Differentiable Rendering

Input parameters
shape & position of objects, materials, light sources, camera pose, etc.

Update scene

Differentiable physical simulation
\[ y = f(x) \]
\[ \frac{\partial y}{\partial x} = \frac{\partial}{\partial x} f(x) \]

Output rendered image

Differentiable objective function
\[ z = g(y) \]
\[ \frac{\partial z}{\partial y} = \frac{\partial}{\partial y} g(y) \]

Gurprit Singh
PHYSICS-BASED DIFFERENTIABLE RENDERING
A COMPREHENSIVE INTRODUCTION

Shuang Zhao
Wenzel Jakob
Tzu-Mao Li

University of California, Irvine
EPFL, Lausanne, Switzerland
MIT CSAIL, Cambridge
FORWARD VS. INVERSE RENDERING

Geometry, materials, emitters, ...

Scene: “bed classic” from jiraniano
FORWARD VS. INVERSE RENDERING

Geometry, materials, emitters, ...

Scene: “bed classic” from jiraniano
FORWARD VS. INVERSE RENDERING

Scene: "bed classic" from jiraniano

Geometry, materials, emitters, ...

$x \rightarrow y = f(x)$
FORWARD VS. INVERSE RENDERING

Geometry, materials, emitters, ...

\[ y = f(x) \]

Inverse rendering

\[ x = f^{-1}(y) \]

Scene: "bed classic" from jiraniano
INVERSE RENDERING IN COMPUTER VISION

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

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

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


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

BlockGAN: Learning 3D Object-aware Scene Representations from Unlabelled Images [Nguyen-Phuoc et al. 2020]
Focus on inverse rendering for realistic functions $f(x)$

*Global illumination, complex materials, participating media, polarization, color spectra, etc.*
- Focus on inverse rendering for realistic functions $f(x)$

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

- No neural networks.

Shouldn’t need them, we understand the underlying equations. (Of course still possible to use neural nets inside or outside of the renderer)
SHAPE & MATERIAL RECONSTRUCTION

<table>
<thead>
<tr>
<th>Target</th>
<th>Target</th>
<th>Target</th>
<th>Target</th>
</tr>
</thead>
</table>

Reparameterizing discontinuous integrands for differentiable rendering [Loubet et al. 2019]
Reparameterizing discontinuous integrands for differentiable rendering [Loubet et al. 2019]
SHAPE & MATERIAL RECONSTRUCTION

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

Optimized density voxels
Multi-view setup
Uniform illumination
View
Optimized albedo voxels
Optimized gradient-index lens
Directional area lights
Projected caustic

Schwartzburg et al. 2014

Optimized geometry
Directional area light

Schwartzburg et al. 2014

Realistic Image Synthesis SS2024
(META-) MATERIAL DESIGN
(META-) MATERIAL DESIGN
(META-) MATERIAL DESIGN

Mitsuba 2: A Retargetable Forward and Inverse Renderer [Nimier-David et al. 2019]
(META-) MATERIAL DESIGN
FABRICATION: 3D PRINT OPTIMIZATION

Elek et al. 2017

Target
Naive print

Uniform illumination

View

Optimized albedo voxels

Optimized density voxels

Multi-view setup

Uniform illumination

View

Optimized albedo voxels

Optimized geometry

Directional area light

Projected caustic

Optimized gradient-index lens

Directional area lights

Caustic design: surface displacement

Caustic design: gradient-index optics

Volume density reconstruction

Textured translucent slab

Elek et al. 2017

Naive print

Target
Reference: diffuse surface texture
Reference: diffuse surface texture
WHY DIFFERENTIABLE RENDERING

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

- Inverse subsurface scattering [Che et al. 2020]

![Diagram of differentiable rendering pipeline](image)

**Figure 1:** Overview of our pipeline at training and test time.
WHY DIFFERENTIABLE RENDERING

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

• Inverse subsurface scattering [Che et al. 2020]

![Diagram of the differentiable rendering process](image)

- Utilizing *image loss* (provided by a volume path tracer) to regularize training
WHY DIFFERENTIABLE RENDERING

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

- Inverse subsurface scattering [Che et al. 2020]

![Diagram showing the process of differentiable rendering]

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

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

irradiance gradient [Ward 1992]
path differentials [Suykens and Williams 2001]
H2MC [Li et al. 2015]
Langevin MC [Luan et al. 2020]
Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.
Many disciplines rely on understanding or controlling the behavior of light in images or other kinds of measurements.
Current rendering
Target
OBJECTIVE FUNCTION (A.K.A. “LOSS”)

\[ g(\text{Rendering}) = \| g(\text{Target}) \|^2 \]
OBJECTIVE FUNCTION (A.K.A. “LOSS”)  

\[ g(x) = \text{Rendering} - \text{Target} \]

The problem: \( \min_{x \in \mathcal{X}} g(f(x)) \)
The problem: \( \min_{x \in \mathcal{X}} g(f(x)) \)
The problem: \( \min_{x \in \mathcal{X}} g(f(x)) \)

- meshes
- material (BSDF) parameters
  - textures, etc.
- parameters of procedural models
- volumes, light sources, …
The problem: $\minimize_{x \in \mathcal{X}} g(f(x))$
The problem: minimize \( g(f(x)) \)

\[ x \in \mathcal{X} \]
The problem: \[
\text{minimize } g(f(x))
\]
\[\forall x \in \mathcal{X}\]
DIFFERENTIABLE RENDERING

\[ f(x) \rightarrow y \]

\[ g(y, ...) \rightarrow z \]

\[ \frac{\partial z}{\partial x} \]

\[ X \]

\[ Y \]

\[ Z \]
$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}$

**Chain Rule**

**Introduction**
DIFFERENTIABLE RENDERING

\[ \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \]

**CHAIN RULE**
DIFFERENTIABLE RENDERING

Vector

\[ \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \]

CHAIN RULE

\[ \frac{\partial z}{\partial y} \]
DIFFERENTIABLE RENDERING

Vector

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}
\]

Matrix

CHAIN RULE

\[
f(x) \rightarrow y \rightarrow g(y, \ldots) \rightarrow z
\]
**DIFFERENTIABLE RENDERING**

**Vector**

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}
\]

**Matrix**

**CHAIN RULE**

\[
f(x) \rightarrow g(y, \ldots) \rightarrow z
\]

\[
\frac{\partial y}{\partial x}
\]

\[
\frac{\partial z}{\partial y}
\]
DIFFERENTIABLE RENDERING

\[ \frac{\partial y}{\partial x} \]

\[ X \]
DIFFERENTIABLE RENDERING

\[ \frac{\partial y}{\partial x} \]

\[ \frac{\partial z}{\partial y} \]
INTRODUCTION
DIFFERENTIABLE RENDERING

\[ \frac{\partial y}{\partial x} \]

\[ \frac{\partial z}{\partial y} \]
DIFFERENTIABLE RENDERING

\[ \frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x} \]
DIFFERENTIABLE RENDERING

Challenges

1. Differentiating $f$
2. Matrix multiplication
3. Efficiency?
4. How to deal with edges?
HOW TO DO THIS (AT ALL?)

Use finite differences!

\[
\frac{\partial y}{\partial x_i} = \frac{f(x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon}
\]
HOW TO DO THIS (AT ALL?)

Use finite differences!

\[
\frac{\partial y}{\partial x_i} = \frac{f(x + \epsilon e_i) - f(x - \epsilon e_i)}{2\epsilon}
\]

Potential problems:
- Bad approximation (big \( \epsilon \)), rounding error (small \( \epsilon \))
HOW TO DO THIS (AT ALL?)

Use finite differences!

\[
\frac{\partial y}{\partial x_i} = \frac{f(x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon}
\]

Potential problems:
- Bad approximation (big \(\varepsilon\)), rounding error (small \(\varepsilon\))
- Need to correlate Monte Carlo samples

[Wikipedia]
HOW TO DO THIS (AT ALL?)

Use finite differences!

\[
\frac{\partial y}{\partial x_i} = \frac{f(x + \varepsilon e_i) - f(x - \varepsilon e_i)}{2\varepsilon}
\]

Potential problems:

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

\[ f(x) \]
AUTOMATIC DIFFERENTIATION

\[ f(x) \xrightarrow{\text{AD}} \frac{\partial}{\partial x} f(x) \]
ISSUES WITH AUTOMATIC DIFFERENTIATION (AD)
ISSUES WITH AUTOMATIC DIFFERENTIATION (AD)
Precautions must be taken to ensure **correctness**
- Symbolically differentiating a Monte Carlo estimator path tracer does not always work!
Precautions must be taken to ensure correctness

Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

Example 1: Distributional parameters

Estimate \( \int_0^\infty f(\lambda, x) \, dx \) (with \( \lambda \) given)

(Single-sample) Monte Carlo estimator:

- Draw \( x \sim \text{Exp}[\lambda] \)
- \( f \leftarrow f(\lambda, x) \)
- \( p \leftarrow \lambda e^{-\lambda x} \)  # This is the pdf of \( \text{Exp}[\lambda] \)
- Return \( f/p \)
Precautions must be taken to ensure **correctness**

- Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

**Example 1: Distributional parameters**

Estimate $\int_{0}^{\infty} f(\lambda, x) \, dx$ (with $\lambda$ given)

(Single-sample) Monte Carlo estimator:
- Draw $x \sim \text{Exp}[\lambda]$
- $f \leftarrow f(\lambda, x)$
- $p \leftarrow \lambda e^{-\lambda x}$  # This is the pdf of $\text{Exp}[\lambda]$
- Return $f/p$

Estimate $\frac{d}{d\lambda} \int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{\infty} \frac{\partial f}{\partial \lambda} (\lambda, x) \, dx$

(Single-sample) Monte Carlo estimator:
- Draw $x \sim \text{Exp}[\lambda]$
- $f' \leftarrow \frac{\partial f}{\partial \lambda} (\lambda, x)$
- $p \leftarrow \lambda e^{-\lambda x}$
- Return $f'/p$
Precautions must be taken to ensure correctness

- Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

**Example 1:** Distributional parameters

Estimate \( \int_0^\infty f(\lambda, x) \, dx \) (with \( \lambda \) given)

\[
\text{(Single-sample) Monte Carlo estimator:}
\begin{align*}
&\text{Draw } x \sim \text{Exp}[\lambda] \\
&f \leftarrow f(\lambda, x) \\
p \leftarrow \lambda e^{-\lambda x} \quad \# \text{This is the pdf of } \text{Exp}[\lambda] \\
&\text{Return } f/p
\end{align*}
\]

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

\[
\text{(Single-sample) Monte Carlo estimator:}
\begin{align*}
&\text{Draw } x \sim \text{Exp}[\lambda] \quad x \text{ has zero gradient} \\
&f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda, x) \\
p \leftarrow \lambda e^{-\lambda x} \quad p \text{ is NOT differentiated} \\
&\text{Return } f'/p
\end{align*}
\]
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

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

Estimate $\int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{1} \frac{f(\lambda, x)}{\lambda \xi} \, d\xi$

(Single-sample) Monte Carlo estimator:

• Draw $\xi \sim U[0, 1]$
• $x \leftarrow -\log(\xi)/\lambda$  # $x \sim \text{Exp}(\lambda)$
• $f \leftarrow f(\lambda, x)$
• $p \leftarrow \lambda e^{-\lambda x}$  # $p = \lambda \xi$
• Return $f/p$
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

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

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

(Single-sample) Monte Carlo estimator:
  • Draw $\xi \sim U[0,1)$
  • $x \leftarrow -\log(\xi)/\lambda$  # $x \sim \text{Exp}(\lambda)$
  • $f \leftarrow f(\lambda, x)$
  • $p \leftarrow \lambda e^{-\lambda x}$  # $p = \lambda \xi$
  • Return $f/p$

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

(Single-sample) Monte Carlo estimator:
  • Draw $\xi \sim U[0,1)$
  • $x \leftarrow -\log(\xi)/\lambda$
  • $f \leftarrow f(\lambda, x)$
  • $p \leftarrow \lambda e^{-\lambda x}$  # $p = \lambda \xi$
  • Return $\partial (f/p)/\partial \lambda$
• Precautions must be taken to ensure correctness
  – Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

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

Estimate $\int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{1} \frac{f(\lambda, x)}{\lambda \xi} \, d\xi$

(Single-sample) Monte Carlo estimator:
  • Draw $\xi \sim U[0,1)$
  • $x \leftarrow -\log(\xi)/\lambda$  # $x \sim \text{Exp(}\lambda)$
  • $f \leftarrow f(\lambda, x)$
  • $p \leftarrow \lambda e^{-\lambda x}$  # $p = \lambda \xi$
  • Return $f/p$

Estimate $\frac{d}{d\lambda} \int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{1} \frac{\partial}{\partial \lambda} f(\lambda, x) \frac{1}{\lambda \xi} \, d\xi$

(Single-sample) Monte Carlo estimator:
  • Draw $\xi \sim U[0,1)$
  • $x \leftarrow -\log(\xi)/\lambda$  # $x$ has nonzero gradient
  • $f \leftarrow f(\lambda, x)$
  • $p \leftarrow \lambda e^{-\lambda x}$  # $p = \lambda \xi$
  • Return $\frac{\partial(f/p)}{\partial \lambda}$  # $f$ and $p$ are both differentiated
• Precautions must be taken to ensure **correctness**
  – Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

**Example 1:** Distributional parameters

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

(Single-sample) Monte Carlo estimator:
- **Draw** \( x \sim \text{Exp}[\lambda] \) \( x \) has zero gradient
- \( f' \leftarrow \frac{\partial f}{\partial \lambda}(\lambda, x) \)
- \( p \leftarrow \lambda e^{-\lambda x} \) \( p \) is NOT differentiated
- Return \( f' / p \)

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

(Single-sample) Monte Carlo estimator:
- **Draw** \( \xi \sim U[0,1] \)
- **Draw** \( x \leftarrow -\log(\xi) / \lambda \) \( x \) has nonzero gradient
- \( f \leftarrow f(\lambda, x) \)
- \( p \leftarrow \lambda e^{-\lambda x} \quad \# p = \lambda \xi \)
- Return \( \frac{\partial (f/p)}{\partial \lambda} \) \( f \) and \( p \) are both differentiated
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

• **Example 1:** Distributional parameters

Estimate \( \frac{d}{d\lambda} \int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{\infty} \frac{\partial f(\lambda, x)}{\partial \lambda} \, dx \)

(Single-sample) Monte Carlo estimator:
- **Draw** \( x \sim \text{Exp}[\lambda] \)
- \( f' \leftarrow \frac{\partial f(\lambda, x)}{\partial \lambda} \)
- \( p \leftarrow \lambda e^{-\lambda x} \)
- Return \( \frac{f'}{p} \)

\( x \) has zero gradient

\( p \) is NOT differentiated

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

Estimate \( \frac{d}{d\lambda} \int_{0}^{\infty} f(\lambda, x) \, dx = \int_{0}^{1} \frac{\partial f(\lambda, x)}{\partial \lambda} \, \lambda \xi \, d\xi \)

(Single-sample) Monte Carlo estimator:
- **Draw** \( \xi \sim U[0,1] \)
- \( x \leftarrow -\log(\xi)/\lambda \) \( x \) has nonzero gradient
- \( f \leftarrow f(\lambda, x) \)
- \( p \leftarrow \lambda e^{-\lambda x} \) \# \( p = \lambda \xi \)
- Return \( \frac{\partial (f/p)}{\partial \lambda} \)

\( f \) and \( p \) are both differentiated
• Precautions must be taken to ensure **correctness**
  – Symbolically differentiating a Monte Carlo estimator path tracer does not always work!

• **Example 2:** Discontinuities

Estimate \[ \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx \] with \( 0 < p < 1 \)

(Single-sample) Monte Carlo estimator:

• Draw \( X \sim U[0, 1] \)
• Return \( X < p \ ? \ 1 : 0.5 \)
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

• Example 2: Discontinuities

Estimate \( \int_0^1 (x < p ? 1 : 0.5) \, dx \) with \( 0 < p < 1 \)

(Single-sample) Monte Carlo estimator:
  • Draw \( X \sim U[0, 1) \)
  • Return \( X < p ? 1 : 0.5 \)

Ground-truth:
\[
\int_0^1 (x < p ? 1 : 0.5) \, dx = \int_0^p \, dx + \int_p^1 0.5 \, dx = \frac{1 + p}{2}
\]
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

• Example 2: Discontinuities

Estimate \[ \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx \] with \( 0 < p < 1 \)

(Single-sample) Monte Carlo estimator:
  • Draw \( X \sim U[0, 1) \)
  • Return \( X < p \ ? \ 1 : 0.5 \)

Ground-truth:
\[
\int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \int_0^p dx + \int_p^1 0.5 \, dx = \frac{1 + p}{2}
\]

Estimate \[ \frac{d}{dp} \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx \] with \( 0 < p < 1 \)

(Single-sample) Monte Carlo estimator:
  • Draw \( X \sim U[0, 1) \)
  • Return \( d(X < p \ ? \ 1 : 0.5)/dp \)

Ground-truth:
\[
\frac{d}{dp} \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \frac{d}{dp} \frac{1 + p}{2} = \frac{1}{2}
\]
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

**Example 2:** Discontinuities

Estimate \(\int_0^1 (x < p \ ? \ 1 : 0.5) \, dx\) with \(0 < p < 1\)

(Single-sample) Monte Carlo estimator:
- Draw \(X \sim U[0, 1]\)
- Return \(X < p \ ? \ 1 : 0.5\)

Ground-truth:
\[
\int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \int_0^p dx + \int_p^1 0.5 \, dx = \frac{1 + p}{2}
\]

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

(Single-sample) Monte Carlo estimator:
- Draw \(X \sim U[0, 1]\)
- Return \(d(X < p \ ? \ 1 : 0.5)/dp\) Zero! (constant)

Ground-truth:
\[
\frac{d}{dp} \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \frac{d}{dp} \frac{1 + p}{2} = \frac{1}{2}
\]
WHY IS DIFFERENTIABLE RENDERING DIFFICULT

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

• **Example 2:** Discontinuities

Estimate \( \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx \) with \( 0 < p < 1 \)

(Single-sample) Monte Carlo estimator:
- Draw \( X \sim U[0, 1] \)
- Return \( X < p \ ? \ 1 : 0.5 \)

Ground-truth:
\[
\int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \int_0^p dx + \int_p^1 0.5 \, dx = \frac{1 + p}{2}
\]

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

(Single-sample) Monte Carlo estimator:
- Draw \( X \sim U[0, 1] \)
- Return \( \frac{d}{dp}(X < p \ ? \ 1 : 0.5)/dp \) Zero! (constant)

Ground-truth:
\[
\frac{d}{dp} \int_0^1 (x < p \ ? \ 1 : 0.5) \, dx = \frac{d}{dp} \frac{1 + p}{2} = \frac{1}{2}
\]

More on this example later
Basics
COURSE OUTLINE

Basics

State-of-the-art theories and algorithms
COURSE OUTLINE

Introduction

Basics

State-of-the-art theories and algorithms

Implementation details
BASICS
DIFFERENTIATING (RENDERING) PROGRAMS

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

```cpp
auto scatter_contribution = Vector3(0, 0, 0);
auto scatter_pdf = Vector3(0, 0, 0);
if (bsdf_isect_valid()) {
    const auto bsdf_shape = scene.shapes[bsdf_isect.shape_id];
    auto dir = bsdf_point_position - p;
    auto dist_sq = length_sqaured(dir);
    auto wo = dir / sqrt(dist_sq);
    auto pdf_pdf = bsdf_pdf(material, shading_point, wi, wo, min_depth);
    if (dist_sq > 1e-20f && pdf_pdf > 1e-20f) {
        auto bsdf_val = bsdf(material, shading_point, wi, wo, min_depth);
        if (bsdf_shape.light_id == 0) {
            const auto &light = scene.area_lights[bsdf_shape.light_id];
            if (light_two_sided || dot(-wo, bsdf_shape.shading_direction) >
                auto light_contribution = light.intensity;
                auto light_pdf = scene.light_pdf[bsdf_shape.light_id];
                auto light_area = scene.light_areas[bsdf_shape.light_id];
                auto light_area = 1 / light_area;
                auto geometry_term = light_pdf * light_area / geometry_term;
                auto mis_weight = Real(1) / (1 + square((double)pdf_pdf) * bsdf_val * light
            }
        }
        scatter_pdf = bsdf_val / pdf_pdf;
        next_throughput = throughput * scatter_pdf;
    }
}
```
A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

• automatic differentiation v.s. symbolic differentiation

```python
function f(x):
    result = x
    for i = 1 to 8:
        result = exp(result)
    return result
```
A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

- automatic differentiation v.s. symbolic differentiation

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

symbolic differentiation (37 exponents):

\[
\frac{df(x)}{dx} = e^x + e^{e^x} + e^{e^{e^x}} + e^{e^{e^{e^x}}} + e^{e^{e^{e^{e^x}}}} + e^{e^{e^{e^{e^{e^x}}}}} + e^{e^{e^{e^{e^{e^{e^x}}}}}} + e^{e^{e^{e^{e^{e^{e^{e^x}}}}}}} + e^{e^{e^{e^{e^{e^{e^{e^{e^x}}}}}}}} + e^{e^{e^{e^{e^{e^{e^{e^{e^{e^x}}}}}}}}}
\]
• automatic differentiation v.s. symbolic differentiation

symbolic differentiation (37 exponents):

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

forward-mode automatic differentiation (8 exponents):

```python
function d_f(x):
    result = x
    d_result = 1
    for i = 1 to 8:
        result = exp(result)
        d_result = d_result * result
    return d_result
```
A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

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

\[ y = f(x) \]
\[ z = g(y) \]
A CRASH COURSE OF AUTOMATIC DIFFERENTIATION

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

\[ y = f(x) \]
\[ z = g(y) \]

\[ \frac{dz}{dx} = \frac{dz}{dy} \frac{dy}{dx} \]
MENTAL MODEL: COMPUTATIONAL GRAPH

\[ y = f(x) \]
\[ z = g(y) \]

\[
\begin{align*}
\frac{dy}{dx} & \quad \text{X} \\
\frac{dz}{dy} & \quad \text{Y} \\
\end{align*}
\]
MULTIVARIATE EXAMPLE

\[ y = f(x_0, x_1) \]
\[ z = g(y) \]
\[ y = f(x_0, x_1) \]
\[ z = g(y) \]
\[ \frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x_0} \]
MULTIVARIATE EXAMPLE

\[ y = f(x_0, x_1) \]
\[ z = g(y) \]

\[ \frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x_0} \]
\[ \frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x_1} \]
MULTIVARIATE EXAMPLE

\[ y = f(x_0, x_1) \]
\[ z = g(y) \]

\[ \frac{\partial z}{\partial x_0} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x_0} \]
\[ \frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x_1} \]

\[ \frac{\partial z}{\partial y} \text{ can be factored out and be only computed once!} \]
AUTODIFF = A PATH FINDING PROBLEM
AUTODIFF = A PATH FINDING PROBLEM
AUTODIFF = A PATH FINDING PROBLEM
AUTODIFF = A PATH FINDING PROBLEM
Reversible Mode Automatic Differentiation = A Greedy Path Factorization Algorithm

\[ \frac{\partial z}{\partial y_0} \]

\[ \frac{\partial z}{\partial y_1} \]
REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

\[
\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial w_0}
\]

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial x} + \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x}
\]

\[
\frac{\partial z}{\partial y_0}
\]

\[
\frac{\partial z}{\partial y_1}
\]
REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

\[
\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial w_0} + \frac{\partial z}{\partial x} \frac{\partial x}{\partial w_0}
\]

\[
\frac{\partial z}{\partial y_0} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial y_0}
\]

\[
\frac{\partial z}{\partial y_1} = \frac{\partial z}{\partial w_0} \frac{\partial w_0}{\partial y_1} + \frac{\partial z}{\partial x} \frac{\partial x}{\partial y_1}
\]

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial x} + \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x}
\]
REVERSE-MODE AUTOMATIC DIFFERENTIATION = A GREEDY PATH FACTORIZATION ALGORITHM

\[
\frac{\partial z}{\partial w_0} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial w_0} + \frac{\partial z}{\partial x} \frac{\partial x}{\partial w_0}
\]

\[
\frac{\partial z}{\partial y_0}
\]

\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y_0} \frac{\partial y_0}{\partial x} + \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x}
\]

\[
\frac{\partial z}{\partial y_1}
\]
REVERSE-MODE AUTOMATIC DIFFERENTIATION PRODUCES EFFICIENT GRADIENTS

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

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

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

```python
function d_f(x):
    result = x
    results = []
    for i = 1 to 8:
        results.push(result)
        result = exp(result)
    for i = 8 to 1:
        d_results = d_result * exp(results[i])
    return result
```
SOURCE TRANSFORM V.S. TAPEING

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

```latex
function f(x):
...

function d_f(x):
...
...
```

`f(5)` trace source transform
if (hit the red triangle)
    return red
elif (hit the blue triangle)
    return blue
else
    return white
if (hit the red triangle) 
  return red
elif (hit the blue triangle) 
  return blue
else 
  return white

• derivative of color w.r.t. triangle vertex is 0
if (hit the red triangle)
    return red
elif (hit the blue triangle)
    return blue
else
    return white

• derivative of color w.r.t. triangle vertex is 0
  – or is it?
rendering = computing integrals

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

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

pixel filter support

area light

shutter time (motion blur)

camera aperture (defocus blur)

global illumination

• wavelength
• participating media
• ...
• and more!
THE RENDERING INTEGRALS ARE DIFFERENTIABLE!

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

\[
\int
\begin{cases}
  \text{if (hit the red triangle)} & \text{return red} \\
  \text{elif (hit the blue triangle)} & \text{return blue} \\
  \text{else} & \text{return white}
\end{cases}
\]

more blue, less white

pixel filter support
We evaluate these integrals by sampling them

\[ \int \ldots \approx \sum \ldots \]

more blue, less white
DIFFERENTIATING INTEGRAL SAMPLES GIVES WRONG DERIVATIVES

\[ \frac{\partial}{\partial p} = 0 \]

more blue, less white
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

more blue, less white
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

\[ \frac{\partial}{\partial p} = \text{more blue, less white} \]
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

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

\[ \frac{\partial}{\partial p} = 0 \]
KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

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

\[ \frac{\partial}{\partial p} = 0 \]
LET’S DERIVE THE DERIVATIVES IN 1D

\[ \int_{x=0}^{x=1} \begin{cases} 1 & \text{for } x < p \quad \text{(the blue area)} \\ 0.5 & \text{otherwise} \end{cases} \]

BASICS
LET'S DERIVE THE DERIVATIVES IN 1D

\[
\text{derivative w.r.t. } p = \begin{cases} 1 & \text{if } x < p \\ 0.5 & \text{otherwise} \end{cases}
\]

\[
\int_{x=0}^{x=1} \text{(the blue area)}
\]

\[
x < p \quad ? \quad 1 \quad : \quad 0.5
\]
LET’S DERIVE THE DERIVATIVES IN 1D

• Trick: move the discontinuities to the integral boundaries

\[
\begin{align*}
\int_{x=0}^{1} x = \begin{cases} 
1 & \text{if } x < p \\
0.5 & \text{otherwise}
\end{cases}
\end{align*}
\]

\[
= \int_{x=0}^{p} 1 + \int_{x=p}^{1} 0.5
\]
LET’S DERIVE THE DERIVATIVES IN 1D

• Trick: move the discontinuities to the integral boundaries

\[
\int_{x=0}^{x=1} \left( x < p \ ? \ 1 : 0.5 \right)
\]

(the blue area)

\[
= \int_{x=0}^{x=p} 1 + \int_{x=p}^{x=1} 0.5
\]
DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

\[
\int_{x=0}^{x=1} \text{if } x < p \, ? \, 1 : 0.5
\]

\[
\frac{\partial}{\partial p} \left( \int_{x=0}^{x=p} 1 + \int_{x=p}^{x=1} 0.5 \right) = 1 - 0.5
\]
DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

\[
\frac{\partial}{\partial p} \int f dp = \int \frac{\partial}{\partial p} f dp + \sum f_- - f_+
\]

“the Leibniz’s integral rule”
DISCONTINUITY DERIVATIVES = DIFFERENCES AT DISCONTINUITIES

\[
\frac{\partial}{\partial p} \int f(p) \, dp = \int \frac{\partial}{\partial p} f(p) \, dp + \sum (f_+ - f_-)
\]

“the Leibniz’s integral rule”
Reynolds transport theorem
[Reynolds 1903]
GENERALIZE TO 2D

\[ \frac{\partial}{\partial p} \int \int \partial p = \int \int \frac{\partial}{\partial p} \]

Reynolds transport theorem
[Reynolds 1903]
DERIVING THE 2D BOUNDARY DERIVATIVE

- boundary derivative = infinitesimal volume change w.r.t. parameter
DERIVING THE 2D BOUNDARY DERIVATIVE

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

3D view around the purple sample
DERIVING THE 2D BOUNDARY DERIVATIVE

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

3D view around the purple sample
DERIVING THE 2D BOUNDARY DERIVATIVE

\[ \nu = \text{boundary movement w.r.t. param} \]

\[ \nu = \frac{\partial x}{\partial p} \]

\[ n \cdot \nu \]

\[ \text{(width)} \]

\[ \text{dt} \]

\[ \text{(length)} \]

\[ \int dt \]
\[
\int dt = \int (f_- - f_+) (n \cdot v) \, dt
\]

*height*  *width*  *length*

*red - blue*
RECAP

RENDERING = COMPUTING INTEGRALS

- shutter time (motion blur)
- camera aperture (defocus blur)
- area light

- pixel filter support
- global illumination

- wavelength
- transmittance
- and more!

Realistic Image Synthesis SS2024
RECAP

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

more blue, less white

pixel filter support
DIFFERENTIATING INTEGRAL SAMPLES GIVES WRONG DERIVATIVES

\[
\frac{\partial}{\partial p} = 0
\]

more blue, less white
RECAP

KEY IDEA: EXPLICITLY INTEGRATE THE BOUNDARIES

more blue, less white
RECAP

\[ \frac{\partial}{\partial p} \int \int \text{boundary derivative} = \int \int \frac{\partial}{\partial p} \text{interior derivative} \]

Reynolds transport theorem [Reynolds 1903]
DISCUSSION

• Ray tracing vs rasterization
• Approximated solutions
• Geometry representation
• Limitations
RAY TRACING VS RASTERIZATION

• The boundary sampling is not very compatible with z-buffer rendering
RAY TRACING VS RASTERIZATION

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

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

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

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

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

23823 vertices, 44702 faces

• 1024x1024 at 2 spp (Titan Xp) forward + backward
  – Ray tracing + edge sampling: 0.05—0.1 sec
  – PyTorch3D: 0.15 sec
RAY TRACING VS RASTERIZATION

23823 vertices, 44702 faces

initial

edge sampling optimization video (1 view over 20)

abs. error

Low

High
RAY TRACING VS RASTERIZATION

23823 vertices, 44702 faces

initial

edge sampling optimization video (1 view over 20)

abs. error
RAY TRACING VS RASTERIZATION

23823 vertices, 44702 faces

initial

PyTorch3D optimization video (1 view over 20)

abs. error

23823 vertices, 44702 faces
RAY TRACING VS RASTERIZATION

23823 vertices, 44702 faces

initial

PyTorch3D optimization video (1 view over 20)

abs. error

Low

High
RAY TRACING VS RASTERIZATION

Optimization results after 5000 iterations (with identical settings)

- Optimized (ray tracing)
- Target
- Optimized (PyTorch3D)
Our boundary integral is \textit{correct, i.e.}, when the number of samples grows it converges to the integral.

Two other kinds of approximation:

- Keep the rendering model, approximate the derivatives (de La Gorce 2011, OpenDR 2014, Kato 2018, …)
- Change the rendering model (Rhodin 2015, SoftRas 2019, PyTorch3D 2020…)

\[ l_j = (x_i, y_i) \]

\[ l_j \]

\[ l_j \]

\[ l_j \]

Kato 2018

Rhodin 2015

PyTorch3D [Ravi 2020]
GEOMETRY REPRESENTATION

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

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

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

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

need to extract this edge
LIMITATIONS

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

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

(a) original objective function $L$

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

Kawaguchi and Kaelbling 2019

William 1983
PHYSICS-BASED DIFFERENTIABLE RENDERING: A COMPREHENSIVE INTRODUCTION

THEORY & ALGORITHMS
• Warm-up: differential irradiance
• Warm-up: differential irradiance

• Differentiable path tracing with edge sampling

• Differential radiative transfer
DIFFERENTIABLE RENDERING THEORY & ALGORITHMS

- Warm-up: differential irradiance
- Differentiable path tracing with edge sampling
- Differential radiative transfer
- Another way of dealing with discontinuities
- Radiative backpropagation
• Warm-up: differential irradiance

• Differentiable path tracing with edge sampling
  • Differential radiative transfer

• Another way of dealing with discontinuities
  • Radiative backpropagation

• Path-space differentiable rendering
Irradiance at \( x \): \[
E = \int_{\mathbb{H}^2} f_E(\omega) \frac{L_1(\omega)}{\cos \theta} \, d\sigma(\omega)
\]
WARM-UP: DIFFERENTIAL IRRADIANCE

Irradiance at \( x \):

\[
E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega) \cos \theta} d\sigma(\omega)
\]
WARM-UP: DIFFERENTIAL IRRADIANCE

\[ E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_i(\omega) \cos \theta} d\sigma(\omega) \]

\( \pi \): emitter size

Irradiance at \( \mathbf{x} \):

- Low intensity
- High intensity
WARM-UP: DIFFERENTIAL IRRADIANCE

Irradiance at $\mathbf{x}$:

$$ E = \int_{\mathcal{H}^2} f_E(\omega) L_1(\omega) \cos \theta \, d\sigma(\omega) $$

$\pi$: emitter size

$f_E(\omega)$

Low & High

$\partial \mathcal{H}^2$
WARM-UP: DIFFERENTIAL IRRADIANCE

\[ E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega) \cos \theta} d\sigma(\omega) \]

\[ \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E(\omega)}{d\pi} d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \Delta f_E(\omega) d\ell(\omega) \]

\( \pi \): emitter size

Low \( \rightarrow \) High

Interior integral

Boundary integral

Reynolds
**WARM-UP: DIFFERENTIAL IRRADIANCE**

\[ E = \int_{\mathcal{H}^2} f_E(\omega) L_1(\omega) \cos \theta \, d\sigma(\omega) \]

\[ \frac{dE}{d\pi} = \int_{\mathcal{H}^2} \frac{df_E(\omega)}{d\pi} (\omega) \, d\sigma(\omega) + \int_{\partial\mathcal{H}^2} V_{\partial\mathcal{H}^2}(\omega) \Delta f_E(\omega) \, d\ell(\omega) \]

\( \pi \): emitter size

**Reynolds**

Low

High

Interior integral = 0
**WARM-UP: DIFFERENTIAL IRRADIANCE**

- **π**: emitter size

\[
E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega)} \cos \theta \, d\sigma(\omega)
\]

Reynolds

\[
\frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E(\omega)}{d\pi} d\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega) \Delta f_E(\omega) \, d\ell(\omega)
\]

**Boundary integral**
WARM-UP: DIFFERENTIAL IRRADIANCE

\[ E = \int_{\mathbb{H}^2} f_E(\omega) L(\omega) \cos \theta d\sigma(\omega) \]

\[ \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E(\omega)}{d\pi} d\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \Delta f_E(\omega) d\ell(\omega) \]

\( \pi \): emitter size

\[ f_E(\omega) \]

Boundary integral
Warm-up: Differential Irradiance

- \( \pi \): emitter size

Scalar normal “velocity” of \( \omega \)

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

Boundary integral

\[
E = \int_{\mathbb{H}^2} f_E(\omega) L_1(\omega) \cos \theta \, d\sigma(\omega)
\]

Reynolds

\[
\frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E}{d\pi}(\omega) \, d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \, \Delta f_E(\omega) \, d\mathcal{L}(\omega)
\]
WARM-UP: DIFFERENTIAL IRRADIANCE

$\pi$: emitter size

$E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega) \cos \theta} d\sigma(\omega)$

Scalar normal “velocity” of $\omega$

$V_{\partial \mathbb{H}^2}(\omega) = \left< n(\omega), \frac{d\omega}{d\pi} \right>$

Boundary integral

$\frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E(\omega)}{d\pi} d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \Delta f_E(\omega) d\ell(\omega)$
WARM-UP: DIFFERENTIAL IRRADIANCE

\[ f_E(\omega) \]

Scalar normal “velocity” of \( \omega \)

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

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

Difference of the integrand \( f_E \) across the boundary

\[ E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega)} \cos \theta \, d\sigma(\omega) \]

\[ \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E(\omega)}{d\pi} \, d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \, \Delta f_E(\omega) \, d\ell(\omega) \]

\( \pi \): emitter size

Low \quad High

Difference of the integrand across the boundary

Boundary integral

Reynolds
WARM-UP: DIFFERENTIAL IRRADIANCE

\( \pi \): emitter size

\[
f_E(\omega) = \frac{f_E(\omega)}{L_1(\omega) \cos \theta} \, d\sigma(\omega)
\]

\[
E = \int_{\mathbb{H}^2} f_E(\omega) \, \cos \theta \, d\sigma(\omega)
\]

Scalar normal “velocity” of \( \omega \)

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

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

Difference of the integrand \( f_E \) across the boundary

\[
\Delta f_E(\omega) = f_E^- (\omega) - f_E^+ (\omega)
\]

Boundary integral

\[
\frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E}{d\pi}(\omega) \, d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \, \Delta f_E(\omega) \, d\ell(\omega)
\]
**WARM-UP: DIFFERENTIAL IRRADIANCE**

\[ E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_1(\omega)} \cos \theta \, d\sigma(\omega) \]

**Interior integral**

\[ \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E}{d\pi}(\omega) \, d\sigma(\omega) + \int_{\partial\mathbb{H}^2} V_{\partial\mathbb{H}^2}(\omega) \, \Delta f_E(\omega) \, d\ell(\omega) \]

**Boundary integral**

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

Scalar normal “velocity” of \( \omega \)

Difference of the integrand \( f_E \) across the boundary

\[ \Delta f_E(\omega) = f_E^-(\omega) - f_E^+(\omega) \]

**General result**

\( \pi \): emitter size
\[ E = \int_{\mathbb{H}^2} f_E(\omega) \frac{L_1(\omega) \cos \theta}{d\sigma(\omega)} \text{ d}E \]

\[ \frac{dE}{d\pi} = \int_{\mathbb{H}^2} \frac{df_E}{d\pi}(\omega) d\sigma(\omega) + \int_{\partial \mathbb{H}^2} V_{\partial \mathbb{H}^2}(\omega) \Delta f_E(\omega) \text{ d}\ell(\omega) \]
This can be generalized easily to obtain the differential rendering equation:

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

Rendering equation

$$L(\omega_o) = \int_{S^2} L_i(\omega_i) f_s(\omega_i, \omega_o) d\sigma(\omega_i) + L_e(\omega_o)$$

$$f_s : \text{cosine-weighted BSDF}$$
This can be generalized easily to obtain the differential rendering equation:

\[
E = \int_{\mathbb{H}^2} \frac{f_E(\omega)}{L_i(\omega)} \cos \theta \, d\sigma(\omega)
\]

\[
\frac{dE}{d\pi} = \int \frac{df_E(\omega)}{d\pi} \, d\sigma(\omega) + \int \frac{df_E(\omega)}{d\pi} \, d\sigma(\omega) + \int V_{\partial \mathbb{H}^2(\omega)} \, d\mathcal{L}(\omega)
\]

Rendering equation

\[
L(\omega_o) = \int_{\mathbb{S}^2} L_i(\omega_i) f_s(\omega_i, \omega_o) \, d\sigma(\omega_i) + L_e(\omega_o)
\]

Differential rendering equation

\[
\frac{d}{d\pi} L(\omega_o) = \int_{\mathbb{S}^2} \frac{d}{d\pi} f_{RE}(\omega_i) \, d\sigma(\omega_i) + \int \frac{d}{d\pi} L_e(\omega_o)
\]
SOURCES OF DISCONTINUITIES

Assumptions:

No zero-measure (point and directional) lights

No perfectly specular surfaces

Continuous BSDFs
Sources of Discontinuities

Assumptions:

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

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

Continuous BSDFs
SOURCES OF DISCONTINUITIES

Assumptions:

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

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

**Continuous BSDFs**

These limitations are largely practical and can be easily mitigated
SOURCES OF DISCONTINUITIES

**Boundary edges**

(Topological) boundary of an object
SOURCES OF DISCONTINUITIES

**Boundary edges**

(Topological) boundary of an object

**Sharp edges**

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

**Boundary edges**
(Topological) boundary of an object

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

**Silhouette edges**
View-dependent object silhouettes
Path tracing can be generalized to estimate $L$ and $dL/d\pi$ jointly.

Rendering equation

$$L(\omega_o) = \int_{S^2} \left( f_{RE}(\omega_i) \right) f_s(\omega_i, \omega_o) L_i(\omega_i) d\sigma(\omega_i) + L_e(\omega_o)$$

Differential rendering equation

$$\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{RE}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o)$$
Path tracing can be generalized to estimate $L$ and $dL/d\pi$ jointly.

**Rendering equation**

$$L(\omega_o) = \int_{S^2} f_{RE}(\omega_i) \frac{f_s(\omega_i, \omega_o) L_i(\omega_i)}{f_s(\omega_i, \omega_o) L_i(\omega_i) d\sigma(\omega_i)} + L_e(\omega_o)$$

**Differential rendering equation**

$$\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{RE}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o)$$

Standard path tracing

**Interior integral**

Boundary integral
Path tracing can be generalized to estimate $L$ and $dL/d\pi$ jointly.

\[
L(\omega_o) = \int_{S^2} f_{RE}(\omega_i) f_s(\omega_i, \omega_o) L_1(\omega_i) \, d\sigma(\omega_i) + L_e(\omega_o)
\]

\[
\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{RE}(\omega_i) \, d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) \, d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o)
\]

Rendering equation

Differential rendering equation

Interior integral

Boundary integral

Standard path tracing

Edge sampling
Differentiable Monte Carlo Ray Tracing through Edge Sampling

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

SIGGRAPH Asia 2018
DPT(\(x, \omega_o\)): # Estimate \(L(x, \omega_o)\) and \(\frac{\partial}{\partial \pi} [L(x, \omega_o)]\) jointly

\[
\text{sample } \omega_{i,1} \in S^2 \text{ with probability } p_{i,1} \\
y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \\
(L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \\
L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}} \\
\hat{L} \leftarrow \frac{\frac{\partial}{\partial \pi} [f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}}
\]

\[
\text{sample } \omega_{i,2} \in \partial S^2 \text{ with probability } p_{i,2} \\
\hat{L} \leftarrow \hat{L} + V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2}) \\
\text{return } \left( L + L_e(x, \omega_o), \hat{L} + \frac{\partial}{\partial \pi} L_e(x, \omega_o) \right)
\]

Rendering equation

\[
L(\omega_o) = \int_{S^2} \frac{f_{RE}(\omega_i)}{f_s(\omega_i, \omega_o) L_i(\omega_i)} d\sigma(\omega_i) + L_e(\omega_o)
\]

Differential rendering equation

\[
\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{RE}(\omega_i) d\sigma(\omega_i) \\
+ \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) d\ell(\omega_i) \\
+ \frac{d}{d\pi} L_e(\omega_o)
\]
dPT(x, ω₀): # Estimate L(x, ω₀) and \( \frac{d}{d\pi}[L(x, ω₀)] \) jointly

sample \( ω_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\[ y \leftarrow \text{rayIntersect}(x, ω_{i,1}) \]

\( (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -ω_{i,1}) \)

\[ L \leftarrow \frac{f_s(x, ω_{i,1}, ω₀) L_i}{p_{i,1}} \]

\[ \hat{L} \leftarrow \frac{d}{d\pi}[f_s(x, ω_{i,1}, ω₀)] L_i + f_s(x, ω_{i,1}, ω₀) \hat{L}_i \]

\( \hat{L} \)

sample \( ω_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[ \hat{L} \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, ω_{i,2}) f_s(x, ω_{i,2}, ω₀) ΔL_i(x, ω_{i,2})}{p_{i,2}} \]

return \( L + L_e(x, ω₀), \hat{L} + \frac{d}{d\pi}L_e(x, ω₀) \)

Rendering equation

\[ L(ω₀) = \int_{S^2} \frac{f_{RE}(ω_i)}{f_s(ω_i, ω₀) L_i(ω_i) dσ(ω_i)} + L_e(ω₀) \]

Differential rendering equation

\[ \frac{d}{d\pi} L(ω₀) = \int_{S^2} \frac{d}{d\pi} f_{RE}(ω_i) dσ(ω_i) \]

\[ + \int_{\partial S^2} V_{\partial S^2}(ω_i) Δf_{RE}(ω_i) dℓ(ω_i) \]

\[ + \frac{d}{d\pi} L_e(ω₀) \]
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[
d\text{PT}(x, \omega_o): \quad \# \text{Estimate } L(x, \omega_o) \text{ and } \frac{d}{d\pi}[L(x, \omega_o)] \text{ jointly}
\]

- Sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)
  \[
y \leftarrow \text{rayIntersect}(x, \omega_{i,1})
\]
- \( (L_i, \hat{L}_i) \leftarrow d\text{PT}(y, -\omega_{i,1}) \)
  \[
  L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o)}{p_{i,1}} L_i
  \]
  \[
  \hat{L} \leftarrow \frac{d}{d\pi} \left[ f_s(x, \omega_{i,1}, \omega_o) \right] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i
  \]
- Sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)
- \( \hat{L} \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \)
- Return \( \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\pi} L_e(x, \omega_o) \right) \)

Rendering equation
\[
L(\omega_o) = \int_{S^2} f_{\text{RE}}(\omega_i) \left( f_s(\omega_i, \omega_o) L_i(\omega_i) d\sigma(\omega_i) + L_e(\omega_o) \right)
\]

Differential rendering equation
\[
\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{\text{RE}}(\omega_i) d\sigma(\omega_i)
\]
\[
\quad + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{\text{RE}}(\omega_i) d\ell(\omega_i)
\]
\[
\quad + \frac{d}{d\pi} L_e(\omega_o)
\]
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

**dPT(x, \( \omega_o \))**

# Estimate \( L(x, \omega_o) \) and \( \frac{d}{d\pi}[L(x, \omega_o)] \) jointly

Sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\[
y \leftarrow \text{rayIntersect}(x, \omega_{i,1})
\]

\[
(L_i, \dot{L}_i) \leftarrow \text{dPT}(y, - \omega_{i,1})
\]

\[
L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o)}{p_{i,1}} L_i
\]

\[
\dot{L} \leftarrow \frac{d}{d\pi} \left[ f_s(x, \omega_{i,1}, \omega_o) \right] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i
\]

Sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[
\dot{L} \leftarrow \dot{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}
\]

Return \( L + L_e(x, \omega_o), \dot{L} + \frac{d}{d\pi} L_e(x, \omega_o) \)

**Rendering equation**

\[
L(\omega_o) = \int_{S^2} f_{RE}(\omega_i) \left( f_s(\omega_i, \omega_o) L_i(\omega_i) d\sigma(\omega_i) \right) + L_e(\omega_o)
\]

**Differential rendering equation**

\[
\frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{RE}(\omega_i) d\sigma(\omega_i)
\]

\[
+ \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) \, d\ell(\omega_i)
\]

\[
+ \frac{d}{d\pi} L_e(\omega_o)
\]
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

dPT(\(x, \omega_o\)):  # Estimate \(L(x, \omega_o)\) and \(\frac{d}{d\pi} [L(x, \omega_o)]\) jointly

sample \(\omega_{i,1} \in \mathbb{S}^2\) with probability \(p_{i,1}\)

\(y \leftarrow \text{rayIntersect}(x, \omega_{i,1})\)

\((L_i, \dot{L}_i) \leftarrow dPT(y, -\omega_{i,1})\)

\(L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}\)

\(\dot{L} \leftarrow \frac{\frac{d}{d\pi} [f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}}\)

sample \(\omega_{i,2} \in \partial\mathbb{S}^2\) with probability \(p_{i,2}\)

\(\dot{L} \leftarrow \dot{L} + \frac{V_{\partial\mathbb{S}^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}\)

return \(\left(L + L_e(x, \omega_o), \dot{L} + \frac{d}{d\pi} L_e(x, \omega_o)\right)\)

Rendering equation

\(L(\omega_o) = \int_{\mathbb{S}^2} \underbrace{f_{\text{RE}}(\omega_i)}_{\mathbb{S}^2} \underbrace{f_s(\omega_i, \omega_o) L_i(\omega_i)}_{\mathbb{S}^2} d\sigma(\omega_i) + L_e(\omega_o)\)

Differential rendering equation

\(\frac{d}{d\pi} L(\omega_o) = \int_{\mathbb{S}^2} \frac{d}{d\pi} f_{\text{RE}}(\omega_i) d\sigma(\omega_i)\)

\(+ \int_{\partial\mathbb{S}^2} V_{\partial\mathbb{S}^2}(\omega_i) \Delta f_{\text{RE}}(\omega_i) d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o)\)
dPT(x, \omega_o): \# Estimate L(x, \omega_o) and \frac{d}{d\pi}[L(x, \omega_o)] jointly

- sample \omega_{i,1} \in S^2 with probability p_{i,1}
- y ← rayIntersect(x, \omega_{i,1})
- (L_i, \hat{L}_i) ← dPT(y, - \omega_{i,1})
- L ← \frac{f_s(x, \omega_{i,1}, \omega_o)L_i}{p_{i,1}}
- \hat{L} ← \frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)]L_i + f_s(x, \omega_{i,1}, \omega_o)\hat{L}_i

- sample \omega_{i,2} \in \partial S^2 with probability p_{i,2}
- \hat{L} ← \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2})f_s(x, \omega_{i,2}, \omega_o)\Delta L_i(x, \omega_{i,2})}{p_{i,2}}

return \left( L + L_e(x, \omega_o), \frac{d}{d\pi}L_e(x, \omega_o) \right)

Rendering equation

\[ L(\omega_o) = \int_{S^2} f_{RE}(\omega_i) f_s(\omega_i, \omega_o) L_i(\omega_i) d\sigma(\omega_i) + L_e(\omega_o) \]

Differential rendering equation

\[ \frac{d}{d\pi}L(\omega_o) = \int_{S^2} \frac{d}{d\pi}f_{RE}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{RE}(\omega_i) d\ell(\omega_i) \]

+ \frac{d}{d\pi}L_e(\omega_o)
DIFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[ \text{dPT}(x, \omega_o): \quad \# \text{Estimate } L(x, \omega_o) \text{ and } \frac{d}{d\pi}[L(x, \omega_o)] \text{ jointly} \]

sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\[ y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \]

\[ (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \]

\[ L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o)}{p_{i,1}} L_i \]

\[ \hat{L} \leftarrow \frac{\frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}} \]

sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[ \hat{L} \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \]

return \( \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\pi} L_e(x, \omega_o) \right) \)

Rendering equation

\[ L(\omega_o) = \int_{S^2} \underbrace{f_{\text{RE}}(\omega_i)}_{f_s(\omega_i, \omega_o) L_i(\omega_i)} d\sigma(\omega_i) + L_e(\omega_o) \]

Differential rendering equation

\[ \frac{d}{d\pi} L(\omega_o) = \int_{S^2} \frac{d}{d\pi} f_{\text{RE}}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{\text{RE}}(\omega_i) d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o) \]

Monte Carlo edge sampling

Standard PT w/ symbolic differentiation

DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[ \text{dPT}(x, \omega_o): \quad \# \text{Estimate } L(x, \omega_o) \text{ and } \frac{\text{d}}{\text{d}\omega} L(x, \omega_o) \text{ jointly} \]

\[ \begin{align*}
\text{sample } & \omega_{i,1} \in S^2 \text{ with probability } p_{i,1} \\
y & \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \\
(L_i, \hat{L}_i) & \leftarrow \text{dPT}(y, -\omega_{i,1}) \\
L & \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o)}{p_{i,1}} \cdot L_i \\
\hat{L} & \leftarrow \frac{\frac{\text{d}}{\text{d}\omega} f_s(x, \omega_{i,1}, \omega_o) \cdot L_i}{p_{i,1}} + f_s(x, \omega_{i,1}, \omega_o) \cdot \hat{L}_i \\
\text{sample } & \omega_{i,2} \in \partial S^2 \text{ with probability } p_{i,2} \\
\hat{L} & \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \\
\text{return } & \left( L + L_e(x, \omega_o), \hat{L} + \frac{\text{d}}{\text{d}\omega} L_e(x, \omega_o) \right)
\end{align*} \]

Rendering equation

\[ L(\omega_o) = \int_{S^2} \underbrace{f_{\text{RE}}(\omega_i)}_{\text{diff. render. equation}} \cdot L_i(\omega_i) \, d\sigma(\omega_i) + L_e(\omega_o) \]

Differential rendering equation

\[ \frac{\text{d}}{\text{d}\omega} L(\omega_o) = \int_{S^2} \frac{\text{d}}{\text{d}\omega} f_{\text{RE}}(\omega_i) \, d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \cdot \Delta f_{\text{RE}}(\omega_i) \, d\ell(\omega_i) \]

\[ + \frac{\text{d}}{\text{d}\omega} L_e(\omega_o) \]
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[ dPT(x, \omega_o): \] # Estimate \( L(x, \omega_o) \) and \( \frac{d}{d\tau}[L(x, \omega_o)] \) jointly

sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\[ y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \]

\( (L_i, \hat{L}_i) \leftarrow dPT(y, - \omega_{i,1}) \)

\[ L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}} \]

\[ \hat{L} \leftarrow \frac{\frac{d}{d\tau}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}} \]

sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[ \hat{L} \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \]

return \( \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\tau} L_e(x, \omega_o) \right) \)

\[ \Delta f_{\text{RE}} = \Delta(f_s L_i) = f_s \Delta L_i \]

(assuming \( f_s \) to be continuous)

Rendering equation

\[ L(\omega_o) = \int_{S^2} f_{\text{RE}}(\omega_i) L_i(\omega) d\sigma(\omega_i) + L_e(\omega_o) \]

Differential rendering equation

\[ \frac{d}{d\tau} L(\omega_o) = \int_{S^2} \frac{d}{d\tau} f_{\text{RE}}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2}(\omega_i) \Delta f_{\text{RE}}(\omega_i) d\ell(\omega_i) + \frac{d}{d\tau} L_e(\omega_o) \]

Monte Carlo edge sampling

Standard PT w/ symbolic differentiation

Realistic Image Synthesis SS2024
MONTE CARLO EDGE SAMPLING

\[ \text{dPT}(x, \omega_o): \quad \# \text{Estimate } L(x, \omega_o) \text{ and } \frac{d}{d\pi} [L(x, \omega_o)] \text{ jointly} \]

sample \( \omega_{i,1} \in \mathbb{S}^2 \) with probability \( p_{i,1} \)

\( y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \)

\( (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \)

\[
L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}
\]

\[
\dot{L} \leftarrow \frac{\frac{d}{d\pi} [f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}}
\]

sample \( \omega_{i,2} \in \partial \mathbb{S}^2 \) with probability \( p_{i,2} \)

\[
\dot{L} \leftarrow \dot{L} + \frac{V_{\partial \mathbb{S}^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}
\]

return \( (L + L_e(x, \omega_o), \dot{L} + \frac{d}{d\pi} L_e(x, \omega_o)) \)

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining \( \partial \mathbb{S}^2 \), the discontinuity points of \( \Delta L_i \) (w.r.t. incident direction \( \omega_i \))
MONTE CARLO EDGE SAMPLING

\[ \text{dPT}(x, \omega_o): \] # Estimate \( L(x, \omega_o) \) and \( \frac{d}{d\pi}[L(x, \omega_o)] \) jointly

sample \( \omega_{i,1} \in \mathbb{S}^2 \) with probability \( p_{i,1} \)
\( y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \)
\( (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, - \omega_{i,1}) \)
\[
L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}
\]
\[
\hat{L} \leftarrow \frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i
\]

sample \( \omega_{i,2} \in \partial\mathbb{S}^2 \) with probability \( p_{i,2} \)
\[
\hat{L} \leftarrow \hat{L} + \frac{V_{\partial\mathbb{S}^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}
\]
return \( \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\pi} L_e(x, \omega_o) \right) \)

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining \( \partial\mathbb{S}^2 \), the discontinuity points of \( \Delta L_i \) (w.r.t. incident direction \( \omega_i \))

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

\[ \text{dPT}(x, \omega_o): \]  
\# Estimate \( L(x, \omega_o) \) and \( \frac{d}{d\pi}[L(x, \omega_o)] \) jointly

sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\[ y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \]

\( (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \)

\[ L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o)}{p_{i,1}} L_i \]

\[ \hat{L} \leftarrow \frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i \]

sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[ \hat{L} \leftarrow \hat{L} + V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2}) \]

\[ \text{return} \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\pi} L_e(x, \omega_o) \right) \]

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining \( \partial S^2 \), the discontinuity points of \( \Delta L_i \) (w.r.t. incident direction \( \omega_i \))

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

- Requires traversing a 6D BVH
- Expensive for complex scenes
MONTE CARLO EDGE SAMPLING

\( \text{dPT}(x, \omega_o): \) # Estimate \( L(x, \omega_o) \) and \( \frac{d}{d\tau}[L(x, \omega_o)] \) jointly

sample \( \omega_{i,1} \in \mathbb{S}^2 \) with probability \( p_{i,1} \)

\( y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \)

\( (L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \)

\[
L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}
\]

\[
\hat{L} \leftarrow \frac{\frac{d}{d\tau}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}}
\]

sample \( \omega_{i,2} \in \partial \mathbb{S}^2 \) with probability \( p_{i,2} \)

\( \hat{L} \leftarrow \hat{L} + \frac{V_{\partial \mathbb{S}^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \)

return \( \left( L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\tau} L_e(x, \omega_o) \right) \)

- A new sampling procedure introduced by Li et al. [2018]
- **Key:** determining \( \partial \mathbb{S}^2 \), the discontinuity points of \( \Delta L_i \) (w.r.t. incident direction \( \omega_i \))
- For polygonal meshes, \( \partial \mathbb{S}^2 \) can involve:
  - Boundary edges (associated with only one face)
  - Face edges (when not using smooth shading)
  - Silhouette edges (shared by a front and a back face)
  - Requires traversing a 6D BVH
  - Expensive for complex scenes
  - To be addressed later!
COMPUTING $\Delta L_i$

$dPT(x, \omega_o)$: # Estimate $L(x, \omega_o)$ and $\frac{d}{d\pi}[L(x, \omega_o)]$ jointly

- sample $\omega_{i,1} \in S^2$ with probability $p_{i,1}$
- $y \leftarrow \text{rayIntersect}(x, \omega_{i,1})$
- $(L_i, \hat{L}_i) \leftarrow dPT(y, -\omega_{i,1})$
- $L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}$
- $\hat{L} \leftarrow \frac{\frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}}$

- sample $\omega_{i,2} \in \partial S^2$ with probability $p_{i,2}$
- $\hat{L} \leftarrow \hat{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}$

- return $\left(L + L_e(x, \omega_o), \hat{L} + \frac{d}{d\pi} L_e(x, \omega_o)\right)$

Monte Carlo edge sampling
COMPUTING $\Delta L_i$

dPT($x, \omega_o$): # Estimate $L(x, \omega_o)$ and $\frac{d}{d\pi}[L(x, \omega_o)]$ jointly

sample $\omega_{i,1} \in \mathbb{S}^2$ with probability $p_{i,1}$

$y \leftarrow \text{rayIntersect}(x, \omega_{i,1})$

$(L_i, \dot{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1})$

$L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}$

$\dot{L} \leftarrow \frac{\frac{d}{d\pi}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}}$

sample $\omega_{i,2} \in \partial \mathbb{S}^2$ with probability $p_{i,2}$

$\hat{L} \leftarrow \dot{L} + \frac{V_{\partial \mathbb{S}^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}$

return $\left(L + L_e(x, \omega_o), \dot{L} + \frac{d}{d\pi}L_e(x, \omega_o)\right)$

$\Delta L_i(x, \omega_{i,2}) = \pm \left[ L(y_1, -\omega_{i,2}) - L(y_2, -\omega_{i,2}) \right]$
**COMPUTING $\Delta L_i$**

$$
\text{dPT}(\mathbf{x}, \omega_o): \quad \# \text{Estimate } L(\mathbf{x}, \omega_o) \text{ and } \frac{d}{d\tau}[L(\mathbf{x}, \omega_o)] \text{ jointly} \\
\text{sample } \omega_{i,1} \in \mathbb{S}^2 \text{ with probability } p_{i,1} \\
y \leftarrow \text{rayIntersect}(\mathbf{x}, \omega_{i,1}) \\
(L_i, \hat{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \\
L \leftarrow \frac{f_s(\mathbf{x}, \omega_{i,1}, \omega_o) L_i}{p_{i,1}} \\
\hat{L} \leftarrow \frac{\frac{d}{d\tau}[f_s(\mathbf{x}, \omega_{i,1}, \omega_o)] L_i + f_s(\mathbf{x}, \omega_{i,1}, \omega_o) \hat{L}_i}{p_{i,1}}
$$

sample $\omega_{i,2} \in \partial \mathbb{S}^2$ with probability $p_{i,2}$

$$
\hat{L} \leftarrow \hat{L} + \frac{V_{\partial\mathbb{S}^2}(\mathbf{x}, \omega_{i,2}) f_s(\mathbf{x}, \omega_{i,2}, \omega_o) \Delta L_i(\mathbf{x}, \omega_{i,2})}{p_{i,2}}
$$

return \( \left( L + L_c(\mathbf{x}, \omega_o), \hat{L} + \frac{d}{d\tau} L_c(\mathbf{x}, \omega_o) \right) \)

---

Monte Carlo edge sampling

$$
\Delta L_i(\mathbf{x}, \omega_{i,2}) = \pm \left[ L(y_1, -\omega_{i,2}) - L(y_2, -\omega_{i,2}) \right]
$$

Radiance values $L(y_1, -\omega_{i,2})$ and $L(y_2, -\omega_{i,2})$ can be computed by tracing additional “side” paths
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[ \text{dPT}(x, \omega_o): \quad \# \text{Estimate } L(x, \omega_o) \text{ and } \frac{d}{d\tau}[L(x, \omega_o)] \text{ jointly} \]

sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)

\( y \leftarrow \text{rayIntersect}(x, \omega_{i,1}) \)

\( (L_i, \dot{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1}) \)

\[ L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}} \]

\[ \dot{L} \leftarrow \frac{\frac{d}{d\tau}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}} \]

sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)

\[ \dot{L} \leftarrow \dot{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}} \]

return \( (L + L_e(x, \omega_o), \dot{L} + \frac{d}{d\tau}L_e(x, \omega_o)) \)

Standard PT w/ symbolic differentiation

Monte Carlo edge sampling
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

dPT(\(x, \omega_o\)): # Estimate \(L(x, \omega_o)\) and \(\frac{d}{d\tau}[L(x, \omega_o)]\) jointly

sample \(\omega_{i,1} \in S^2\) with probability \(p_{i,1}\)

\[y \leftarrow \text{rayIntersect}(x, \omega_{i,1})\]

\((L_i, \dot{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1})\)

\[L \leftarrow \frac{f_s(x, \omega_{i,1}, \omega_o) L_i}{p_{i,1}}\]

\[\dot{L} \leftarrow \frac{\frac{d}{d\tau}[f_s(x, \omega_{i,1}, \omega_o)] L_i + f_s(x, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}}\]

sample \(\omega_{i,2} \in \partial S^2\) with probability \(p_{i,2}\)

\[\dot{L} \leftarrow \dot{L} + \frac{V_{\partial S^2}(x, \omega_{i,2}) f_s(x, \omega_{i,2}, \omega_o) \Delta L_i(x, \omega_{i,2})}{p_{i,2}}\]

return \(\left( L + L_c(x, \omega_o), \dot{L} + \frac{d}{d\tau}L_c(x, \omega_o) \right)\)

Standard PT w/ symbolic differentiation

Monte Carlo edge sampling
DIFFERENTIABLE PATH TRACING WITH EDGE SAMPLING

\[ \text{dPT}(\mathbf{x}, \omega_o): \quad \# \text{Estimate } L(\mathbf{x}, \omega_o) \text{ and } \frac{d}{d\pi}[L(\mathbf{x}, \omega_o)] \text{ jointly} \]

- sample \( \omega_{i,1} \in S^2 \) with probability \( p_{i,1} \)
- \( y \leftarrow \text{rayIntersect}(\mathbf{x}, \omega_{i,1}) \)
- \((L_i, \dot{L}_i) \leftarrow \text{dPT}(y, -\omega_{i,1})\)

\[ L \leftarrow \frac{f_s(\mathbf{x}, \omega_{i,1}, \omega_o) L_i}{p_{i,1}} \]

\[ \dot{L} \leftarrow \frac{\frac{d}{d\pi}[f_s(\mathbf{x}, \omega_{i,1}, \omega_o)] L_i + f_s(\mathbf{x}, \omega_{i,1}, \omega_o) \dot{L}_i}{p_{i,1}} \]

- sample \( \omega_{i,2} \in \partial S^2 \) with probability \( p_{i,2} \)
- \( \dot{L} \leftarrow \dot{L} + \frac{V_{\partial S^2}(\mathbf{x}, \omega_{i,2}) f_s(\mathbf{x}, \omega_{i,2}, \omega_o) \Delta L_i(\mathbf{x}, \omega_{i,2})}{p_{i,2}} \)

return \( \left( L + L_c(\mathbf{x}, \omega_o), \dot{L} + \frac{d}{d\pi}L_c(\mathbf{x}, \omega_o) \right) \)
A Differential Theory of Radiative Transfer

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

SIGGRAPH Asia 2019
RECAP: RADIATIVE TRANSFER THEORY
RECAP: RADIATIVE TRANSFER THEORY
RECAP: RADIATIVE TRANSFER THEORY
RECAP: RADIATIVE TRANSFER THEORY

Radiative transfer equation (RTE) in operator form:

\[ L = K_T K_C L + Q \]
DIFFERENTIATING THE RTE

\[ L = K_T K_C L + Q \]

Differentiating both sides

\[ \partial_\pi L = \partial_\pi (K_T K_C L) + \partial_\pi Q \]
Differentiating individual operators

\[ L = K_T K_C L + Q \]

\[ \partial_\pi L = \partial_\pi (K_T K_C L) + \partial_\pi Q \]


Realistic Image Synthesis SS2024
DIFFERENTIATING THE COLLISION OPERATOR

RTE: \( L = K_T K_c L + Q \)

\( (KcL)(\omega) = \int_{S^2} \sigma_s f(\omega_i) f_p(\omega_i, \omega) L(\omega_i) d\omega_i \)

(\( x \) omitted for notational simplicity)

\( \partial_\pi \int_{S^2} f(\omega_i) d\omega_i = ? \)

Requires differentiating a spherical integral
DIFFERENTIATING THE COLLISION OPERATOR

\[ (KcL) (\omega) = \sigma_s \int_{S^2} f_p (\omega_i, \omega) L(\omega_i) d\omega_i \]

\[ \partial_\pi \int_{S^2} f(\omega_i) d\omega_i \]
DIFFERENTIATING THE COLLISION OPERATOR

\[ (KcL)(\omega) = \sigma_s \int_{S^2} f_p(\omega_i, \omega) L(\omega_i) d\omega_i \]

\[
\partial_\pi \int_{S^2} f(\omega_i) d\omega_i = \int_{S^2} \partial_\pi f(\omega_i) d\omega_i + \int_{\partial S^2} \left< n, \frac{\partial \omega_i}{\partial \pi} \right> \Delta f(\omega_i) d\omega_i
\]

By applying Reynolds transport theorem

(largely identical to the differentiation of the rendering equation)
OTHER TERMS IN THE RTE

\[ L = K_T K_C L + Q \]

Transport operator

\[ (K_T K_C L)(x, \omega) = \int_0^D T(x', x) (K_C L)(x', \omega) d\tau \]

Transmittance

Source

\[ Q = T(x_0, x)L_S(x_0, \omega) \]
OTHER TERMS IN THE RTE

\[ L = K_T K_c L + Q \]

**Transport operator** (can be differentiated using Leibniz’s rule)

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

**Source**

\[ Q = T(x_0, x) L_s(x_0, \omega) \]
This is Eq. (32) of the work by Zhang et al. [2019]
DIFFERENTIAL RTE, OPERATOR FORM

\[ L = K_T K_c L + Q \]
\[ \partial_\pi L = \partial_\pi (K_T K_c L) + \partial_\pi Q \]
Differential radiative transfer equation

\[ L = K_T K_C L + Q \]

\[ \partial_\pi L = \partial_\pi (K_T K_C L) + \partial_\pi Q \]

\[
\begin{pmatrix}
\partial_\pi L \\
L
\end{pmatrix} =
\begin{pmatrix}
K_T & K_* \\
0 & K_T K_C
\end{pmatrix}
\begin{pmatrix}
\partial_\pi L \\
L
\end{pmatrix} +
\begin{pmatrix}
\partial_\pi Q \\
Q
\end{pmatrix}
\]
Differential radiative transfer equation

\[ L = K_T K_C L + Q \]

\[ \partial_\pi L = \partial_\pi (K_T K_C L) + \partial_\pi Q \]

Differential radiative transfer equation

Captures the boundary integrals
SIGNIFICANCE OF THE BOUNDARY INTEGRAL

\[ P_{\text{light}}(\pi) = P_0 + \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix} \]

\[ P_{\text{cube}}(\pi) = P_1 + \begin{pmatrix} 0 \\ \pi \\ 0 \end{pmatrix} \]

Original image
SIGNIFICANCE OF THE BOUNDARY INTEGRAL

Original image

Derivative image

Derivative image (w/o boundary integral)
SIGNIFICANCE OF THE BOUNDARY INTEGRAL

Original image  Derivative image  Derivative image (w/o boundary integral)
DIFFERENTIABLE VOLUMETRIC PATH TRACING

Component 1: (interior)
Derivative of path measurement contribution
DIFFERENTIABLE VOLUMETRIC PATH TRACING

Component 1: (interior)
Derivative of path measurement contribution

Component 2: (boundary)
Side paths estimating $\Delta L$
INVERSE-RENDERING RESULTS

• Scene configurations
  – Participating media
  – Changing geometry

• Optimization
  – Using only image loss (L2)
INVERSE-RENDERING RESULTS

Apple in a box

Parameters

Apple position

Cube roughness

Target

Optimization process
INVERSE-RENDERING RESULTS

Apple in a box

Parameters

Apple position

Cube roughness

Target

Optimization process

Realistic Image Synthesis SS2024
Non-line-of-sight inverse rendering

Heterogeneous medium

Medium orientation (parameter)

Medium optical density (parameter)
INVERSE-RENDERING RESULTS

Non-line-of-sight inverse rendering

Target | Optimization process | Different view
INVERSE-RENDERING RESULTS

Non-line-of-sight inverse rendering

Target

Optimization process

Different view
Design-inspired inverse rendering

**Parameters**
- Light direction
- Light color
- Light falloff angle
Inverse-Rendring Results

Design-inspired inverse rendering

Target

Optimization process
Design-inspired inverse rendering

TARGET

Optimization process
\[
L(\omega_0) = \int_{S^2} \left( L_i(\omega_i) f_{\text{RE}}(\omega_i, \omega_0) \right) d\sigma(\omega_i) + L_e(\omega_0)
\]

\[
\frac{d}{d\pi} L(\omega_0) = \int_{S^2} \frac{d}{d\pi} f_{\text{RE}}(\omega_i) d\sigma(\omega_i) + \int_{\partial S^2} V_{\partial S^2} f_{\text{RE}}(\omega_i) d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_0)
\]
### CHALLENGES

Rendering equation

\[
L(\omega_o) = \int_{\mathbb{S}^2} \frac{f_{\text{RE}}(\omega_i)}{L_i(\omega_i) f_s(\omega_i, \omega_o)} \, d\sigma(\omega_i) + L_e(\omega_o)
\]

Differential rendering equation

\[
\frac{d}{d\pi} L(\omega_o) = \int_{\mathbb{S}^2} \frac{d}{d\pi} f_{\text{RE}}(\omega_i) \, d\sigma(\omega_i) + \int_{\partial \mathbb{S}^2} \frac{d}{d\pi} V_{\partial \mathbb{S}^2}(\omega_i) \, \Delta f_{\text{RE}}(\omega_i) \, d\ell(\omega_i) + \frac{d}{d\pi} L_e(\omega_o)
\]

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

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

- **Scaling out to millions of parameters**
Reparameterizing Discontinuous Integrals for Differentiable Rendering

Guillaume Loubet, Nicolas Holzschuch, Wenzel Jakob

SIGGRAPH Asia 2019
MOVING DISCONTINUITIES

Pixel integrals
\[ \iint d x \, d y \]

Light integrals
\[ \int d \omega \]

BSDF integrals
\[ \int d \omega \]
MOVING DISCONTINUITIES

Pixel integrals: $\iiint \ dx \ dy$

Light integrals: $\int d\omega$

BSDF integrals: $\int d\omega$

Scene parameter $x_i$
MOVING DISCONTINUITIES

Pixel integrals
\[ \iint \text{d}x \text{d}y \]

Light integrals
\[ \int \text{d}\omega \]

BSDF integrals
\[ \int \text{d}\omega \]

Scene parameter \( x_i \)
MOVING DISCONTINUITIES

Cannot differentiate standard Monte Carlo estimates
EDGE SAMPLING
We currently don’t have good acceleration data structures for this operation.
REPARAMETERIZE INTEGRALS?

Non-differentiable Monte Carlo estimates

Differentiable Monte Carlo estimates

Pixel filter or BRDF

$x_i$
REPARAMETERIZE INTEGRALS?

Non-differentiable Monte Carlo estimates

Differentiable Monte Carlo estimates

Pixel filter or BRDF

$x_i$
REPARAMETERIZE INTEGRALS?

Non-differentiable Monte Carlo estimates

Differentiable Monte Carlo estimates

Pixel filter or BRDF

$\mathbf{x}_i$
REPARAMETERIZE INTEGRALS?

Non-differentiable Monte Carlo estimates

Pixel filter or BRDF

Differentiable Monte Carlo estimates

Change of variables

$\mathbf{x}_i$
RESULTS

Ours

Reference
(Finite differences)

Without changes of variables
RESULTS

Ours

Reference (Finite differences)

Without changes of variables
RESULTS

Ours

Reference
(Finite differences)

Without changes of variables
RESULTS

Ours

Reference (Finite differences)

Without changes of variables

Realistic Image Synthesis SS2024

(Realistic Image Synthesis SS2024)
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

Realistic Image Synthesis SS2024
Dealing with discontinuities is not enough.

Want to propagate derivative information through complex simulations with millions of differentiable parameters.
DIFFERENTIAL MONTE CARLO

“Monte-Carlo calculation of derivatives of functionals from the solution of the transfer equation according to the parameters of the system”
G. A. Mikhailov, Novosibirsk, July 1966

“Monte Carlo Analysis of Reactivity Coefficients in Fast Reactors, General Theory and Applications”
L.B. Miller, Argonne Natl. Laboratory, March 1967
DIFFERENTIATING THE RENDERING EQN

\[ L_0(x, \omega) = L_e(x, \omega) + \int_{S^2} L_i(x, \omega') f_s(x, \omega, \omega') \cos \theta \, d\omega' \]
\[ L_0(x, \omega) = L_e(x, \omega) + \int_{S^2} L_i(x, \omega') f_s(x, \omega, \omega') \cos \theta \, d\omega' \]
DIFFERENTIATING THE RENDERING EQN

\[ L_o(x, \omega) = L_e(x, \omega) + \int_{S^2} L_i(x, \omega') f_s(x, \omega, \omega') \cos \theta \, d\omega' \]
DIFFERENTIATING THE RENDERING EQN

\[ L_0(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_{S^2} L_i(\mathbf{x}, \omega') f_s(\mathbf{x}, \omega, \omega') \cos \theta \, d\omega' \]

\[ \partial_x \] derivative wrt. scene parameters
DIFFERENTIATING THE RENDERING EQN

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

\(\partial_x\) derivative wrt. scene parameters
\[ \partial_x L_o(x, \omega) = \partial_x L_e(x, \omega) + \int_{S^2} L_i(x, \omega') f_s(x, \omega, \omega') \cos \theta \, d\omega' \]
DIFFERENTIATING THE RENDERING EQN

\[
\frac{\partial}{\partial x} L_0(x, \omega) = \frac{\partial}{\partial x} L_e(x, \omega) + \int_{S^2} L_i(x, \omega') \frac{\partial}{\partial x} f_s(x, \omega, \omega')
\]
\[
\partial_x L_0(x, \omega) = \partial_x L_e(x, \omega) \\
+ \int_{S^2} L_i(x, \omega') \partial_x f_s(x, \omega, \omega') \\
+ \partial_x L_i(x, \omega') f_s(x, \omega, \omega') \cos \theta \, d\omega'
\]
DIFFERENTIATING THE RENDERING EQN

\[ \frac{\partial}{\partial x} L_0(x, \omega) = \left[ L_e(x, \omega) + \int [L_i(x, \omega', w) + \frac{\partial}{\partial x} L_i(x, \omega') f_s(x, \omega, \omega')] \cos \theta \, d\omega' \right] \]

**TL;DR**

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

Differential radiance “scatters” like normal radiance.
Realistic Image Synthesis SS2024
\[
\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial y}{\partial x}
\]
Gradients

Derivative wrt. parameters

Derivative wrt. objective

Dot product (discrete)
Gradients

Derivative wrt. parameters

Derivative wrt. objective

Dot product (discrete)
Realistic Image Synthesis SS2024

Gradients

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

Derivative wrt. parameters

Derivative wrt. objective

Dot product (discrete)
WHAT’S WRONG WITH THIS?

1MPix rendering &
1M parameters:
WHAT’S WRONG WITH THIS?

1MPix rendering & 1M parameters:

\[
\frac{\partial y}{\partial x} \in \mathbb{R}^{1000000 \times 1000000}
\]

(~3.6 TiB)
DIRECTIONALITY OF DIFFERENTIATION

Forward mode

\[ y = x_0 \cdot x_1 + x_2 \]
DIRECTIONALITY OF DIFFERENTIATION

Forward mode

$$y = x_0 \cdot x_1 + x_2$$
DIRECTIONALITY OF DIFFERENTIATION

Forward mode

\[ y = x_0 \cdot x_1 + x_2 \]

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

Gradient
DIRECTIONALITY OF DIFFERENTIATION

\[ y = x_0 \cdot x_1 + x_2 \]
DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

\[ y = x_0 \cdot x_1 + x_2 \]
DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

\[ y = x_0 \cdot x_1 + x_2 \]

Gradient

\[ \frac{\partial y}{\partial x_0}, \frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2} \]
DIRECTIONALITY OF DIFFERENTIATION

Reverse mode

\[ y = x_0 \cdot x_1 + x_2 \]

Program execution

Differentiation

\[ \frac{\partial y}{\partial x_0} \]
\[ \frac{\partial y}{\partial x_1} \]
\[ \frac{\partial y}{\partial x_2} \]
Autodiff-based differentiable rendering

Scene parameters $x \in \mathcal{X}$ → Rendering algorithm
Autodiff-based differentiable rendering

Scene parameters $x \in \mathcal{X}$

Gradients

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

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

SIGGRAPH 2020
MOTIVATION: ADJOINT SENSITIVITY METHOD

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

Pontryagin et al. 1962

Pontryagin et al. 1962

The Mathematical Theory of Optimal Processes

L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, E. F. Mishchenko

Recipients of the 1961 State Prize for Science and Technology

Authorized Translation from the Russian

Translator: H. N. Terras
Edition: L. W. Neustadt

Interscience Publishers
a division of John Wiley & Sons
New York • London • Sydney

Realistic Image Synthesis SS2024
MOTIVATION: ADJOINT SENSITIVITY METHOD

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

Pontryagin et al. 1962
MOTIVATION: ADJOINT SENSITIVITY METHOD

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

Pontryagin et al. 1962
"ADJOINT" – THAT SOUNDS FAMILIAR!

Bidirectional Estimators for Light Transport
Veach & Guibas, 1994

\[ \langle Oa, b \rangle = \langle a, Ob \rangle \]
(Underlying principle: self-adjoint operators)
Derivatives projected into the scene
Gradients

- Deriv. from objects
- Deriv. from sensor
- Product integral
\[ f_s : (\omega_i, \omega_0, x_1, x_2, \ldots) \rightarrow \mathbb{R} \]
\[ f_s : (\omega_i, \omega_0, x_1, x_2, \ldots) \rightarrow \mathbb{R} \]
Normal rendering
Normal rendering

- Transporting from sensor/light may yield lower variance.
Normal rendering

- Transporting from sensor/light may yield lower variance.
Normal rendering

- Transporting from sensor/light may yield lower variance.

Differentiable rendering
ANOTHER PERSPECTIVE

Normal rendering

- Transporting from sensor/light may yield lower variance.

Differentiable rendering

\[ \partial \text{ from objects} \quad \rightarrow \quad \partial \text{ from sensor} \]
ANOTHER PERSPECTIVE

Normal rendering

- Transporting from sensor/light may yield lower variance.

Differentiable rendering

- Transporting from objects is completely impractical.
Surface texture optimization

Initial state

Target state
Realistic Image Synthesis SS2024

Optimized texture

Target
Optimized texture

Target
Surface BSDF optimization

Reference
Initial state
Ours (biased I)
Autodiff-based
Ours (biased I+II)

Method
Autodiff-based
Ours
Ours (biased II)
Ours (biased I)
Ours (biased I+II)

Time (min)

0 20 40 60 80

10^{-4}

10^{-5}
Surface BSDF optimization

![Reference](image1.png) ![Initial state](image2.png) ![Ours (biased I)](image3.png) ![Autodiff-based](image4.png) ![Ours (biased II)](image5.png) ![Ours (biased I)](image6.png) ![Ours (biased I+II)](image7.png)

**Objective function (log scale)**

- Method: Autodiff-based, Ours, Ours (biased II), Ours (biased I), Ours (biased I+II)

**Iteration count**

- 0 20 40 60 80 100 120

**Time (min)**

- 0 20 40 60 80

**Realistic Image Synthesis SS2024**
Volume density optimization

Mitsuba 2 (AD-based)  Radiative Backprop. (biased I + II)  Target
Volume density optimization

Mitsuba 2 (AD-based)  Radiative Backprop. (biased I + II)  Target
Volume density optimization

Reference

Initial state

Ours (biased I)

Autodiff-based

Ours (biased I+II)

Ours

Reference

Initial state

Ours (biased I)

Autodiff-based

Ours (biased I+II)

Ours

Iteratio\(\text{count}\) (log scale)

\(0.000\)

\(0.005\)

\(0.010\)

\(0.015\)

\(0.020\)

\(0.025\)

\(0.030\)

\(10^1\)

\(10^2\)

\(10^3\)

\(10^4\)

Obj\(\text{ecti}\)\(\text{ve}\) function

Time (s, log scale)

\(0.000\)

\(0.005\)

\(0.010\)

\(0.015\)

\(0.020\)

\(0.025\)

\(0.030\)

\(10^1\)

\(10^2\)

\(10^3\)

\(10^4\)

Method

Autodiff-based

Ours

Ours (biased II)

Ours (biased I)

Ours (biased I+II)

RADIATIVE BACKPROPAGATION
Relative speedups vs autodiff-based

Surface texture optimization

Samples per pixel | Ours | Ours (biased II) | Ours (biased I) | Ours (biased I+II)
--- | --- | --- | --- | ---
4 | 180× | 230× | 205× | 257×
512 | 40× | 50× | 45× | 56×

Volume density optimization

Samples per pixel | Ours | Ours (biased II) | Ours (biased I) | Ours (biased I+II)
--- | --- | --- | --- | ---
4 | 402× | 486× | 546× | 608×
512 | 454× | 511× | 786× | 992×

Relative speedup
- Ours
- Ours (biased II)
- Ours (biased I)
- Ours (biased I+II)
TL;DR

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

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

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

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

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

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

TZU-MAO LI
POSTDOCTORAL RESEARCHER
MIT CSAIL, Cambridge

(joining UCSD as an assistant professor in 2021)
ACKNOWLEDGEMENTS

SHUANG ZHAO
ASSISTANT PROFESSOR
University of California, Irvine

WENZEL JAKOB
ASSISTANT PROFESSOR
EPFL, Lausanne, Switzerland

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