Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR

YU-CHEN WANG, University of California, Irvine, USA
CHRIS WYMAN, NVIDIA, USA
LIFAN WU, NVIDIA, USA
SHUANG ZHAO, University of California, Irvine, USA

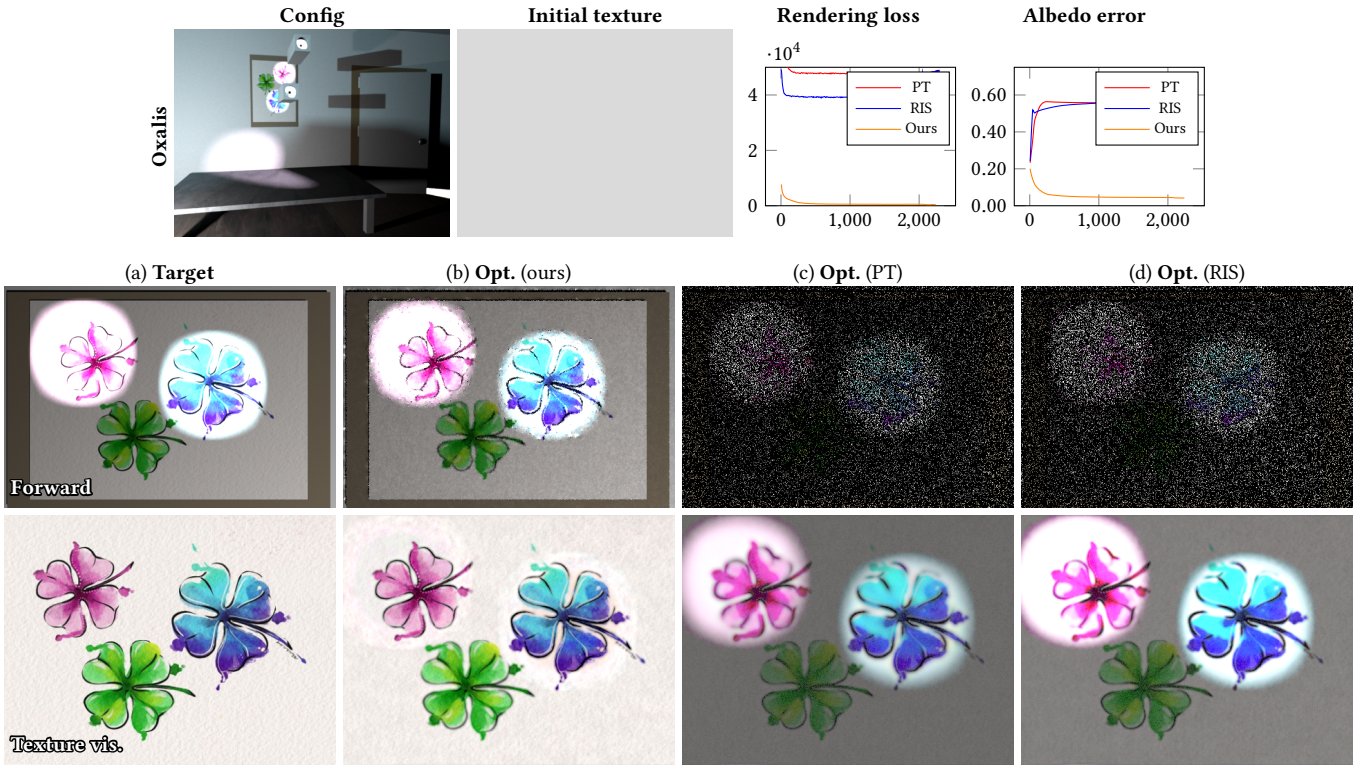


Fig. 1. Our new technique reuses temporal data in the context of physics-based inverse direct illumination. During inverse-rendering, when optimizing a scene iteratively using gradient-based methods such as stochastic gradient descent or Adam [Kingma and Ba 2014], we reuse light samples spatially and temporally (across iterations), offering significantly cleaner forward rendering and gradient estimates than baseline methods without reuse. This example uses the "Oxalis" painting lit by several bright spot lights and a dim fill light. We optimize the spatially varying albedo (initialized using the gray texture shown) of the painting. (c, d) The baseline methods PT (**B.1**) and RIS (**B.2**) produce high variance—especially in dark areas only lit by the fill light, causing highly biased results. (b) Our method, on the other hand, enjoys significantly more accurate reconstructions.

Recently, great progress has been made in physics-based differentiable rendering. Existing differentiable rendering techniques typically focus on *static* scenes, but during inverse rendering—a key application for differentiable rendering—the scene is updated *dynamically* by each gradient step. In this paper, we take a first step to leverage temporal data in the context of inverse direct illumination. By adopting reservoir-based spatiotemporal resampled importance resampling (ReSTIR), we introduce new Monte Carlo estimators for both interior and boundary components of differential direct illumination integrals. We also integrate ReSTIR with antithetic sampling to further improve its effectiveness. At equal frame time, our methods produce gradient estimates with up to 100× lower relative error than baseline methods. Additionally, we propose an inverse-rendering pipeline that incorporates these estimators and provides reconstructions with up to 20× lower error.

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

Additional Key Words and Phrases: Differentiable rendering, inverse rendering, importance sampling, sample reuse, ReSTIR

Authors' addresses: Yu-Chen Wang, yuchew23@uci.edu, University of California, Irvine, USA; Chris Wyman, chris.wyman@acm.org, NVIDIA, USA; Lifan Wu, lifanw@nvidia.com, NVIDIA, USA; Shuang Zhao, shz@ics.uci.edu, University of California, Irvine, USA.

0730-0301/2023/12-ART214
https://doi.org/10.1145/3618331

ACM Reference Format:
Yu-Chen Wang, Chris Wyman, Lifan Wu, and Shuang Zhao. 2023. Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR. *ACM Trans. Graph.* 42, 6, Article 214 (December 2023), 17 pages. https://doi.org/10.1145/3618331

1 INTRODUCTION

Physics-based *forward rendering* synthesizes photorealistic images of fully described 3D scenes. Given its importance, decades of forward rendering research have led to numerous techniques capable of accurately reproducing complex lighting, including soft shadows, environmental illumination, and interreflection.

Inverse rendering instead infers parameters such as object shapes and reflectance from images of the scene. Due to the complexity of forward rendering, analytical inversions are generally infeasible. Typically, inverse rendering relies on numerical optimizations to find scene parameters that minimize differences (according to some *rendering loss*) between target and forward-rendered images.

Efficient inverse-rendering optimizations using gradient-based methods (e.g., stochastic gradient descent or Adam [Kingma and Ba 2014]) require differentiating physics-based forward rendering with respect to arbitrary scene parameters—a process known as physics-based *differentiable rendering*.

Recently, great progress has been made in differentiable rendering. Derivations for differentiating the rendering equation [Li et al. 2018], radiative transfer [Zhang et al. 2019], and path integrals [Zhang et al. 2020, 2021b] have demonstrated that physics-based light transport is generally differentiable. Building on these formulations, new Monte Carlo methods (e.g., [Zeltner et al. 2021; Zhang et al. 2021a]) and differentiable computations (e.g., [Nimier-David et al. 2020; Vicini et al. 2021]) further improved efficiency.

Unfortunately, most differentiable rendering techniques focus on *static* scenes. However, during inverse-rendering optimizations, the scene is updated *dynamically* by each gradient step. Exploiting temporal consistency between steps to improve efficiency has remained largely unexplored.

In this paper, we take a first step to leverage temporal data in the context of physics-based inverse direct illumination. Specifically, we adopt *reservoir-based spatiotemporal importance resampling* (ReSTIR) [Bitterli et al. 2020]—a powerful technique for real-time direct lighting in dynamic scenes—to significantly improve the efficiency of differentiable direct illumination under complex lighting conditions.

Concretely, we make the following contributions:

- We introduce new ReSTIR-based Monte Carlo estimators for both *interior* and *boundary* components of differential direct-illumination integrals (§4 and §5).
- We discuss the interplay between ReSTIR and pixel-level antithetic sampling [Yu et al. 2022] and introduce a simple technique to improve the effectiveness of antithetic sampling (§6).
- We propose an efficient inverse-rendering pipeline that utilizes our ReSTIR estimators (§7). Additionally, we discuss how multi-view configurations can be supported efficiently by: (i) preserving reservoirs across multiple iterations; (ii) using slightly biased gradient estimates with respect to object shapes.

To validate our technique, we compare our gradient estimates with those from prior unbiased methods (Figure 6). Lastly, we demonstrate our effectiveness using several differentiable-rendering (Figures 7–9) and inverse-rendering (Figures 1, 10–14) examples.

2 RELATED WORKS

We review works in forward and differentiable rendering of surfaces.

Forward rendering of dynamic scenes. Forward rendering played a major role in graphics for the past 50 years, with significant bodies of work on predictive and film rendering (e.g., Pharr et al. [2016]) and interactive techniques targeting gaming and visualization (e.g., Akenine-Moller et al. [2018]). There are common simplifications that reduce cost by assuming scenes remain static or have fixed animations.

Until recently, forward rendering for truly dynamic scenes was often quite biased, with image appeal trumping predictive convergence properties as rasterization efficiency pushed renderers to use approximations like ambient occlusion [Bavoil and Sainz 2009], image-space ray tracing [McGuire and Mara 2014], probe-based global light [McGuire et al. 2017], and precomputed lightmaps [Seyb et al. 2020] rather than more accurate light transport.

While researchers [Laine et al. 2020; Liu et al. 2019] have demonstrated differentiable rasterization, it remains limited to fairly simple scenes involving primary visibility.

Recent ray tracing hardware [Burgess 2020] offers an opportunity to use unbiased, physics-based light transport while dynamically changing a scene each frame, but current per-pixel ray budgets have limited use in many renderers to improve shadows [Heitz et al. 2018], reflections [Deligiannis and Schmid 2019], and diffuse global illumination [Majercik et al. 2019].

Resampling and ReSTIR. Recent work builds on importance resampling [Talbot et al. 2005] to amortize ray costs, aiming to minimize rendering overhead in dynamic scenes by spatiotemporally reusing sampled rays [Bitterli et al. 2020]. This reservoir-based spatiotemporal importance sampling, aka ReSTIR, observes that spatially- and temporally-neighboring integrals have similar forms. With careful reweighting, samples can be reused despite being pulled from different integration domains [Lin et al. 2022]. This is a form of filtering, similar to post-process denoisers [Schied et al. 2017], but applied to the sampling pdfs. While it assumes signals change smoothly, edge-stopping functions and similar image-processing tricks [Durand and Dorsey 2002] can be applied to better handle sharper discontinuities.

While we build on Bitterli et al.'s [2020] early formulation to efficiently sample complex many-light dynamic scenes, ReSTIR can also reuse complex paths including diffuse global illumination [Ouyang et al. 2021], volumetric media [Lin et al. 2021], and more complex multi-bounce light transport [Lin et al. 2022].

Physics-based differentiable rendering. A main challenge in developing general-purpose differentiable renderers has been differentiating with respect to scene geometry, which generally requires calculating additional boundary integrals. To address this, Li et al. [2018] introduced a Monte Carlo edge-sampling method giving unbiased estimates of boundary integrals, but it requires detecting object silhouettes, which can be expensive. Later, reparameterization-based

ALGORITHM 1: Streaming resampled importance sampling (RIS) [Bitterli et al. 2020]

```
UpdateReservoir(r, x, w)
  Input: A reservoir r, a new candidate x with the weight w
begin
    r.w_sum ← r.w_sum + w;
    if rand() < w/r.w_sum then
        r.y ← x;
    end
    r.M ← r.M + 1;
end
StreamingRIS(M)
  Input: Number of candidates M ≥ 1
begin
    r.w_sum ← 0;
    r.M ← 0;
    for m = 1, 2, ..., M do
        Draw x_m ~ p_c;
        UpdateReservoir(r, x, p̂(x_m)/p_c(x_m)) ; // Updating r
    end
    r.W ← r.w_sum/(p̂(r.y) r.M);
    return r;
end
```

methods [Loubet et al. 2019; Bangaru et al. 2020] were introduced to avoid computing boundary integrals. Further, by differentiating Veach's path integrals [1997], Zhang et al. [2020] formulated differential path integrals, extending Monte Carlo differentiable rendering beyond unidirectional path tracing.

Additionally, several Monte Carlo sampling methods [Zeltner et al. 2021; Zhang et al. 2021a; Yan et al. 2022; Yu et al. 2022] and efficient differentiation techniques [Nimier-David et al. 2020; Vicini et al. 2021] have improved the performance of differentiable rendering. All these methods, unfortunately, focus on static scenes.

Below, based on Zhang et al.'s formulation, we introduce new methods for efficient differentiable rendering of dynamic scenes.

Concurrent works. The idea of adopting animation-rendering techniques for inverse rendering has recently been explored by two concurrent works. Specifically, Chang et al. [2023] introduced a parameter-space ReSTIR method that focuses on material reconstruction and is closely related to our approach for interior integrals (§4). On the other hand, by neglecting boundary integrals (§5), this technique does not offer variance reduction for shape optimizations.

Another concurrent work applies recursive control variate for inverse rendering [Nicolet et al. 2023]. This technique applies variance reduction for forward rendering only and is largely orthogonal to our technique.

3 PRELIMINARIES

3.1 Direct Illumination

Under a path-integral formulation [Veach 1997; Pauly et al. 2000] of the rendering equation for surface-only direct illumination, the intensity I of a pixel is given by

$$I = \int_\Omega f(\bar{\boldsymbol{x}})\, \mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{1}$$

where $\Omega = \mathcal{M}^3$ (with $\mathcal{M}$ the union of all surfaces) is the one-bounce **path space** containing **light paths** $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2)$ with $\boldsymbol{x}_0$ on the light and $\boldsymbol{x}_2$ on the detector; μ is the area-product measure given by $\mathrm{d}\mu(\bar{\boldsymbol{x}}) = \prod_{n=0}^{2} \mathrm{d}A(\boldsymbol{x}_n)$; and f is the **measurement contribution function** defined as

$$f(\bar{\boldsymbol{x}}) = L_\mathrm{e}(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1)\, f_\mathrm{s}(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2) \\ G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2)\, W_\mathrm{e}(\boldsymbol{x}_1 \to \boldsymbol{x}_2). \tag{2}$$

Here L_e and W_e are the **source emission** and **detector response** functions, respectively; f_s is the **bidirectional scattering distribution function** (BSDF); and G denotes the (visibility-aware) **geometric term**.

The pixel intensity I in Eq. (1) is often estimated by (i) tracing a *camera ray* from $\boldsymbol{x}_2$ that intersects the scene at surface $\boldsymbol{x}_1 \in \mathcal{M}$, and (ii) sampling point $\boldsymbol{x}_0$—which we term a **light vertex**—on the surface of a light source. This gives a Monte Carlo estimator:

$$\langle I \rangle = \frac{W_\mathrm{e}(\boldsymbol{x}_1 \to \boldsymbol{x}_2)\, G(\boldsymbol{x}_1 \leftrightarrow \boldsymbol{x}_2)}{p(\boldsymbol{x}_1, \boldsymbol{x}_2)} \\ \frac{L_\mathrm{e}(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, f_\mathrm{s}(\boldsymbol{x}_0 \to \boldsymbol{x}_1 \to \boldsymbol{x}_2)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1)}{p(\boldsymbol{x}_0 \,|\, \boldsymbol{x}_1, \boldsymbol{x}_2)}, \tag{3}$$

for $p(\boldsymbol{x}_1, \boldsymbol{x}_2)$ the joint probability density of sampling $\boldsymbol{x}_1, \boldsymbol{x}_2$ and $p(\boldsymbol{x}_0 \,|\, \boldsymbol{x}_1, \boldsymbol{x}_2)$ the conditional probability density of drawing $\boldsymbol{x}_0$.

In practice, the camera ray from $\boldsymbol{x}_2$ to $\boldsymbol{x}_1$ can be sampled using standard ray tracing. In the rest of this paper, we assume the perspective pinhole camera model. In this case, $\boldsymbol{x}_2$ becomes fixed at the center of projection and $\boldsymbol{x}_1$ is obtained by tracing a ray from $\boldsymbol{x}_2$ through a randomly selected point on the image plane.

With $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ obtained, importance sampling the light vertex $\boldsymbol{x}_0$ remains challenging—especially under complex illumination and visibility. Bitterli et al. [2020] recently introduced a technique that efficiently samples $\boldsymbol{x}_0$. We briefly revisit this work in §3.2 and build on it later.

3.2 **Re**servoir-based **S**patio**T**emporal **I**mportance **R**esampling (ReSTIR) for Direct Illumination

Resampled importance sampling. Monte Carlo estimation of

$$I = \int h(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}, \tag{4}$$

requires generating samples $\boldsymbol{x}$ with probability density proportional to some function $\hat{p}$ (e.g., $\hat{p} = h$ when h is nonnegative). But for complicated $\hat{p}$, analytically sampling from this probability density can be difficult.

Instead, *resampled importance sampling* (RIS) [Talbot et al. 2005] samples from a simple randomized distribution approximating $\hat{p}$. First, RIS draws M independent candidates $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\}$ from some other distribution p_c. Then, one sample $\boldsymbol{y}$ is selected from the M candidates with (discrete) probability $\mathbb{P}[\boldsymbol{y} = \boldsymbol{x}_m]$ proportional to

$$\mathrm{w}(\boldsymbol{x}_m) := \frac{\hat{p}(\boldsymbol{x}_m)}{p_\mathrm{c}(\boldsymbol{x}_m)}, \tag{5}$$

for $m = 1, 2, \ldots, M$. This gives a one-sample RIS estimator of Eq. (4):

$$\langle I \rangle^M_{\text{ris}} = \frac{h(\boldsymbol{y})}{\hat{p}(\boldsymbol{y})} \left(\frac{1}{M} \sum_{m=1}^{M} \mathrm{w}(\boldsymbol{x}_m) \right). \tag{6}$$

Naïve implementations of Eq. (6) generate and store all M candidates $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ before selecting final sample $\boldsymbol{y}$. This can be inefficient. Bitterli et al. [2020] proposed a *streaming* version using *weighted reservoir sampling* (WRS) [Chao 1982] offering significantly better efficiency.

Algorithm 1 summarizes this method, which processes a candidate stream and maintains a *reservoir* structure `r`. After handling candidate $\boldsymbol{x}_m$ inside `UpdateReservoir()`, reservoir `r` stores the following data: (i) a sum of weights $\mathtt{r}.\mathrm{w}_{\text{sum}} = \sum_{i=1}^{m} \mathrm{w}(x_i)$; (ii) a count of candidates processed $\mathtt{r}.M = m$; and (iii) a sample $\mathtt{r}.\boldsymbol{y}$ randomly drawn from prior candidates $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ with selection probability $\mathbb{P}[\mathtt{r}.\boldsymbol{y} = \boldsymbol{x}_i] = \mathrm{w}(\boldsymbol{x}_i)/\mathtt{r}.\mathrm{w}_{\text{sum}}$ for all $i = 1, 2, \ldots, m$. One can easily evaluate the RIS estimator in Eq. (6) using the reservoir `r` output from `StreamingRIS()` using $\langle I \rangle^M_{\text{ris}} = h(\mathtt{r}.\boldsymbol{y})\, \mathtt{r}.W$.

Reservoir-based spatiotemporal importance resampling. To estimate direct illumination using Eq. (3), after generating the camera ray $\boldsymbol{x}_2 \to \boldsymbol{x}_1$, RIS (Algorithm 1) can be used to select the light vertex $\boldsymbol{x}_0$ by setting the target function $\hat{p}$ to the measurement contribution of the path $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2)$. However, under complex illumination conditions, RIS needs to use many candidates (i.e., large M) to achieve high efficiency, which is typically impractical.

To address this problem, Bitterli et al. [2020] introduced *reservoir-based spatiotemporal importance resampling* (ReSTIR) that amortizes sampling costs by reusing spatial and temporal candidates via iterative applications of RIS.

Specifically, when rendering a pixel, ReSTIR first generates a reservoir using RIS (Algorithm 1). Then, the reservoir is merged with reservoirs from various neighboring pixels (i.e., spatial reuse) as well as prior frames (i.e., temporal reuse). To ensure *unbiasedness*, Bitterli et al. [2020] proposed recomputing $\mathtt{r}.W$ after determining the sample $\mathtt{r}.\boldsymbol{y}$, as expressed in Algorithm 2.[1] This is because the target function $\hat{p}^{(q)}$ for a pixel q generally differs from those $\hat{p}^{(q_i)}$ for its spatial/temporal neighbors q_i. Further, precautions must be taken when evaluating $\hat{p}^{(q_i)}(\mathtt{r}.\boldsymbol{y})$ (Line 9 of Algorithm 2) when q_i is from the previous frame (i.e., via temporal reuse) and the scene geometry varies between the previous and the current frames. Specifically, the evaluation should use the scene geometry of the previous frame with the light sample $\mathtt{r}.\boldsymbol{y}$ moved accordingly (by, for example, using motion vectors).

Algorithm 2 has recently been generalized by Lin et al. [2022] to allow reusing samples from a wider range of domains (including full path samples from varying domains).

3.3 Differential Direct Illumination

In physics-based differentiable rendering, Zhang et al. [2020; 2021b] recently introduced a formulation of differential path integrals as well as several associated Monte Carlo estimators. Below, we briefly describe how these apply to direct illumination.

[1] Our Algorithm 2 is identical to Algorithm 6 in Bitterli et al.'s [2020] work. Please refer to their paper for a proof of unbiasedness.

ALGORITHM 2: Combining reservoirs from spatial and temporal neighbors [Bitterli et al. 2020]

```
1  SpatiotemporalReuse(r, q, Q', P')
   Input: Reservoir r for pixel q; reservoirs Q' = {r_1, ..., r_k} from
          pixels P' = {q_1, ..., q_k} to combine
   Output: One combined reservoir
2  begin
3      for i = 1, ..., k do
4          UpdateReservoir(r, r_i.y, p̂^(q)(r.y) r.W r.M)
5      end
6      r.M ← Σ_{i=1}^{k} r_i.M;
7      Z ← 0;
8      for i = 1, ..., k do
9          if p̂^(q_i)(r.y) > 0 then
10             Z ← Z + r_i.M;
11         end
12     end
13     r.W ← r.w_sum / (p̂^(q)(r.y) Z);
14     return r;
15 end
```

When scene geometry $\mathcal{M}$ evolves with parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ (for integer $m_{\boldsymbol{\theta}} \geq 1$), Zhang et al. proposed a *material-form reparameterization* to facilitate differentiating Eq. (1) with respect to $\boldsymbol{\theta}$. This parameterization introduces a **motion** $\mathtt{X}$ such that for any $\boldsymbol{\theta}$, $\mathtt{X}(\cdot, \boldsymbol{\theta})$ is a differentiable one-to-one mapping from some predetermined **reference configuration** $\mathcal{B}$ (independent of parameters $\boldsymbol{\theta}$) to the scene geometry $\mathcal{M}(\boldsymbol{\theta})$. To distinguish points in the ordinary geometry from those in the reference configuration, we term any $\boldsymbol{x} \in \mathcal{M}(\boldsymbol{\theta})$ as **spatial points** and $\boldsymbol{p} \in \mathcal{B}$ as **material points**.

The mapping $\mathtt{X}(\cdot, \boldsymbol{\theta})$ induces a path-level map $\bar{\mathtt{X}}(\cdot, \boldsymbol{\theta})$ that takes a one-bounce **material path** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ to an ordinary path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2)$ where $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \boldsymbol{\theta})$ for $n = 0, 1, 2$.

Applying the change of variable of $\bar{\boldsymbol{x}} = \bar{\mathtt{X}}(\bar{\boldsymbol{p}}, \boldsymbol{\theta})$ to Eq. (1) yields its *material-form* variant:

$$I = \int_{\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{7}$$

where $\hat{\Omega} := \mathcal{B}^3$ is termed the **material path space**, and

$$\hat{f}(\bar{\boldsymbol{p}}) := f(\bar{\boldsymbol{x}}) \left\| \frac{\mathrm{d}\mu(\bar{\boldsymbol{x}})}{\mathrm{d}\mu(\bar{\boldsymbol{p}})} \right\|, \tag{8}$$

with $\left\| \mathrm{d}\mu(\bar{\boldsymbol{x}})/\mathrm{d}\mu(\bar{\boldsymbol{p}}) \right\| = \prod_{n=0}^{2} \left\| \mathrm{d}A(\boldsymbol{x}_n)/\mathrm{d}A(\boldsymbol{p}_n) \right\|$ being the Jacobian determinant capturing this change of variable.

Zhang et al. showed differentiating Eq. (7) with respect to $\boldsymbol{\theta}$ equals

$$\frac{\mathrm{d}I}{\mathrm{d}\boldsymbol{\theta}} = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \hat{f}(\bar{\boldsymbol{p}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \Delta\hat{f}_K(\bar{\boldsymbol{p}}) \left(\boldsymbol{n}_\partial(\boldsymbol{p}_K)^\top \frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\boldsymbol{\theta}} \right) \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}}, \tag{9}$$

where the interior component comes from differentiating the integrand $\hat{f}$ of Eq. (7).

The boundary integral instead covers **material boundary path space** $\partial\hat{\Omega}$, is unique to differentiable rendering, and usually depends on scene parameters $\boldsymbol{\theta}$. For direct lighting, $\partial\hat{\Omega}$ comprises **material boundary paths** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ with exactly one **boundary segment** $\overline{\boldsymbol{p}_K\,\boldsymbol{p}_{K+1}}$ ($K \in \{0, 1\}$) such that corresponding spatial point $\boldsymbol{x}_K = \mathsf{X}(\boldsymbol{p}_K, \boldsymbol{\theta})$ is a jump discontinuity of the mutual visibility function $\mathbb{V}(\boldsymbol{x}_K \leftrightarrow \boldsymbol{x}_{K+1})$ when fixing $\boldsymbol{x}_{K+1} = \mathsf{X}(\boldsymbol{p}_{K+1}, \boldsymbol{\theta})$. Further, $\boldsymbol{n}_\partial(\boldsymbol{p}_K)^\top \in \mathbb{R}^{1\times 3}$ is the unit normal of the visibility boundary at $\boldsymbol{p}_K$, and $\mathrm{d}\boldsymbol{p}_K/\mathrm{d}\boldsymbol{\theta} \in \mathbb{R}^{3\times m_{\boldsymbol{\theta}}}$ is the boundary gradient. The differential measure $\dot{\mu}$ over $\partial\hat{\Omega}$ is given by $\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}) = \mathrm{d}\ell(\boldsymbol{p}_K)\prod_{n\neq K}\mathrm{d}A(\boldsymbol{p}_n)$ with ℓ being the curve-length measure.

Choice of reference configuration. In practice, to estimate gradients at $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ we use $\mathcal{B} = \mathcal{M}(\boldsymbol{\theta}_0)$ as the reference configuration for convenience. In this case, the material path space $\hat{\Omega}$ coincides with ordinary path space Ω, and mapping $\mathsf{X}(\cdot, \boldsymbol{\theta})$ reduces to the identity. In other words, a material point $\boldsymbol{p}$ is essentially the *detached* copy of its spatial representation $\boldsymbol{x} = \mathsf{X}(\boldsymbol{p}, \boldsymbol{\theta})$ at $\boldsymbol{\theta} = \boldsymbol{\theta}_0$.

We note that while the Jacobian $\|\mathrm{d}\mu(\bar{\boldsymbol{x}})/\mathrm{d}\mu(\bar{\boldsymbol{p}})\|$ in Eq. (8) is one at $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, its gradient with respect to $\boldsymbol{\theta}$ generally remains nonzero.

3.4 Inverse Rendering and Radiative Backpropagation

Inverse-rendering optimizations. Optimization-based inverse rendering, or analysis by synthesis, infers scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ by minimizing a predetermined rendering loss $\mathcal{L}$ (e.g., L^1 or L^2) between rendered images[2] $\boldsymbol{I} \in \mathbb{R}^{m_{\boldsymbol{I}}}$ with $m_{\boldsymbol{I}}$ pixels and user-specified targets $\boldsymbol{I}_0 \in \mathbb{R}^{m_{\boldsymbol{I}}}$:

$$\arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{I}(\boldsymbol{\theta}); \boldsymbol{I}_0). \tag{10}$$

In practice, additional *regularization* improves robustness. We omit such details as they are largely orthogonal to our work.

Minimizing Eq. (10) with gradient-based methods such as stochastic gradient descent (SGD) or Adam [Kingma and Ba 2014] requires differentiating the rendering loss $\mathcal{L}$ with respect to parameters $\boldsymbol{\theta}$. According to the chain rule, the gradient $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L} := \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ satisfies

$$\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L} = (\partial_{\boldsymbol{I}}\mathcal{L})\,(\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}), \tag{11}$$

where $\partial_{\boldsymbol{I}}\mathcal{L} := \partial\mathcal{L}/\partial\boldsymbol{I}$ on the right comes from differentiable evaluation of the loss $\mathcal{L}$, and $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I} := \mathrm{d}\boldsymbol{I}/\mathrm{d}\boldsymbol{\theta}$ is given by Eq. (9).

Radiative backpropagation. When the number of parameters $m_{\boldsymbol{\theta}}$ and pixels $m_{\boldsymbol{I}}$ are large, storing the rendered image $\boldsymbol{I}$ and full computational graph (for reverse-mode automatic differentiation [Griewank and Walther 2008]) can lead to performance and storage problems. To this end, previous methods [Nimier-David et al. 2020; Vicini et al. 2021] apply the chain rule in Eq. (11) at the path-integral level. We do this by left-multiplying the gradient $\partial_{\boldsymbol{I}}\mathcal{L}$ on both sides of Eq. (9), yielding:

$$\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L} = \underbrace{\int_{\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L})[q]\left(\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\hat{f}^{(q)}(\bar{\boldsymbol{p}})\right)\mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L})[q]\,\Delta\hat{f}_K^{(q)}(\bar{\boldsymbol{p}})\left(\boldsymbol{n}_\partial(\boldsymbol{p}_K)^\top\frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\boldsymbol{\theta}}\right)\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}}, \tag{12}$$

where q denotes the pixel index where the path $\bar{\boldsymbol{p}}$ contributes, and $(\partial_{\boldsymbol{I}}\mathcal{L})[q] \in \mathbb{R}$ indicates the value in $\partial_{\boldsymbol{I}}\mathcal{L}$ at pixel q (where $\boldsymbol{I}$ and $\partial_{\boldsymbol{I}}\mathcal{L}$ have the same shape). Also, $\hat{f}^{(q)}$ denotes the material measurement contribution of the path $\bar{\boldsymbol{p}}$ to pixel q, and $\Delta\hat{f}_K^{(q)}$ indicates the difference in $\hat{f}^{(q)}$ across visibility boundaries (as per §3.3).

In §4 and §5, we derive our ReSTIR algorithms that efficiently estimate the interior and boundary integrals on the right-hand side of Eq. (12) for optimization-based inverse rendering.

4 RESTIR FOR INTERIOR INTEGRALS

We now describe our ReSTIR-based method to efficiently estimate the (vector-valued) interior component of Eq. (12):

$$(\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})_{\text{int}} := \int_{\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L})[q]\left(\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\hat{f}^{(q)}(\bar{\boldsymbol{p}})\right)\mathrm{d}\mu(\bar{\boldsymbol{p}}). \tag{13}$$

When computing gradients at a fixed $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, as discussed in §3.3, we set the reference configuration to $\mathcal{B} = \mathcal{M}(\boldsymbol{\theta}_0)$. Note that during inverse-rendering optimization, $\boldsymbol{\theta}_0$ varies at each gradient step.

Overview. We introduce our differentiable RIS estimator in §4.1 followed by the full ReSTIR version—our main contribution of this section—in §4.2. Lastly, we discuss the relation between our method and forward-rendering ReSTIR as well as several other technical aspects in §4.3.

4.1 Differentiable Streaming RIS

Estimating Eq. (13) via Monte Carlo integration requires sampling material light paths $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$. Since the material path space $\hat{\Omega}$ coincides with the ordinary one $\Omega(\boldsymbol{\theta}_0)$, we can sample $\bar{\boldsymbol{p}}$ by repurposing methods developed for forward rendering.

We first construct a camera ray from $\boldsymbol{p}_2$ and compute the closest intersection $\boldsymbol{p}_1$ between this ray and the scene (specifically, the reference configuration $\mathcal{B}$). With $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ determined, we apply streaming RIS (Algorithm 1) to sample $\boldsymbol{p}_0$ on a light source by letting[3]

$$\hat{p}(\boldsymbol{p}_0; \boldsymbol{p}_1) = L_{\mathrm{e}}(\boldsymbol{p}_0 \to \boldsymbol{p}_1)\, f_{\mathrm{s}}(\boldsymbol{p}_0 \to \boldsymbol{p}_1 \to \boldsymbol{p}_2)\, G(\boldsymbol{p}_0 \leftrightarrow \boldsymbol{p}_1), \tag{14}$$

and $p_{\mathrm{e}}(\boldsymbol{p}_0)$ provided by standard light sampling (i.e., being proportional to emitted radiance $L_{\mathrm{e}}(\boldsymbol{p}_0 \to \boldsymbol{p}_1)$). Using the resulting reservoir r from M candidate light samples, we obtain a one-sample differential RIS estimator:

$$\langle(\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})_{\text{int}}\rangle_{\text{ris}}^{M} = \frac{(\partial_{\boldsymbol{I}}\mathcal{L})[q]\;\mathsf{r}.W}{p(\boldsymbol{p}_1)}\left(\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\hat{f}^{(q)}(\bar{\boldsymbol{p}})\right), \tag{15}$$

where $\bar{\boldsymbol{p}} = (\mathsf{r}.\boldsymbol{y},\ \boldsymbol{p}_1,\ \boldsymbol{p}_2)$ and $\mathsf{r}.\boldsymbol{y}$ is the reservoir's stored sample.

[2] We assume all images are gray-scale (with scalar pixel values) for notational simplicity.

[3] On the left hand side of Eq. (14), we omit the dependency of $\hat{p}$ on $\boldsymbol{p}_2$ because, for perspective pinhole cameras—which is the case we focus on, $\boldsymbol{p}_2$ is constant.

ALGORITHM 3: Combining reservoirs from spatial and temporal neighbors

```
1  SpatiotemporalReuse(r, Q)
   Input: Reservoir r; reservoirs Q = {r_1, ..., r_k} to combine
   Output: One combined reservoir
2  begin
3      for i = 1, ..., k do
4          UpdateReservoir(r, r_i.y, p̂(r.y; r.p) r.W r.M)
5      end
6      r.M ← Σ_{i=1}^{k} r_i.M;
7      Z ← 0;
8      for i = 1, ..., k do
9          if p̂(r.y; r_i.p) > 0 then
10             Z ← Z + r_i.M;
11         end
12     end
13     r.W ← r.w_sum / (p̂(r.y; r.p) Z);
14     return r;
15 end
```

In practice, we efficiently compute Eq. (15) by applying reverse-mode automatic differentiation (AD) to

$$\texttt{detach}\left(\frac{(\partial_{\boldsymbol{\mathcal{I}}}\mathcal{L})[q]\ \texttt{r}.W}{p(\boldsymbol{p}_1)}\right)\hat{f}^{(q)}(\bar{\boldsymbol{p}}), \tag{16}$$

where quantities surrounded by the detach() function are treated as constants (i.e., independent of $\boldsymbol{\theta}$) by AD.

4.2 Spatiotemporal Reuse

Building on our differential streaming RIS estimator in Eq. (15), we introduce a ReSTIR algorithm to reuse spatial and temporal candidates. Note that unlike forward rendering, where temporal reuse applies to neighboring *frames*, our technique reuses temporal candidates on consecutive *inverse-rendering gradient steps*.

As outlined in Algorithm 4, our method consists of path sampling and gradient computation stages with only the latter needing differentiable computation. We detail both stages below.

Path sampling. This stage samples a set of material light paths $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ that are later used to compute gradients. As discussed in §4.1, we sample the path by: (i) constructing a camera ray from $\boldsymbol{p}_2$ on the detector and intersecting the scene at $\boldsymbol{p}_1$ (Line 6 of Algorithm 4); and (ii) drawing $\boldsymbol{p}_0$ on a light source using streaming RIS (Line 7).

To improve RIS effectiveness, we reuse reservoirs from the previous gradient step (Line 24). To this end, we record the *shading point* $\boldsymbol{p}_1$ (where RIS occurs) in each reservoir as $\texttt{r}.\boldsymbol{p}$ (Line 8). After obtaining a reservoir r using RIS, we find neighbor reservoirs Q such that each $\texttt{r}' \in Q$ has its shading point $\texttt{r}'.\boldsymbol{p}$ near $\texttt{r}.\boldsymbol{p}$ (Line 9). We accelerate this nearest neighbor search using point Kd-trees [Wald 2022]. After finding neighbors Q, we combine them with r using Algorithm 3 (Line 10), which is equivalent to Algorithm 2 except for having the target function $\hat{p}(\cdot; \boldsymbol{p}')$ defined with respect to a shading point $\boldsymbol{p}'$ (as opposed to a pixel). The light sample $\texttt{r}.\boldsymbol{y}$ after combining reservoirs completes the path sample $\bar{\boldsymbol{p}} = (\texttt{r}.\boldsymbol{y},\ \boldsymbol{p}_1,\ \boldsymbol{p}_2)$.

ALGORITHM 4: Our streaming RIS with spatiotemporal reuse for the interior integral in Eq. (13)

```
1  EstInterior(prevReservoirs)
2  begin
       /* Sampling light paths (non-differentiable)      */
3      updateInteriorReservoirs(prevReservoirs) ;      // §4.2
4      paths ← {}; reservoirs ← {};
5      foreach pixel q do
           /* Sample camera ray through pixel q          */
6          Sample p_1 with the pdf p(p_1);
           /* Generate initial candidates                */
7          Generate reservoir r using streaming RIS ;      // §4.1
8          r.p ← p_1 ;                  // Record shading point
           /* Spatiotemporal reuse                       */
9          Q ← PickNeighbors(prevReservoirs, r) ;      // §4.2
10         r ← SpatiotemporalReuse(r, Q) ;          // Alg. 3
           /* Store sampled path                         */
11         paths[q].vtx ← (r.y, p_1, p_2);
12         paths[q].pdf ← p(p_1)/r.W;
           /* Store reservoir                            */
13         reservoirs[q] ← r;
           /* Forward rendering                          */
14         I[q] ← f̂^(q)(paths[q].vtx)/paths[q].pdf;
15     end
       /* Gradient computation (differentiable)          */
16     Compute rendering loss L using rendered image I;
17     ∂_I L ← backward(L) ;          // Compute ∂_I L := ∂L/∂I
18     (d_θ L)_int ← 0;
19     foreach pixel q do
20         p̄, pdf ← paths[q].vtx, paths[q].pdf;
21         I ← detach(∂_I L[q]/pdf) f̂^(q)(p̄) ;          // Eq. (16)
           /* Back-propagate gradients                   */
22         (d_θ L)_int ← (d_θ L)_int + backward(I);
23     end
24     return (d_θ L)_int, ∂_I L, reservoirs;
25 end
```

Updating reservoirs. We note that as previous parameters $\boldsymbol{\theta}_{\text{prev}}$ generally differ from current step values $\boldsymbol{\theta}_{\text{cur}}$, reservoirs must be updated (Line 3) before reuse. Specifically, any prior step reservoir r stores light sample $\texttt{r}.\boldsymbol{y}$ and shading point $\texttt{r}.\boldsymbol{p}$ from prior reference configuration $\mathcal{B}_{\text{prev}} = \mathcal{M}(\boldsymbol{\theta}_{\text{prev}})$. The current step must instead use reference configuration $\mathcal{B}_{\text{cur}} = \mathcal{M}(\boldsymbol{\theta}_{\text{cur}})$.

To keep $\texttt{r}.\boldsymbol{y}$ and $\texttt{r}.\boldsymbol{p}$ updated, we focus on the common inverse-rendering setting that represents scene geometries $\mathcal{M}(\boldsymbol{\theta}_{\text{prev}})$ and $\mathcal{M}(\boldsymbol{\theta}_{\text{cur}})$ with triangle meshes of identical topology (but possibly differing vertex positions). We record indices of triangles containing $\texttt{r}.\boldsymbol{y}$ and $\texttt{r}.\boldsymbol{p}$ and corresponding barycentric coordinates. Then, during the updateInteriorReservoirs() on Line 3, we recompute $\texttt{r}.\boldsymbol{y}$ and $\texttt{r}.\boldsymbol{p}$ for each reservoir from the stored triangle indices, barycentrics, and now-current vertex positions (see Figure 2). Lastly, after updating all reservoirs, we rebuild the acceleration structure used for efficient nearest-neighbor search (Line 9).

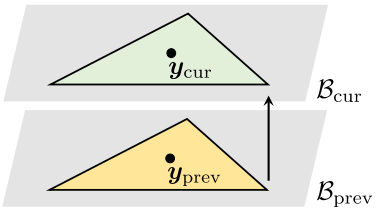


Fig. 2. Updating a reservoir's light sample $\boldsymbol{y}_{\text{cur}}$ from $\boldsymbol{y}_{\text{prev}}$ in the previous gradient step uses stored triangle ID and barycentric coordinates along with new vertex positions from $\mathcal{B}_{\text{cur}}$.

Gradient computation. After path sampling, our gradient computation evaluates path contributions using Eq. (15) with gradient $\mathrm{d}\hat{f}^{(q)}/\mathrm{d}\boldsymbol{\theta}$ obtained using reverse-mode automatic differentiation. As differentiation of $\hat{f}^{(q)}$ is *local* (i.e., occurs per path), no global computation graph is needed—similar to previous radiative backpropagation methods [Nimier-David et al. 2020; Vicini et al. 2021].

4.3 Discussion

Relation to forward-rendering ReSTIR. Although our ReSTIR-based technique expressed in §4.2 is closely related to forward-rendering ReSTIR [Bitterli et al. 2020; Lin et al. 2022], there are several technical differences. Notably, previous ReSTIR methods store reservoirs in the *screen space*. Our method, on the other hand, stores reservoirs at shading points $\mathsf{r}.\boldsymbol{p}$ over the reference surface $\mathcal{B}$. Doing so not only allows our stored reservoirs to evolve with the scene geometry naturally, but also enables efficient multi-view rendering with mini-batching—which we will discuss in §7.

Visibility awareness. The original ReSTIR by Bitterli et al. [2020] proposed an option that neglects visibility from light vertex $\boldsymbol{p}_0$ to the shading point $\boldsymbol{p}_1$ for faster forward rendering:

$$\hat{p}(\boldsymbol{p}_0) = L_{\text{e}}(\boldsymbol{p}_0 \to \boldsymbol{p}_1)\, f_{\text{s}}(\boldsymbol{p}_0 \to \boldsymbol{p}_1 \to \boldsymbol{p}_2)\, G_0(\boldsymbol{p}_0 \leftrightarrow \boldsymbol{p}_1), \tag{17}$$

with G_0 the geometric term G with visibility $\mathbb{V}(\boldsymbol{p}_0 \leftrightarrow \boldsymbol{p}_1)$ omitted.

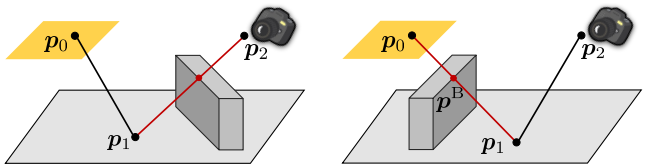
Instead, we consider the full visibility in Eq. (14)—which has been demonstrated necessary for ensuring unbiasedness [Bitterli et al. 2020; Lin et al. 2022]. Further, this allows the sampling of $\boldsymbol{p}_0$ to be more occlusion-aware, improving result quality.

Rejecting reservoirs. Occasionally, a prior reservoir r' (returned by the `PickNeighbors()` function) near the current one r can have a very different surface normal. In other words, $\boldsymbol{n}(\mathsf{r}'.\boldsymbol{p}) \cdot \boldsymbol{n}(\mathsf{r}.\boldsymbol{p}) \ll 1$ with $\boldsymbol{n}(\boldsymbol{p})$ the surface normal at $\boldsymbol{p}$. In this case, we follow Bitterli et al. and exclude any reservoirs r' with $\boldsymbol{n}(\mathsf{r}'.\boldsymbol{p}) \cdot \boldsymbol{n}(\mathsf{r}.\boldsymbol{p})$ below some user-specified threshold from the `SpatiotemporalReuse ()` call.

5 RESTIR FOR BOUNDARY INTEGRALS

We now show how to apply ReSTIR to estimate the boundary component of Eq. (12) from §3.3:

$$(\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})_{\text{bnd}} := \int_{\partial\hat{\Omega}} (\partial_{\mathcal{I}}\mathcal{L})[q]\, \Delta\hat{f}_K^{(q)}(\bar{\boldsymbol{p}}) \left(\boldsymbol{n}_{\partial}(\boldsymbol{p}_K)^{\top} \frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\boldsymbol{\theta}}\right) \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}). \tag{18}$$



(a) Primary (b) Secondary

Fig. 3. Primary and secondary boundary paths with boundary segments in red. We estimate contributions of these paths separately (§5.1 and §5.2).

Since this boundary integral is unique to differentiable rendering, our technique, to our knowledge, is the first to utilize ReSTIR for efficient estimation of these integrals.

Two types of boundary paths. Recall that a material boundary path $\bar{\boldsymbol{p}}$ is identical to an ordinary path except it has exactly one boundary segment. For direct illumination, shown in Figure 3, there are two boundary paths types depending on the location of boundary segment $\overline{\boldsymbol{p}_K\, \boldsymbol{p}_{K+1}}$. Any path with $K = 1$ has endpoint $\boldsymbol{p}_2$ on the detector and is called a **primary boundary path**; any with $K = 0$ has endpoint $\boldsymbol{p}_0$ on a light and is called a **secondary boundary path** (see Figure 3).

Previously, contributions from primary paths could be estimated relatively easily, but the secondary ones remained more challenging. In §5.1 and §5.2, we design new ReSTIR-based estimators specifically to sample both types of boundary paths.

5.1 Sampling Primary Boundary Paths

A primary boundary path has its boundary segment on the camera ray (see Figure 3-a). Similar to sampling ordinary paths (in §4.1), we generate a primary boundary path by: (i) constructing a camera ray defining $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$; and (ii) drawing $\boldsymbol{p}_0$ on a light source. In the first step, we efficiently pick the camera ray using Monte Carlo edge sampling [Li et al. 2018] with object silhouettes projected onto the image plane during preprocessing.

With $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ determined, we sample the light vertex $\boldsymbol{p}_0$ using streaming RIS (Algorithm 1) with the same $\hat{p}$ and p_{c} functions as the interior integral (§4). This gives the following RIS-based estimator:

$$\left\langle (\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})^{\text{pri}}_{\text{bnd}} \right\rangle^{M}_{\text{ris}} = \frac{(\partial_{\mathcal{I}}\mathcal{L})[q]\, \Delta\hat{f}_1^{(q)}(\bar{\boldsymbol{p}})\, \mathsf{r}.W}{p(\boldsymbol{p}_1)} \left(\boldsymbol{n}_{\partial}(\boldsymbol{p}_1)^{\top} \frac{\mathrm{d}\boldsymbol{p}_1}{\mathrm{d}\boldsymbol{\theta}}\right), \tag{19}$$

where $\bar{\boldsymbol{p}} = (\mathsf{r}.\boldsymbol{y},\ \boldsymbol{p}_1,\ \boldsymbol{p}_2)$.

In practice, we efficiently compute Eq. (19) by applying reverse-mode automatic differentiation to

$$\texttt{detach}\left(\frac{(\partial_{\mathcal{I}}\mathcal{L})[q]\, \Delta\hat{f}_1^{(q)}(\bar{\boldsymbol{p}})\, \mathsf{r}.W}{p(\boldsymbol{p}_1)}\, \boldsymbol{n}_{\partial}(\boldsymbol{p}_1)\right) \cdot \boldsymbol{p}_1, \tag{20}$$

where "$\cdot$" denotes a dot product, and $\boldsymbol{p}_1$ outside the `detach()` function is determined by differentiable ray intersection between the camera ray and the scene.

Spatiotemporal reuse. The similarity in sampling $\boldsymbol{p}_0$ for ordinary and primary boundary paths allows us to improve the effectiveness of our boundary estimator in Eq. (19) by reusing reservoirs generated to estimate the interior component (discussed in §4).

ALGORITHM 5: Our streaming RIS with spatiotemporal reuse for primary boundary paths

```
1  EstPrimaryBound(∂_I L, prevReservoirs)
2  begin
3      (d_θ L)^pri_bnd ← 0;
       /* Sampling boundary paths (non-differentiable)   */
4      paths ← {};
5      for i = 1 to N_pri do
6          Sample p_1 with pdf p(p_1) ;          // MC edge sampling
7          Let q be the index of the pixel intersecting p_1 p_2;
           /* Generate initial candidates                */
8          Generate reservoir r using stream RIS ;        // §4.1
           /* Spatiotemporal reuse                       */
9          Q ← PickNeighbors(prevReservoirs, r) ;         // §4.2
10         r ← SpatiotemporalReuse(r, Q) ;              // Alg. 3
           /* Store sampled path                         */
11         paths[i].pid ← q ;                 // Record pixel index
12         paths[i].vtx ← (r.y, p_1, p_2);
           /* Record the detached factor of Eq. (20)     */
13         pdf ← p(p_1) N_pri/r.W;
14         paths[i].vec ← ∂_I L[q] Δf̂^(q)(p̄) n_∂(p_1)/pdf
15     end
       /* Gradient computation (differentiable)          */
16     for i = 1 to N_pri do
17         q, p̄, v ← paths[i].pid, paths[i].vtx, paths[i].vec;
18         Re-compute p_1 using differentiable ray intersection;
19         I ← detach(v) · p_1 ;                          // Eq. (20)
           /* Back-propagate gradients                   */
20         (d_θ L)^pri_bnd ← (d_θ L)^pri_bnd + backward(I);
21     end
22     return (d_θ L)^pri_bnd;
23 end
```

Specifically, Algorithm 5 outlines how we reuse these reservoirs from the prior gradient step (and updated via `updateReservoirs()` in Algorithm 4). Normally it requires only a few samples to estimate the primary boundary, so we opt not to store new reservoirs (obtained after Line 10) in Algorithm 5 to simplify the overall pipeline.

5.2 Sampling Secondary Boundary Paths

Unlike primary boundary paths, secondary paths (as illustrated in Figure 3b) are more challenging as they need knowledge of object silhouettes with respect to arbitrary shading locations $\boldsymbol{p}_1$.

Multi-directional sampling. Previously, Li et al. [2018] leveraged a high-dimensional bounding volume hierarchy to accelerate silhouette detection, but this remains prohibitively expensive for complex geometry. Zhang et al. [2020] introduced a *multi-directional* sampling technique to avoid explicit detection of object silhouettes. Specifically, they build a secondary boundary path in three steps:

S.1 Sample point $\boldsymbol{p}^{\mathrm{B}}$ from a triangle edge with probability $p(\boldsymbol{p}^{\mathrm{B}})$; sample light vertex $\boldsymbol{p}_0$ with conditional probability $p(\boldsymbol{p}_0 \mid \boldsymbol{p}^{\mathrm{B}})$.

S.2 If $\boldsymbol{p}_0$ and $\boldsymbol{p}^{\mathrm{B}}$ are mutually visible, trace a ray from $\boldsymbol{p}^{\mathrm{B}}$ in direction $\overrightarrow{\boldsymbol{p}_0\,\boldsymbol{p}^{\mathrm{B}}}$ to find intersection $\boldsymbol{p}_1$. This guarantees $\overline{\boldsymbol{p}_0\,\boldsymbol{p}_1}$ is a boundary segment.

S.3 Complete path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \boldsymbol{p}_2)$ with $\boldsymbol{p}_2$ being the perspective camera's center of projection.

This sampling process gives an ordinary Monte Carlo estimator:

$$\left\langle (\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})^{\mathrm{snd}}_{\mathrm{bnd}} \right\rangle = \frac{F^{\mathrm{snd}}_{\mathrm{bnd}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0, \boldsymbol{p}_2)}{p(\boldsymbol{p}^{\mathrm{B}})\, p(\boldsymbol{p}_0 \mid \boldsymbol{p}^{\mathrm{B}})} \left(\boldsymbol{n}_{\partial}(\boldsymbol{p}_0)^{\top} \frac{\mathrm{d}\boldsymbol{p}_0}{\mathrm{d}\boldsymbol{\theta}} \right), \tag{21}$$

where

$$F^{\mathrm{snd}}_{\mathrm{bnd}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0, \boldsymbol{p}_2) \coloneqq (\partial_{\mathcal{I}}\mathcal{L})[q]\, \Delta\hat{f}^{(q)}_0(\bar{\boldsymbol{p}})\, J^{\mathrm{B}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0), \tag{22}$$

with

$$J^{\mathrm{B}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0) \coloneqq \left\| \frac{\mathrm{d}A(\boldsymbol{p}_1)\, \mathrm{d}\ell(\boldsymbol{p}_0)}{\mathrm{d}\ell(\boldsymbol{p}^{\mathrm{B}})\, \mathrm{d}A(\boldsymbol{p}_0)} \right\|, \tag{23}$$

the Jacobian determinant for changing from joint probability density $p(\boldsymbol{p}_0, \boldsymbol{p}_1)$ to $p(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0)$. Refer to Zhang et al. [2020] for more details, including the derivation of J^{B}.

Streaming RIS. The first step (**S.1**) in multi-directional sampling draws light vertex $\boldsymbol{p}_0$ given $\boldsymbol{p}^{\mathrm{B}}$ on a triangle edge. For scenes with complex illumination and visibility, sampling $\boldsymbol{p}_0$ via streaming RIS (Algorithm 1) improves efficiency. To this end, given $\boldsymbol{p}^{\mathrm{B}}$ we set $\hat{p}$ as

$$\hat{p}(\boldsymbol{p}_0;\ \boldsymbol{p}^{\mathrm{B}}) = F^{\mathrm{snd}}_{\mathrm{bnd}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0, \boldsymbol{p}_2), \tag{24}$$

with $F^{\mathrm{snd}}_{\mathrm{bnd}}$ defined in Eq. (22), and standard light sampling with pdf p_{c}. Here we neglect gradient term $\left(\boldsymbol{n}_{\partial}(\boldsymbol{p}_0)^{\top} \frac{\mathrm{d}\boldsymbol{p}_0}{\mathrm{d}\boldsymbol{\theta}}\right)$ as it is difficult to obtain in the path sampling stage. This leads to our RIS estimator:

$$\left\langle (\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L})^{\mathrm{snd}}_{\mathrm{bnd}} \right\rangle^{M}_{\mathrm{ris}} = \frac{F^{\mathrm{snd}}_{\mathrm{bnd}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0, \boldsymbol{p}_2)\, \mathsf{r}.W}{p(\boldsymbol{p}^{\mathrm{B}})} \left(\boldsymbol{n}_{\partial}(\boldsymbol{p}_0)^{\top} \frac{\mathrm{d}\boldsymbol{p}_0}{\mathrm{d}\boldsymbol{\theta}} \right), \tag{25}$$

which essentially replaces $p(\boldsymbol{p}_0 \mid \boldsymbol{p}^{\mathrm{B}})$ in the ordinary Monte Carlo estimator of Eq. (21) with $1/\mathsf{r}.W$ as $\boldsymbol{p}_0$ is now drawn with RIS.

With automatic differentiation, Eq. (25) can be computed as:

$$\mathtt{detach}\left(\frac{F^{\mathrm{snd}}_{\mathrm{bnd}}(\boldsymbol{p}^{\mathrm{B}}, \boldsymbol{p}_0, \boldsymbol{p}_2)\, \mathsf{r}.W}{p(\boldsymbol{p}^{\mathrm{B}})}\, \boldsymbol{n}_{\partial}(\boldsymbol{p}_0) \right) \cdot \boldsymbol{p}_0. \tag{26}$$

Spatiotemporal reuse. To improve our estimator's efficiency, we introduce a new spatiotemporal reuse scheme for secondary boundary paths. As described in Algorithm 6, we record $\boldsymbol{p}^{\mathrm{B}}$ in each reservoir r (Line 9) and use this point for nearest neighbor searches (Line 10), rather than using shading point $\boldsymbol{p}_1$ as in our other computations. After merging r with nearby reservoirs from the previous gradient step, we trace a ray to obtain $\boldsymbol{p}_1$ (**S.2**) to complete the path sample $\bar{\boldsymbol{p}} = (\mathsf{r}.\boldsymbol{y},\ \boldsymbol{p}_1,\ \boldsymbol{p}_2)$.

As in Algorithms 4 and 5, the sampled paths are then evaluated using automatic differentiation (Line 24 of Algorithm 6) via Eq. (26).

Importance sampling of $\boldsymbol{p}^{\mathrm{B}}$. Before drawing $\boldsymbol{p}_0$ on a light, Zhang et al.'s [2020] multi-directional sampling picks $\boldsymbol{p}^{\mathrm{B}}$ from a triangle edge (Line 7). A naïve implementation could (i) select an edge $\boldsymbol{e}_k$ with probability p_k proportional to edge length $\|\boldsymbol{e}_k\|$; and (ii) draw $\boldsymbol{p}^{\mathrm{B}}$ uniformly from $\boldsymbol{e}_k$. But this loses efficiency in complex scenes if it frequently returns samples $\boldsymbol{p}^{\mathrm{B}}$ not visible to any light.

ALGORITHM 6: Our streaming RIS with spatiotemporal reuse for secondary boundary paths

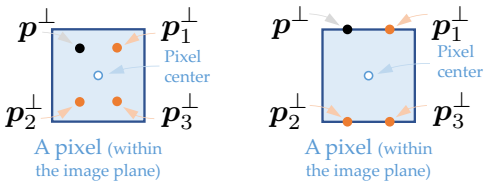
```
1  EstSecondaryBound(∂_I L, prevEdgeReservoirs)
2  begin
3      (d_θ L)^snd_bnd ← 0;
       /* Sampling boundary paths (non-differentiable)   */
4      updateEdgeReservoirs(prevEdgeReservoirs);
5      paths ← {}; edgeReservoirs ← {};
6      for i = 1 to N_snd do
7          Sample p^B from an edge with pdf p(p^B);
           /* Generate initial candidates                */
8          Generate reservoir r using stream RIS ;        // §5.2
9          r.p ← p^B ;                                    // Record edge point
           /* Spatiotemporal reuse                       */
10         Q ← PickNeighbors(prevEdgeReservoirs, r) ;     // §5.2
11         r ← SpatiotemporalReuse(r, Q) ;                // Alg. 3
           /* Complete the path sample                   */
12         p_0 ← r.y; p_1 ← rayIntersect(p^B, p_0 p^B);
13         r.p_1 ← p_1 ;                                  // Used by Alg. 8
           /* Store sampled path                         */
14         Let q be the index of pixel intersecting p_1 p_2;
15         paths[i].pid ← q ;                             // Record pixel index
16         paths[i].vtx ← (p_0, p_1, p_2);
           /* Record the detached factor of Eq. (26)     */
17         pdf ← p(p_B) N_snd/r.W;
18         paths[i].vec ← F^snd_bnd(p^B, r.y, p_2) n_∂(p_0)/pdf;
           /* Store reservoir                            */
19         edgeReservoirs[i] ← r;
20     end
       /* Gradient computation (differentiable)          */
21     for i = 1 to N_snd do
22         q, p̄, v ← paths[i].pid, paths[i].vtx, paths[i].vec;
23         Re-compute p_0 using differentiable ray intersection;
24         I ← detach(v) · p_0 ;                          // Eq. (26)
           /* Back-propagate gradients                   */
25         (d_θ L)^snd_bnd ← (d_θ L)^snd_bnd + backward(I);
26     end
27     return (d_θ L)^snd_bnd, edgeReservoirs;
28 end
```

To address this problem, we leverage our reservoirs to *guide* edge sampling as follows. For an edge $\boldsymbol{e}_k$ containing n_k reservoirs (i.e., n_k reservoirs r with $\mathsf{r}.\boldsymbol{p}$ lying on $\boldsymbol{e}_k$), we set its (discrete) sampling probability p_k to be proportional to the average weight of the n_k reservoirs modulated by the edge length $\|\boldsymbol{e}_k\|$:

$$p_k \propto \left(\frac{\sum_{\mathsf{r} \in \boldsymbol{e}_k} (\mathsf{r}.\mathrm{w}_{\mathrm{sum}} / \mathsf{r}.M)}{n_k} + \epsilon \right) \|\boldsymbol{e}_k\|. \tag{27}$$

Importance sampling for edges using Eq. (27) significantly improves performance, as we demonstrate in §8.1.

Various guiding methods in primary-sample space [Zhang et al. 2020; Yan et al. 2022] have been proposed. They importance sample both $\boldsymbol{p}^{\mathrm{B}}$ and $\boldsymbol{p}_0$ but require nontrivial precomputation. Instead, our edge importance sampling is a by-product of ReSTIR sample reuse and is orthogonal to these methods.



(a) **Interior** sampling (b) **Boundary** sampling

Fig. 4. **Pixel-level antithetic sampling:** To differentiate pixel reconstruction filters using the PSDR formulation [Zhang et al. 2020], Yu et al. [2022] showed pixel-level *antithetic sampling* is crucial to efficiently estimate derivatives relative to object shape. (a) When computing interior integrals (§4), for each camera ray intersection at $\boldsymbol{p}^\perp$, Yu et al. [2022] proposed generating three correlated rays through $\boldsymbol{p}_1^\perp$, $\boldsymbol{p}_2^\perp$, and $\boldsymbol{p}_3^\perp$ symmetrically around the pixel center. (b) When handling pixel boundaries (§6), which emerges from pixel filter discontinuities and can be handled identically to primary boundaries (§5.1), these correlated rays are sampled on the pixel boundary.

6 RESTIR AND ANTITHETIC SAMPLING

When differentiating pixel reconstruction filters with respect to object geometries using PSDR [Zhang et al. 2020], Yu et al. [2022] demonstrated that pixel-level antithetic sampling is crucial for ensuring fast convergence.

As illustrated in Figure 4, antithetic sampling shoots two or four camera rays whose image plane intersections exhibit point symmetry with respect to the pixel center. Additionally, these camera rays must be *correlated* so the derivatives of their contributions cancel.

Previously, multi-path correlation was normally handled by forcing paths to use identical pseudo-random sequences by reusing the random seeds when sampling each path.

Unfortunately, this is insufficient when using ReSTIR. As shown in Figure 5-a, even with identical random numbers, the nearest neighbor search at two antithetic shading points $\boldsymbol{p}_1^{(0)}$ and $\boldsymbol{p}_1^{(1)}$ may return different reservoirs $Q^{(0)}$ and $Q^{(1)}$ (Line 9, Algorithm 4). This causes the merged reservoirs returned by `SpatiotemporalReuse` () (Line 10) to become less correlated.

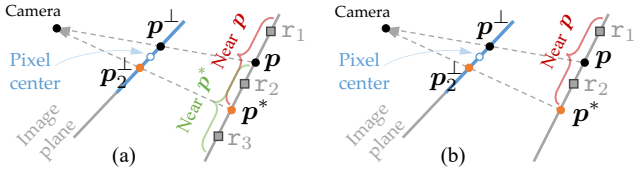


Fig. 5. **Correlating reservoirs:** (a) Independent spatiotemporal reuse at antithetic shading points $\boldsymbol{p}$ and $\boldsymbol{p}^*$ may reuse different reservoirs. Here, $Q = \{\mathsf{r}_1, \mathsf{r}_2\}$ will be used at $\boldsymbol{p}$ and $Q^* = \{\mathsf{r}_2, \mathsf{r}_3\}$ at $\boldsymbol{p}^*$. This weakens antithetic path correlation, reducing antithetic sampling's effectiveness. (b) We randomly select one antithetic shading point and force others to use the same reservoir set. Here $\boldsymbol{p}$ is chosen, so both $\boldsymbol{p}$ and $\boldsymbol{p}^*$ reuse $Q = \{\mathsf{r}_1, \mathsf{r}_2\}$. This better correlates antithetic paths, improving sampling efficiency.

ALGORITHM 7: ReSTIR-based inverse rendering

```
1  EstimateGradient(I_0, resvr_in, edgeResvr_in)
   Input: Target image I_0; reservoirs resvr_in and edgeResvr_in for
          spatiotemporal reuse
   Output: Gradient d_θ L; reservoirs resvr_out and edgeResvr_out for
           later reuse
2  begin
       /* Estimate the interior (Alg. 4)                          */
3      (d_θ L)_int, ∂_I L, resvr_out ← EstInterior(resvr_in);
       /* Estimate the primary boundary (Alg. 5)                  */
4      (d_θ L)^pri_bnd ← EstPrimaryBound(∂_I L, resvr_in);
       /* Estimate the secondary boundary (Alg. 6)                */
5      (d_θ L)^snd_bnd, edgeResvr_out ←
         EstSecondaryBound(∂_I L, edgeResvr_in);
       /* Compute the full gradient                               */
6      d_θ L ← (d_θ L)_int + (d_θ L)^pri_bnd + (d_θ L)^snd_bnd;
7      return d_θ L, resvr_out, edgeResvr_out;
8  end

9  InverseRendering(I_0)
   Input: Target image I_0
   Output: Optimized scene parameters θ
10 begin
11     Initialize θ_0;
12     resvr^(0) ← {}; edgeResvr^(0) ← {};
       /* Gradient-based optimization                             */
13     for t = 1, 2, ..., N_epoch do
           /* Estimate the gradient d_θ L := dL/dθ                */
14         d_θ L, resvr^(t), edgeResvr^(t) ←
             EstimateGradient(I_0, resvr^(t−1), edgeResvr^(t−1));
           /* Update the parameters θ                             */
15         UpdatedParameters(θ, d_θ L);
16     end
17     return θ;
18 end
```

As we show in §8, weakened correlation among antithetic paths increases variance. To address this, we force all antithetic paths to reuse one reservoir set Q. Precisely, let $\mathsf{r}^{(t)}$ be a reservoir generated using streaming RIS (Line 9) for the t-th correlated path, and

$$Q^{(t)} = \texttt{PickNeighbors}\left(\text{prevReservoirs},\ \mathsf{r}^{(t)}\right), \tag{28}$$

be the corresponding nearest neighbor reservoirs. We randomly select one set of reservoirs $Q^* \in \{Q^{(t)} : t = 0, 1, \ldots\}$ for spatiotemporal reuse (Line 10) across all correlated paths:

$$\mathsf{r}^{(t)} \leftarrow \texttt{SpatiotemporalReuse}\left(\mathsf{r}^{(t)},\ Q^*\right), \tag{29}$$

for all t. We illustrate this process in Figure 5-b.

This works for both interior and *pixel-boundary paths*—which emerge when the pixel reconstruction filter is discontinuous (e.g., box) and behave identically to primary boundaries.

7 RESTIR-BASED INVERSE RENDERING

Leveraging spatiotemporal reuse (Algorithms 4, 5, and 6) from §4 and §5, we introduce a novel gradient-based inverse rendering pipeline. As outlined in Algorithm 7, we maintain two sets of reservoirs "resvr" and "edgeResvr" through gradient steps $t = 1, 2, \ldots$. We use "resvr" to estimate interior and primary boundary components (Lines 3–4), and "edgeResvr" for the secondary boundary component (Line 5). After each gradient step, we compute the full gradient $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$ (Line 6) and update scene parameters (Line 15) via simple gradient descent (i.e., $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \lambda_t\, \mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$ for step size $\lambda_t > 0$) or more advanced methods like Adam [Kingma and Ba 2014].

Burn-in stage. To reduce variance even when starting an inverse-rendering optimization we add an optional "burn-in" stage. During this stage, we perform ReSTIR based sampling without backpropagating gradients to obtain temporal information for our method. This makes selecting learning rates easier. In this stage, we repeatedly render the *initial* configuration (without calculating $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$ or applying gradient descent), allowing ReSTIR to establish reservoirs throughout the scene. We note that, as differentiable evaluation (which takes most of the time) is not needed during burn-in, iterations are typically much faster than those during optimization.

In practice, we perform about 32 burn-in iterations for our inverse-rendering experiments.

Supporting multi-view configurations. Thus far, our derivations assume only one target image $\boldsymbol{\mathcal{I}}_0$ and one rendered image $\boldsymbol{\mathcal{I}}$ for inverse-rendering optimizations. In practice, such *single-image* configurations are insufficient to solve many real-world inverse rendering problems. Instead, *multi-view* configurations observing the same scene under multiple viewing conditions are common.

Since these configurations keep scene (and lighting) fixed across multiple views, light samples stored in reservoirs (i.e., $\mathsf{r}.\boldsymbol{y}$ for each reservoir r) remain valid. This allows us to use one set of "resvr" and "edgeResvr" buffers to store reservoirs from all views, largely reusing our single-image pipeline for multi-view optimizations.

But when mini-batching over camera views (i.e., using random subsets of views for each gradient step), reservoirs from the prior step may offer little help if not visible in the current views.

To address this problem we observe that, at each iteration, regions of the scene invisible to all randomly selected camera views will not be updated by gradient-descent because there is generally no gradient. Based on this observation, after each iteration, we only replace previous reservoirs *visible to at least one randomly selected view* with newly generated reservoirs; we preserve all occluded reservoirs for the next iteration.

We detail this process in Algorithm 8. For each camera view k, our method applies the same process (Lines 10–13) to estimate the gradient as the single-view variant expressed in Algorithm 7. Then, we preserve all previous reservoirs contained in $\text{resvr}^{(t-1)}$ and $\text{edgeResvr}^{(t-1)}$ whose shading points are occluded to all camera views in the current mini-batch by "forwarding" them to $\text{resvr}^{(t)}$ and $\text{edgeResvr}^{(t)}$ (Lines 15–24) for future reuse.

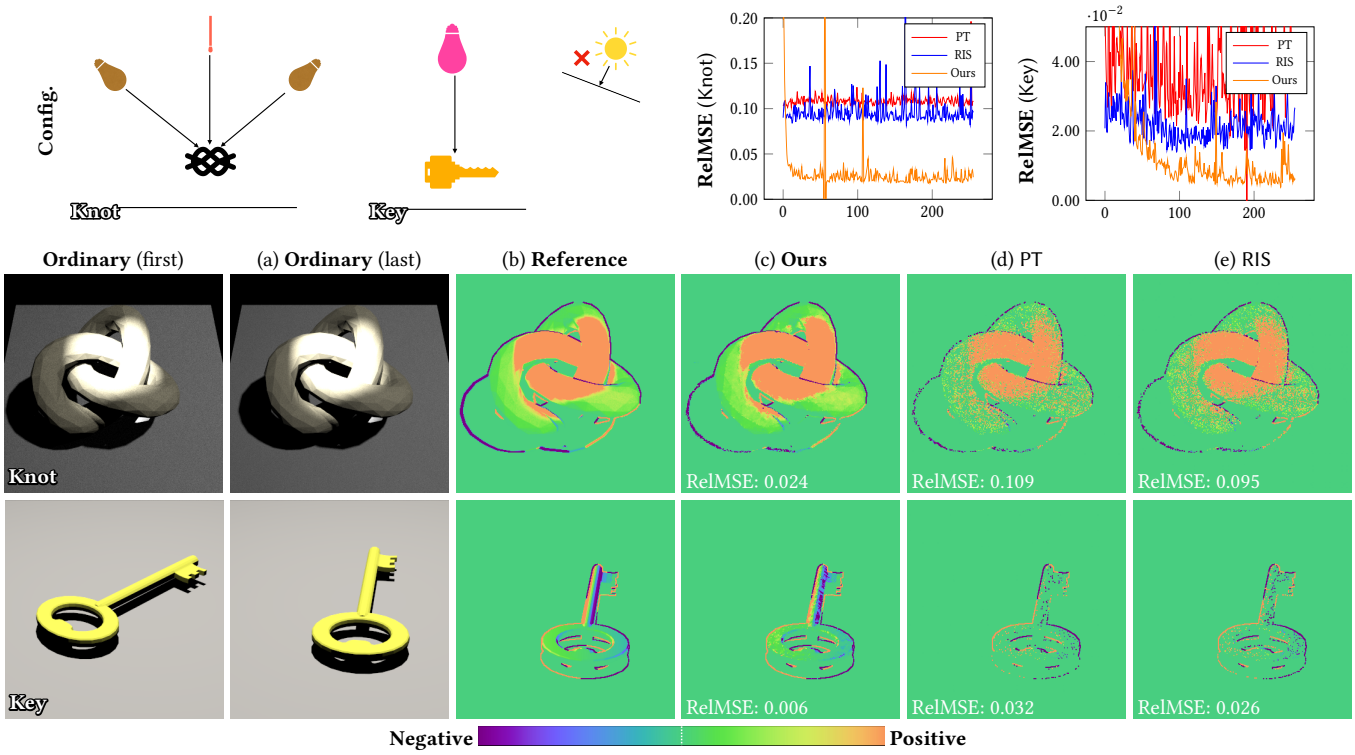


Fig. 6. **Differentiable-rendering comparisons** with PT (**B.1**) and RIS (**B.2**). Both examples use *dynamic scenes* (including 250 frames) where the Knot translates horizontally (top) and the Key rotates around the vertical axis (bottom). The derivatives are computed with respect to the translation for Knot and rotation angle for Key. The derivative renderings shown are for the last frame. At equal per-frame time, our method (c) produces significantly cleaner derivative estimates than the baselines (d, e).

Additionally, when solving for object shapes using multi-view images, combining reservoirs using Algorithm 2 would require storing multiple additional copies (e.g., one per view) of scene geometries (and corresponding acceleration structures)—which can be expensive. In practice, for better performance, we introduce a small amount of bias by checking visibilities (required by Line 9 of Algorithm 2) using only scene geometry from the current iteration. To minimize the amount of bias introduced, we use small learning rates for these optimizations.

Computational overhead. Overall, when applied for inverse rendering, our technique introduces the following computational overhead (compared with streaming RIS): (i) the burn-in stage mentioned above; (ii) nearest-neighbor searches (e.g., Line 9 of Algorithm 4); and (iii) combining reservoirs (e.g., Line 10 of Algorithm 4).

In our experiments, using equal sample, the overheads combined take no more than 20% of the total optimization time. When comparing inverse-rendering results, we allow baseline methods to use higher sample counts so that all methods use approximately identical total optimization times.

8 RESULTS

Experiment setup. We employ ReSTIR [Bitterli et al. 2020] to the PSDR [Zhang et al. 2020] formulation to improve the Monte-Carlo sampling efficiency of PSDR in differentiable and inverse rendering. To show our technique's practical use, we compare differentiable- and inverse-rendering performance against two baseline methods:

B.1 PT: The first baseline is a standard Monte Carlo method generating one light sample per camera ray. This is widely adopted by physics-based differentiable renderers including Mitsuba 3 [Jakob et al. 2022] and PSDR [Zhang et al. 2020].

B.2 RIS: Our second baseline uses streaming RIS (Algorithm 1). This is the most relevant baseline as it outperforms PT (**B.1**) for complex lighting and differs from our technique only by performing no spatiotemporal reuse.

We implement our method and the two baselines using the PSDR formulation on a GPU-based wavefront differentiable renderer. All results are generated on a workstation with an AMD Rayzen 9 5900X 12-core CPU, 32GB of RAM, and an NVIDIA RTX 3090 GPU.

All our inverse rendering results use unbiased gradient estimates expect those using multi-view input images to optimize object geometries (as discussed at the end of §7).

8.1 Evaluation and Ablation

Differentiable-rendering comparisons. We validate our technique and compare it to the two baselines (**B.1** and **B.2**) via differentiable rendering in Figure 6.

ALGORITHM 8: ReSTIR-based inverse rendering with multi-view target images

```
InverseRendering(I_0, I_1, ...)
  Input: Multi-view target images I_0, I_1, ...
  Output: Optimized scene parameters θ
begin
    Initialize θ_0;
    resvr^(0) ← {}; edgeResvr^(0) ← {};
    for t = 1, 2, ..., N_epoch do
        Select a mini-batch of camera views K ⊆ {0, 1, ...};
        /* Initialization                                   */
        d_θ L ← 0;
        resvr^(t) ← {}; edgeResvr^(t) ← {};
        foreach camera view k ∈ K do
            /* Estimate the gradient                        */
            (d_θ L)_k, resvr', edgeResvr' ←
              EstimateGradient(I_k, resvr^(t-1), edgeResvr^(t-1));
            /* Accumulate the gradient                      */
            d_θ L += (d_θ L)_k;
            /* Store new reservoirs                         */
            resvr^(t) ← resvr^(t) ∪ resvr';
            edgeResvr^(t) ← edgeResvr^(t) ∪ edgeResvr';
        end
        /* Preserve reservoirs not visible to any views
           from current the mini-batch K                    */
        foreach r ∈ resvr^(t-1) do
            if r.p is not visible to any camera from K then
                resvr^(t) ← resvr^(t) ∪ {r};
            end
        end
        foreach r ∈ edgeResvr^(t-1) do
            if r.p_1 is not visible to any camera from K then
                edgeResvr^(t) ← edgeResvr^(t) ∪ {r};
            end
        end
        /* Update the parameters θ                          */
        UpdatedParameters(θ, d_θ L);
    end
    return θ;
end
```

The first example in this figure uses the "Knot" with translating geometry, and we compute derivatives with respect to the knot translation. As illustrated on the top, lighting includes one bright spotlight and two dim fill lights. The second example shows a rotating "Key" lit by one small and one occluded large area light. We compute derivatives with respect to the key rotation angle.

For both scenes, the baselines often draw samples from bright lights that make little contribution, leading to high variance. Our method reuses valuable samples from prior frames, quickly adapting to this setting and providing significantly cleaner gradient estimates that closely match the reference.

Per-component ablation. We recall that, under the PSDR formulation [Zhang et al. 2020], a full derivative generally equals the sum of an interior (§4), a primary boundary (§5.1),[4] and a secondary boundary (§5.2) components.

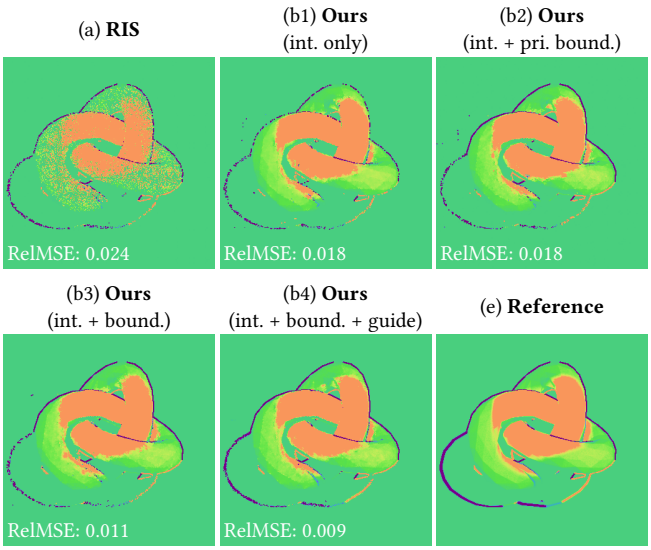


Fig. 7. **Our technique for individual components:** Using the same "Knot" scene from Figure 6, we demonstrate the effectiveness of our method on individual components of the derivative by applying it to: (b1) the interior component only; (b2) the interior and the primary boundary components; (b3) all components; and (b4) all components with importance sampling edge points. The derivative images use the same color map as Figure 6.

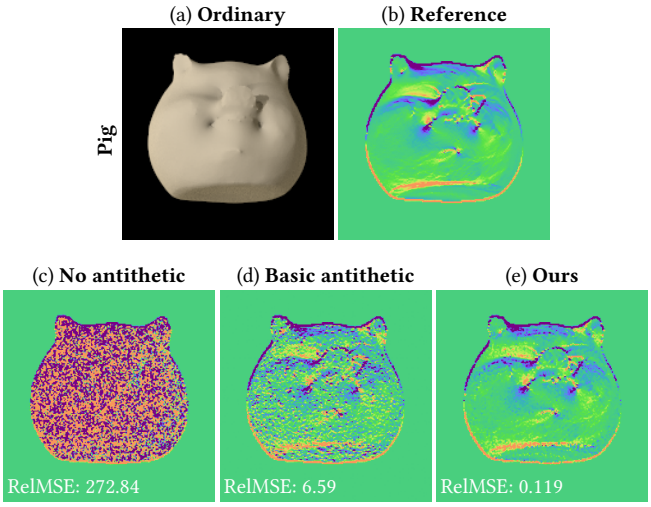


Fig. 8. **Antithetic sampling:** When differentiating with respect to object geometries using PSDR [Zhang et al. 2020], it is crucial to apply antithetic sampling at the pixel level [Yu et al. 2022], as shown in (c) and (d). Further, when ReSTIR is used, our method—which forces all antithetic shading points to reuse the same set of reservoirs—offers further variance reduction, as shown in (e). The derivative images use the same color map as Figure 6.

[4] We treat the pixel-boundary component discussed in §6 as part of the primary boundary as the two behave mostly identically.

Table 1. Inverse-rendering configurations and performance statistics. In this table, "**Trg.**" indicates the number of target images; "**Bat.**" is the size of mini-batches (i.e., the number of images rendered per iteration); "**Param.**" denotes the number of scene parameters being optimized; "**Iter.**" is the number of iterations; and "**DR**" indicates the time spent per iteration on differentiable rendering using our method.

Scene		Trg.	Bat.	Param.	Iter.	DR
Oxalis	(Fig. 1)	1	1	183K	2304	0.45s
Tree	(Fig. 10)	1	1	1	1024	0.9s
Painting	(Fig. 11)	1	1	153K	2048	0.37s
Kitty	(Fig. 12)	32	16	65K	128	14.1s
Octagon	(Fig. 13)	17	17	30K	512	12.2s
World map	(Fig. 14)	6	2	52K	1024	5.2s

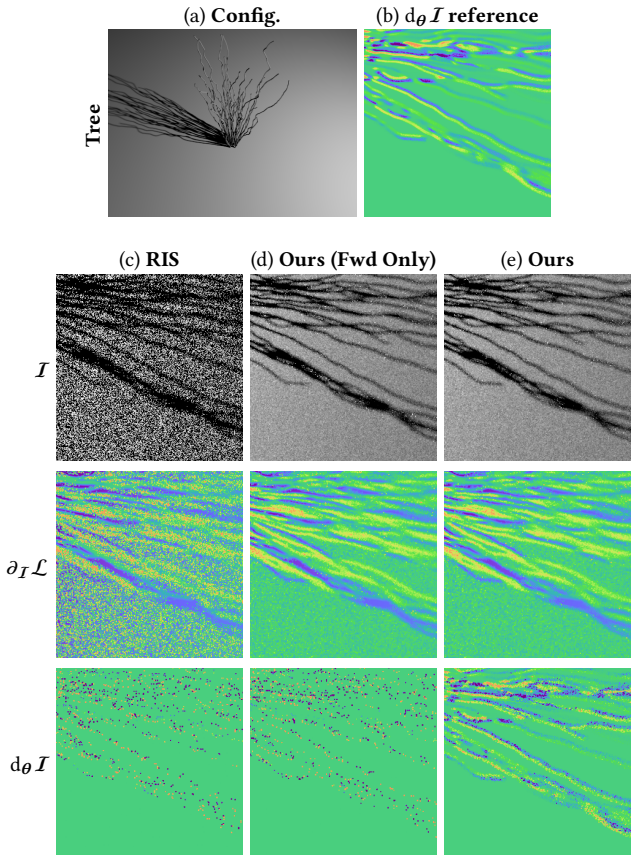


Fig. 9. **Forward-only vs. full ReSTIR:** This example contains a "Tree" lit by one small and one occluded large area lights. Using a predefined loss $\mathcal{L}(I) := \|I - I_0\|_2^2$ with some fixed I_0, we compare estimates of gradients $\partial_I \mathcal{L}$ and $d_\theta I$—whose product gives the gradient $d_\theta \mathcal{L}$ of the loss $\mathcal{L}$. Without applying ReSTIR (c), both estimates are noisy. When applying ReSTIR only for forward rendering (d), one gets better estimates of I, improving the gradient $\partial_I \mathcal{L}$. Applying ReSTIR fully (e) produces low-variance estimates for both gradients.

To evaluate our technique's effectiveness across these components, we conduct an ablation in Figure 7 using the same Knot

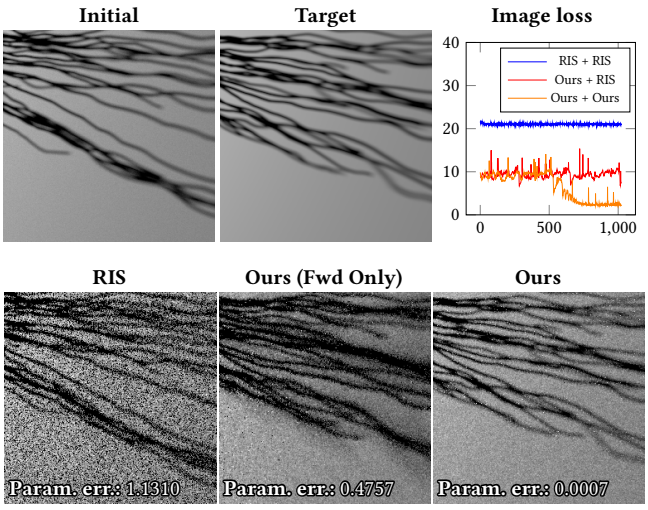


Fig. 10. **Forward-only vs. full ReSTIR:** Using the same "Tree" scene as Figure 9, we compare inverse-rendering results where we optimize the rotation angle of the tree-like object by only looking at its cast shadow. Our full method ("Ours") allows the optimization to smoothly converge to the groundtruth while the other methods fail due to high-variance estimates of the gradient $d_\theta \mathcal{L}$.

as Figure 6. Without spatiotemporal reuse, RIS (**B.2**) suffers from high variance (see Figure 7-a). Using our method only on the interior component, derivative estimates on the knot surface become significantly better (see Figure 7-b1). By also applying our technique to primary-boundaries, derivatives along the silhouette of the knots improve (see Figure 7-b2). Also using our method for secondary-boundaries reduces variance around shadow boundaries (see Figure 7-b3). Lastly, by importance sampling edge vertices, further variance reduction can be achieved (see Figure 7-b4).

Antithetic sampling. As discussed in §6, standard antithetic sampling of reconstruction filters, which is crucial to estimate derivatives relative to object geometry, can be insufficient when using ReSTIR.

We demonstrate this in Figure 8 for a "Pig" scene using a standard box filter and containing a diffuse pig lit by an area light. We estimate derivatives with respect to the vertical displacement of the pig. At *equal time*, without pixel-level antithetic sampling, the derivative estimates are extremely noisy (Figure 8-c). The antithetic sampling proposed by Yu et al. [2022] greatly reduces variance but still suffers from high variance where gradients are smooth (Figure 8-d). By forcing antithetic shading points to reuse identical sets of reservoirs, our method offers estimates with significantly improved quality (Figure 8-e).

Forward-only vs. full ReSTIR. According to the chain rule in Eq. (11), the gradient $d_\theta \mathcal{L}$ of rendering loss $\mathcal{L}$ is the product of $\partial_I \mathcal{L}$ (gradient of $\mathcal{L}$ with respect to forward render I) and $d_\theta I$ (derivative of forward render I with respect to scene parameters θ).

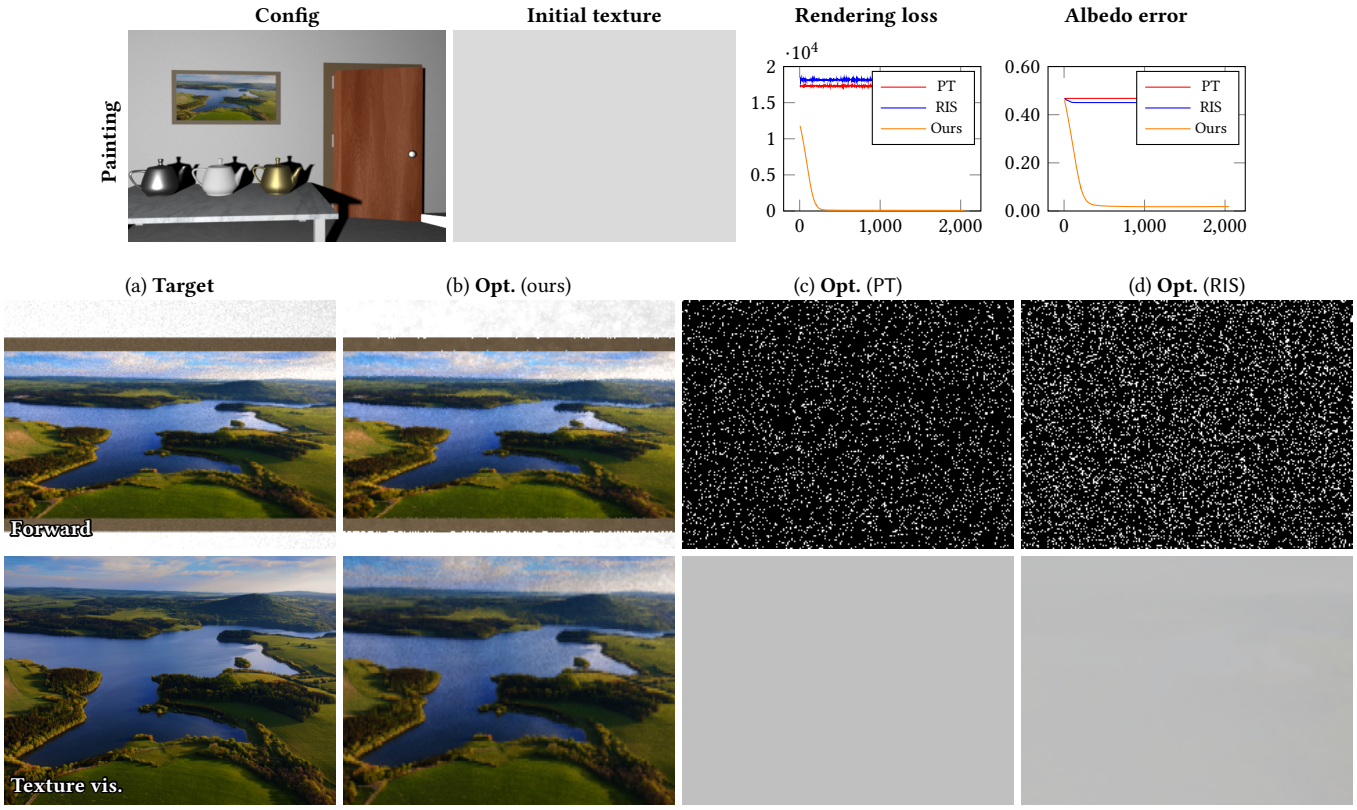


Fig. 11. **Inverse-rendering comparison** using the "Painting" setup modeled after the Veach Ajar scene with an added fill light in the room. We optimize the painting's spatially varying albedo (initialized using a gray texture). (c, d) The baseline methods (**B.1** and **B.2**) suffer from extremely high variance, thus failing to reconstruct the albedo map. (b) Our method, on the other hand, offers the robustness to reconstruct the target albedo map (a) nicely. The albedo error on the top-right is used for evaluation only.

For inverse rendering, one can apply forward-rendering ReSTIR only to improve $\boldsymbol{I}$ (and its gradient $\partial_{\boldsymbol{I}}\mathcal{L}$) while leaving $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$ handled with methods like RIS (**B.2**). We compare with this baseline in Figures 9 and 10 as follows.

Figure 9 shows a "Tree" lit by one small area light and one occluded large light. We compare forward- and differentiable-rendering (relative to the tree's rotation around its vertical axis) of the cast shadow using three configurations:

- **RIS** uses RIS for estimating both $\boldsymbol{I}$ and $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$;
- **Ours (Fwd Only)** uses ReSTIR for estimating $\boldsymbol{I}$ and RIS for $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$;
- **Ours** is our full method, using ReSTIR for $\boldsymbol{I}$ and $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$.

Further, we use $\mathcal{L}(\boldsymbol{I}) := \|\boldsymbol{I} - \boldsymbol{I}_0\|_2^2$ with some fixed $\boldsymbol{I}_0$ for gradient $\partial_{\boldsymbol{I}}\mathcal{L}$ (computed via automatic differentiation). As shown in Figures 9, both "Ours (Fwd Only)" and "Ours" use ReSTIR for forward rendering, so they enjoy low-variance estimates of $\boldsymbol{I}$ and gradient $\partial_{\boldsymbol{I}}\mathcal{L}$.[5] But "Ours (Fwd Only)" does not apply ReSTIR for derivative $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$, producing much the same noise as the "RIS" baseline. Thus this approach's final gradient $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$, given by the chain rule, also suffers from high variance. We note that, while $\mathrm{d}_{\boldsymbol{\theta}}\boldsymbol{I}$ is not explicitly computed for radiative backpropagation methods [Nimier-David et al. 2020; Vicini et al. 2021], the variance in this term's estimates still affect the final gradient $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$.

To further show $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$'s impact, we compare inverse-rendering performance on the Tree in Figure 10, where we optimize tree rotation angle by only looking at its shadow. We reuse the optimization setup (e.g., optimizer and learning rates) and adjust light candidate counts (i.e., M) so all methods take roughly *equal iteration time*.

As demonstrated in Figure 10, both "RIS" and "Ours (Fwd Only)" fail to converge to ground truth due to high variance in the gradient $\mathrm{d}_{\boldsymbol{\theta}}\mathcal{L}$. Our full method "Ours", however, allows the optimization to converge smoothly.

8.2 Inverse-Rendering Comparisons

We now show further inverse-rendering comparisons with our baselines (**B.1** and **B.2**). We adjust sample counts so all methods take roughly equal optimization time (including our burn-in from §7).

Please see Table 1 for performance statistics and the supplemental material for animated versions of these results.

[5] Another major benefit for having low-variance forward rendering $\boldsymbol{I}$ is to lower the bias of the gradient $\partial_{\boldsymbol{I}}\mathcal{L}$ when $\partial_{\boldsymbol{I}}\mathcal{L}$ is nonlinear with respect to $\boldsymbol{I}$.

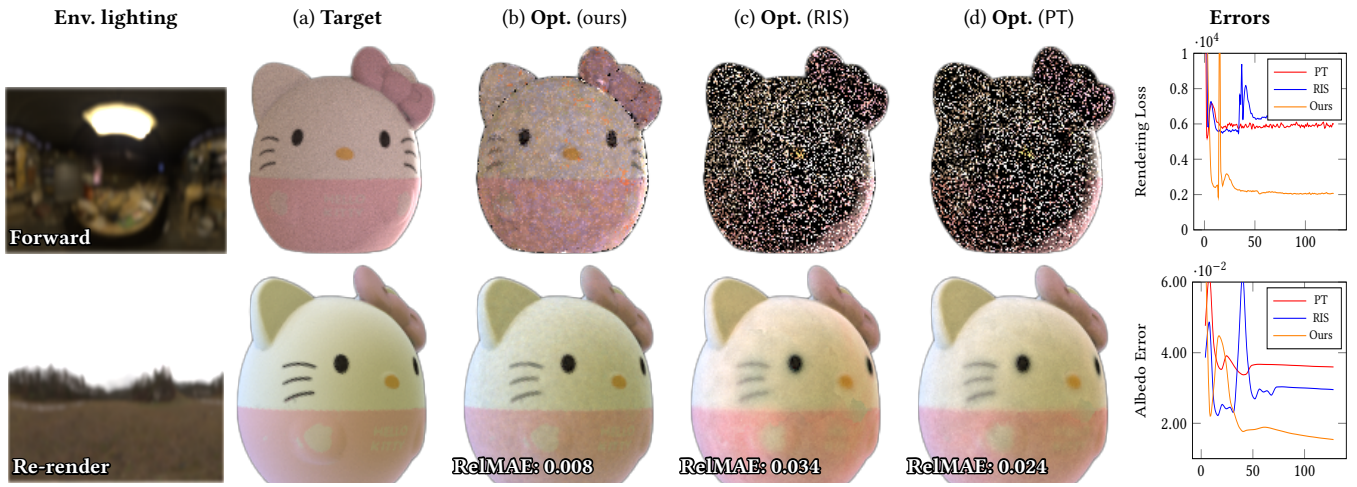


Fig. 12. **Inverse-rendering comparison** using the diffuse "Kitty" under environmental illumination. We optimize the spatially varying albedo (initialized with a constant texture) of the Kitty. (c, d) The baseline methods (**B.1** and **B.2**) produce high variance on the front of the Kitty, causing over-blurring and color shifts in the reconstructions. (b) Our method, on the other hand, provides significantly more accurate results.

Texture optimization. We show texture reconstruction in Figures 1, 11, and 12. As there is no differentiation with geometry, we apply ReSTIR only for forward rendering and the interior component (§4).

Figure 1 contains an "Oxalis" scene with a painting is lit by two bright spotlights and a dim fill light. Initialized with a constant gray image, we optimize the spatially varying painting albedo. The baselines perform okay where directly lit by bright spotlights and very poorly in darker regions lit only by the fill light. Baseline reconstruction results severely overestimate albedos inside the spotlights and underestimate those outside. With spatiotemporal reuse, our method performs significantly better than baselines in darker regions, yielding a far less biased reconstruction.

Figure 11 uses a "Painting" scene modeled after the Veach Ajar scene, with a new fill light in the room. Initialized with a constant gray image, we optimize the spatially varying albedo of the painting. The bright emitter outside the room introduces extreme noise in both PT (**B.1**) and RIS (**B.2**) (row marked "forward"), causing reconstruction failures. But our method quickly adapts, mainly sampling the fill light, and nicely reconstructs the target texture (see row labeled "Texture vis.").

Figure 12 shows a "Kitty" under environmental illumination. It is backlit (i.e., most light from behind), causing the baselines to produce high variance on the front. Our method produces much cleaner estimates on the front, giving a less biased and more detailed reconstruction (demonstrated by the bottom row re-renderings).

Shape optimization. Lastly, we show shape reconstruction in Figures 13 and 14. In both, we use one or multiple images of an object under complex illumination and optimize object shape (expressed as triangle meshes). As geometric derivatives are computed, we apply our method interior (§4) and boundary (§5) components.

Figure 13 uses an "Octagon" lit by 17 spotlights, casting shadows on the walls (shown in "Config."). Using 17 images of casting shadows on the walls, we reconstruct the geometry from a sphere.

Figure 14 use a "World Map" where a relief is on a wall lit by a large area light partially occluded through a window (shown in "Config."). Initialized with a flat rectangle, we optimize the relief shape using six input images (with only one shown).

For both examples, because of the complex lighting, both PT (**B.1**) and RIS (**B.2**) produce noisy forward renderings and gradient estimates. This causes inaccurate reconstructions with visible artifacts (e.g., along the left map edge). Our method, on the contrary, offers the robustness to produce clean reconstructions.

9 DISCUSSION AND CONCLUSION

Limitations and future work. Our technique builds on Bitterli et al.'s [2020] early formulation that supports only direct lighting. Extending our technique to handle more advanced reuse [Lin et al. 2021, 2022] to support multi-bounce light transport is an important topic for future research.

Also, when using multiple input images, our technique focuses only on multi-view configurations observing a static scene under different viewing conditions. Extending our work to support settings that allow, for example, illumination conditions and object poses to change across images would benefit many inverse-rendering applications.

Lastly, other forms of amortized sampling beyond ReSTIR's spatiotemporal reuse are worth exploring.

Conclusion. In this paper, we introduced a technique that exploits temporal consistency in the context of physics-based inverse direct illumination. Specifically, by adopting the reservoir-based spatiotemporal important resampling (ReSTIR) framework, we developed new Monte Carlo methods to efficiently estimate both interior and boundary components of differential illumination integrals under complex illumination conditions. Incorporating these ReSTIR-based estimators, we further proposed a new pipeline for physics-based inverse direct illumination.

We validated our technique by comparing derivatives estimated with unbiased baselines and our methods. Additionally, we demonstrated the effectiveness of our method using several differentiable-rendering and inverse-rendering experiments.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful suggestions. We are also grateful to Aaron Lefohn for his support. This work started when Yu-Chen Wang was an intern at NVIDIA and was partially funded by NSF grant 1900927 and an NVIDIA gift.

REFERENCES

Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. 2018. *Real-Time Rendering, Fourth Edition* (4th ed.). A. K. Peters, Ltd., USA.

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020), 245:1–245:18.

Louis Bavoil and Miguel Sainz. 2009. Multi-Layer Dual-Resolution Screen-Space Ambient Occlusion. In *SIGGRAPH 2009: Talks* (New Orleans, Louisiana) *(SIGGRAPH '09)*. Association for Computing Machinery, New York, NY, USA, Article 45, 1 pages. https://doi.org/10.1145/1597990.1598035

Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. 2020. Spatiotemporal Reservoir Resampling for Real-Time Ray Tracing with Dynamic Direct Lighting. 39, 4 (2020), 148:1–148:17.

John Burgess. 2020. RTX on—The NVIDIA Turing GPU. *IEEE Micro* 40, 2 (2020), 36–44. https://doi.org/10.1109/MM.2020.2971677

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for Differentiable and Inverse Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*. 10 pages.

Min-Te Chao. 1982. A general purpose unequal probability sampling plan. *Biometrika* 69, 3 (1982), 653–656.

Johannes Deligiannis and Jan Schmid. 2019. 'It Just Works': Ray-traced Reflections in 'Battlefield V'. Game Developers Conference. https://www.gdcvault.com/play/1026282/It-Just-Works-Ray-Traced

Frédo Durand and Julie Dorsey. 2002. Fast Bilateral Filtering for the Display of High-Dynamic-Range Images. *ACM Trans. Graph.* 21, 3 (jul 2002), 257–266. https://doi.org/10.1145/566654.566574

Andreas Griewank and Andrea Walther. 2008. *Evaluating Derivatives* (second ed.). Society for Industrial and Applied Mathematics.

Eric Heitz, Stephen Hill, and Morgan McGuire. 2018. Combining Analytic Direct Illumination and Stochastic Shadows. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (Montreal, Quebec, Canada) *(I3D '18)*. Association for Computing Machinery, New York, NY, USA, Article 2, 11 pages. https://doi.org/10.1145/3190834.3190852

Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *ACM Trans. Graph.* 41, 4 (2022), 124:1–124:19.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6, Article 194 (nov 2020), 14 pages. https://doi.org/10.1145/3414685.3417861

Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. 2018. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Trans. Graph.* 37, 6 (2018), 222:1–222:11.

Daqi Lin, Markus Kettunen, Benedikt Bitterli, Jacopo Pantaleoni, Cem Yuksel, and Chris Wyman. 2022. Generalized Resampled Importance Sampling: Foundations of ReSTIR. *ACM Trans. Graph.* 41, 4 (2022), 75:1–75:23.

Daqi Lin, Chris Wyman, and Cem Yuksel. 2021. Fast Volume Rendering with Spatiotemporal Reservoir Resampling. *ACM Trans. Graph.* 40, 6 (2021), 279:1–279:18.

Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. 2019. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *The IEEE International Conference on Computer Vision (ICCV)* (Oct 2019).

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* 38, 6 (2019), 228:1–228:14.

Zander Majercik, Jean-Philippe Guertin, Derek Nowrouzezahrai, and Morgan McGuire. 2019. Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields. *Journal of Computer Graphics Techniques (JCGT)* 8, 2 (5 June 2019), 1–30. http://jcgt.org/published/0008/02/01/

Morgan McGuire and Michael Mara. 2014. Efficient GPU Screen-Space Ray Tracing. *Journal of Computer Graphics Techniques (JCGT)* 3, 4 (9 December 2014), 73–85. http://jcgt.org/published/0003/04/04/

Morgan McGuire, Michael Mara, Derek Nowrouzezahrai, and David Luebke. 2017. Real-Time Global Illumination using Precomputed Light Field Probes. In *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. 11. https://www.casual-effects.com/research/McGuire2017LightField/index.html I3D 2017.

Baptiste Nicolet, Fabrice Rousselle, Jan Novák, Alexander Keller, Wenzel Jakob, and Thomas Müller. 2023. Recursive Control Variates for Inverse Rendering. *ACM Trans. Graph.* 42, 4 (2023).

Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 146:1–146:15.

Yaobin Ouyang, Shiqiu Liu, Markus Kettunen, Matt Pharr, and Jacopo Pantaleoni. 2021. ReSTIR GI: Path Resampling for Real-Time Path Tracing. *Computer Graphics Forum* 40, 8 (2021), 17–29. https://doi.org/10.1111/cgf.14378

Mark Pauly, Thomas Kollig, and Alexander Keller. 2000. Metropolis light transport for participating media. In *Rendering Techniques 2000*. Springer, 11–22.

Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically Based Rendering: From Theory to Implementation* (3rd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. 1266 pages.

Christoph Schied, Anton Kaplanyan, Chris Wyman, Anjul Patney, Chakravarty R Alla Chaitanya, John Burgess, Shiqiu Liu, Carsten Dachsbacher, Aaron Lefohn, and Marco Salvi. 2017. Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In *Proceedings of High Performance Graphics*. 1–12. https://doi.org/10.1145/3105762.3105770

Dario Seyb, Peter-Pike Sloan, Ari Silvennoinen, Michał Iwanicki, and Wojciech Jarosz. 2020. The design and evolution of the UberBake light baking system. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39, 4 (July 2020).

Justin F. Talbot, David Cline, and Parris Egbert. 2005. Importance Resampling for Global Illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques (EGSR '05)*. 139–146.

Eric Veach. 1997. *Robust Monte Carlo methods for light transport simulation*. Vol. 1610. Stanford University PhD thesis.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths Using Constant Memory and Linear Time. *ACM Trans. Graph.* 40, 4, Article 108 (2021), 108:1–108:14 pages.

Ingo Wald. 2022. A Stack-Free Traversal Algorithm for Left-Balanced k-d Trees. https://doi.org/10.48550/ARXIV.2210.12859

Kai Yan, Christoph Lassner, Brian Budge, Zhao Dong, and Shuang Zhao. 2022. Efficient estimation of boundary integrals for path-space differentiable rendering. *ACM Trans. Graph.* 41, 4 (2022), 123:1–123:13.

Zihan Yu, Cheng Zhang, Derek Nowrouzezahrai, Zhao Dong, and Shuang Zhao. 2022. Efficient Differentiation of Pixel Reconstruction Filters for Path-Space Differentiable Rendering. *ACM Trans. Graph.* 41, 6 (2022), 191:1–191:16.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo estimators for differential light transport. *ACM Trans. Graph.* 40, 4 (2021), 78:1–78:16.

Cheng Zhang, Zhao Dong, Michael Doggett, and Shuang Zhao. 2021a. Antithetic sampling for Monte Carlo differentiable rendering. *ACM Trans. Graph.* 40, 4 (2021), 77:1–77:12.

Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. 2020. Path-space differentiable rendering. *ACM Trans. Graph.* 39, 4 (2020), 143:1–143:19.

Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. 2019. A differential theory of radiative transfer. *ACM Trans. Graph.* 38, 6 (2019), 227:1–227:16.

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021b. Path-space differentiable rendering of participating media. *ACM Trans. Graph.* 40, 4 (2021), 76:1–76:15.

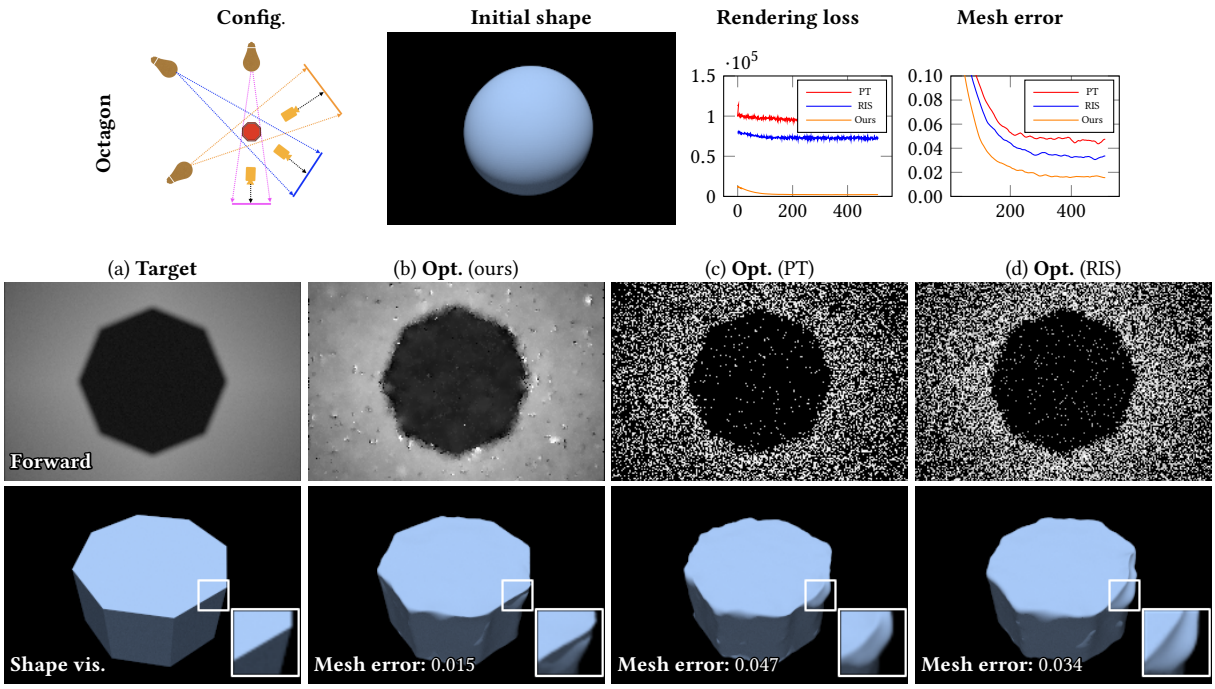


Fig. 13. **Inverse-rendering comparison** using an "Octagon" lit by 17 spotlights, casting several shadows on the walls. We optimize the shape (initialized as a sphere) by merely looking at shadows on the walls. (c, d) The baseline methods (B.1 and B.2) suffer from high variance, leading to artifacts on the reconstructed meshes. (b) Thanks to low-variance forward rendering and gradient estimates, our result closely matches ground truth.

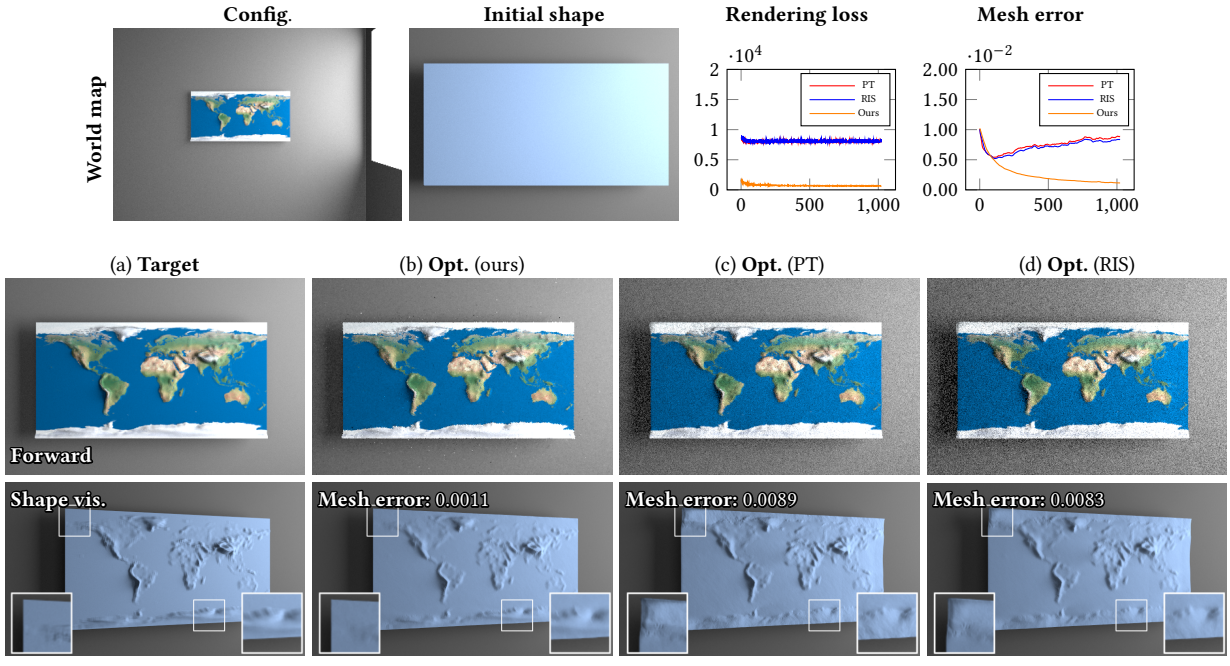


Fig. 14. **Inverse-rendering comparison** using a "World map" relief lit by a large area light partially occluded by a window. We optimize the shape (initialized as a rectangle) of the relief. (c, d) The baseline methods (B.1 and B.2) suffer from high variance, giving artifacts on the left edge (c) and less detailed reconstructions (d). (b) Our method, on the other hand, is capable of producing more accurate and artifact-free results.