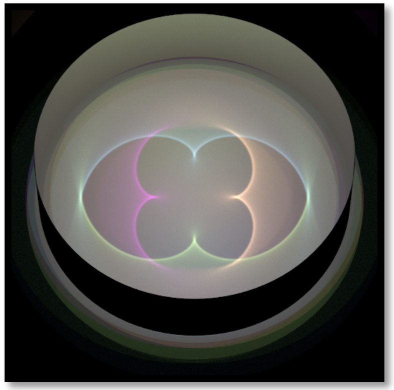
- **Interior**: *bidirectional* path tracing
- **Boundary**: *bidirectional* sampling of subpaths



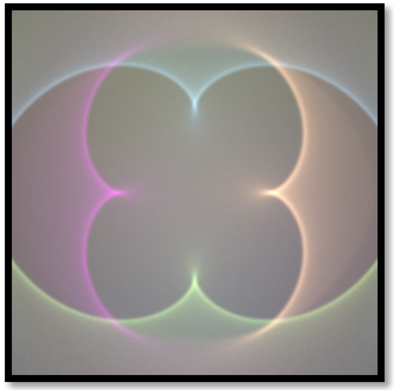
Bidirectional path tracing

Inverse-Rendering Result [Zhang et al. 2020]

Config.



Initial



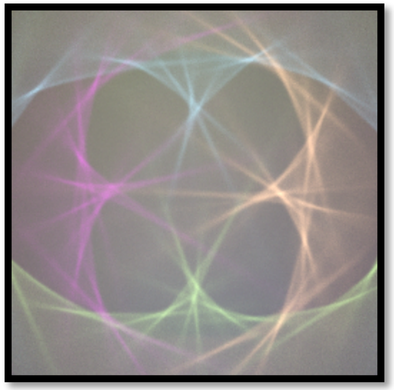
Scene configuration:

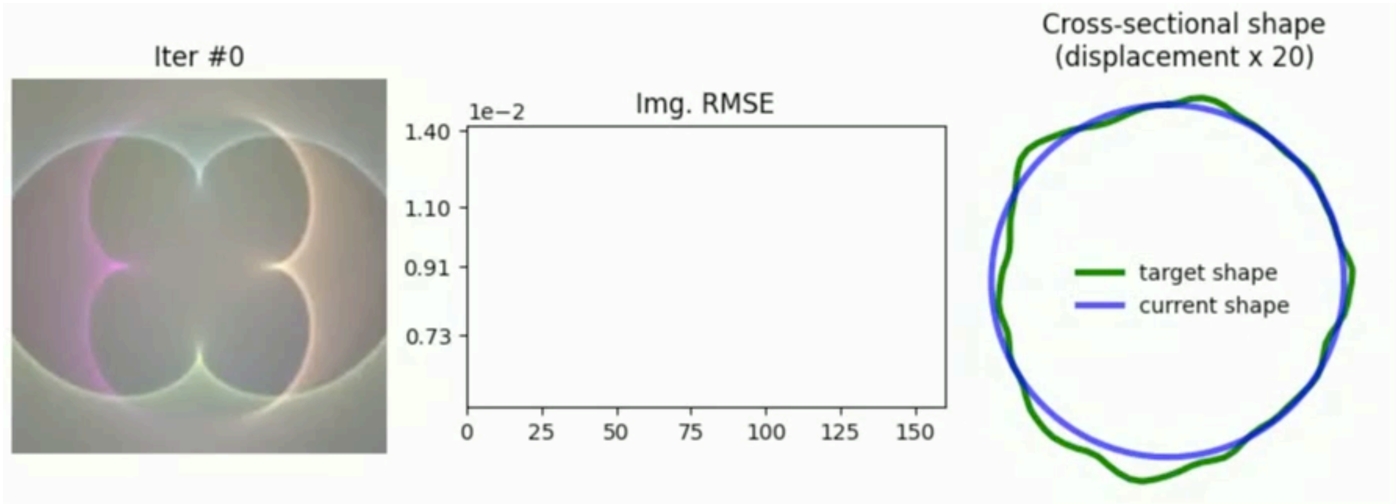
- A glossy ring lit by four colored light sources
- Optimize **cross-sectional shape** of the ring

Light-transport phenomenon:

- Caustics

Target

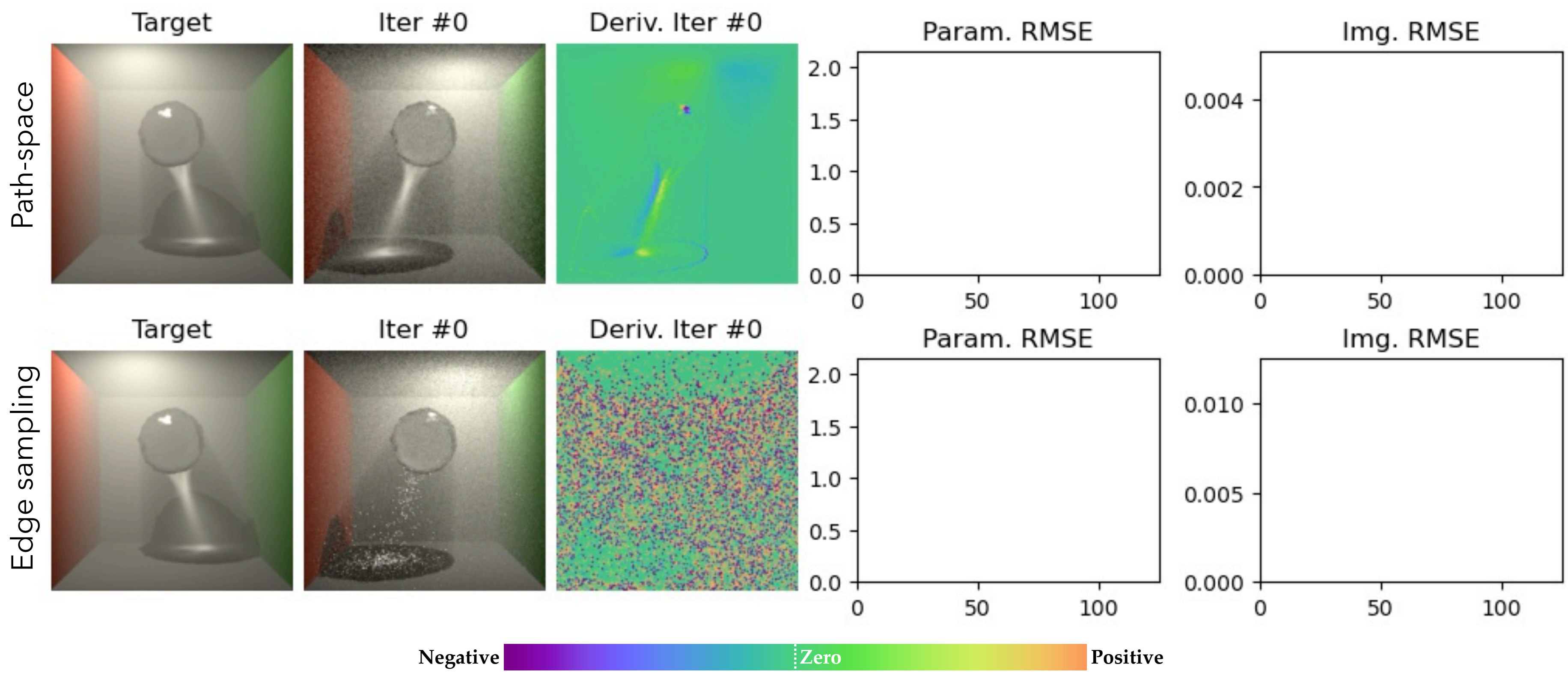




Inverse-Rendering Comparison [Zhang et al. 2021]

Optimizing the position of a small area light

(identical inverse-rendering configurations, equal-time per iteration)



Inverse-Rendering Result [Zhang et al. 2021]

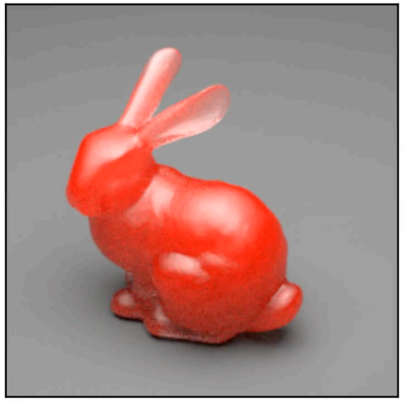
Initial

Target

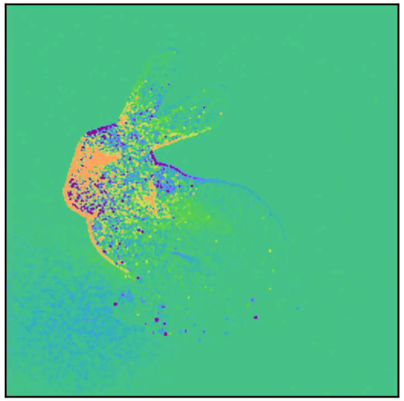
Jointly optimizing of the bunny's:

- Shape
- Surface roughness
- Optical density

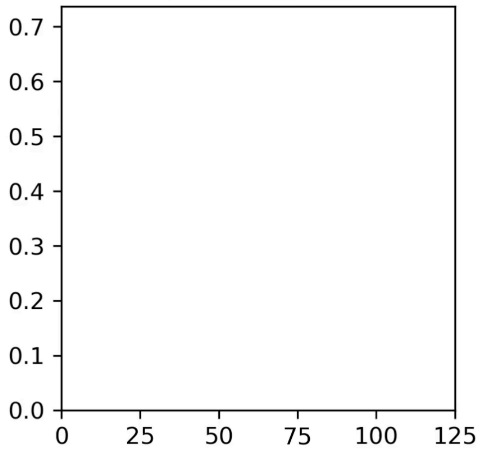
Iter #0



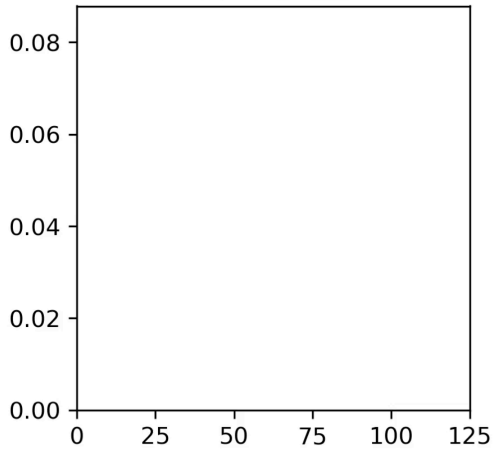
Deriv. Iter #0



Param. RMSE



Img. RMSE



Negative

Positive