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

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 100x lower relative error than baseline methods. Additionally, we propose an inverse-rendering pipeline that incorporates these estimators and provides reconstructions with up to 20x lower error.

CCS Concepts: • Computing methodologies → Rendering.

ACM Reference Format:

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

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
© 2023 Copyright held by the owner/author(s).
0730-0301/2023/12-ART214
https://doi.org/10.1145/3618331

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 100x lower relative error than baseline methods. Additionally, we propose an inverse-rendering pipeline that incorporates these estimators and provides reconstructions with up to 20x lower error.
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 multiview 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-Möller 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 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

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

where $\Omega = \mathbb{M}^3$ (with $\mathbb{M}$ the union of all surfaces) is the one-bounce path space containing light paths $x = (x_0, x_1, x_2)$ with $x_0$ on the light and $x_2$ on the detector; $\mu$ is the area-product measure given by $d\mu(x) = \prod_{n=0}^{2} dA(x_n)$; and $f$ is the measurement contribution function defined as

$$f(x) = L_e(x_0 \rightarrow x_1) \, G(x_0 \rightarrow x_1) \, f_s(x_0 \rightarrow x_1 \rightarrow x_2) \, G(x_1 \rightarrow x_2) \, W_e(x_1 \rightarrow x_2).$$

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 $x_2$ that intersects the scene at surface $x_1 \in \mathbb{M}$, and (ii) sampling point $x_0$—which we term a light vertex—on the surface of a light source. This gives a Monte Carlo estimator:

$$I = \frac{W_e(x_1 \rightarrow x_2) \, G(x_1 \rightarrow x_2)}{p(x_1, x_2)} \, \frac{L_e(x_0 \rightarrow x_1) \, f_s(x_0 \rightarrow x_1 \rightarrow x_2) \, G(x_0 \rightarrow x_1)}{p(x_0 | x_1, x_2)}.$$  

for $p(x_1, x_2)$ the joint probability density of sampling $x_1, x_2$ and $p(x_0 | x_1, x_2)$ the conditional probability density of drawing $x_0$.

In practice, the camera ray from $x_2$ to $x_1$ can be sampled using standard ray tracing. In the rest of this paper, we assume the perspective pinhole camera model. In this case, $x_2$ becomes fixed at the center of projection and $x_1$ is obtained by tracing a ray from $x_2$ through a randomly selected point on the image plane.

With $x_1$ and $x_2$ obtained, importance sampling the light vertex $x_0$ remains challenging—especially under complex illumination and visibility. Bitterli et al. [2020] recently introduced a technique that efficiently samples $x_0$. We briefly revisit this work in §3.2 and build on it later.

3.2 Reservoir-based SpatioTemporal Importance Resampling (ReSTIR) for Direct Illumination

Resampled importance sampling. Monte Carlo estimation of

$$I = \int h(x) \, dx,$$

requires generating samples $x$ 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 $\{x_1, \ldots, x_M\}$ from some other distribution $p_c$. Then, one sample $y$ is selected from the $M$ candidates with (discrete) probability $\mathbb{P}[y = x_m]$ proportional to

$$w(x_m) = \frac{\hat{p}(x_m)}{p_c(x_m)},$$

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

$$\langle I \rangle_{\text{RIS}} = \frac{\hat{h}(y)}{\hat{p}(y)} \left( \frac{1}{M} \sum_{m=1}^{M} w(x_m) \right).$$

Naive implementations of Eq. (6) generate and store all $M$ candidates $x_1, \ldots, x_M$ before selecting final sample $y$. This can be inefficient. Bitterli et al. [2020] proposed a streaming version using

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

begin
  1. UpdateReservoir($r$, $x$, $w$)
  2. Input: A reservoir $r$, a new candidate $x$ with the weight $w$
  3. begin
  4. $r.w_{\text{sum}} \leftarrow r.w_{\text{sum}} + w$
  5. if rand() < $w/r.w_{\text{sum}}$ then
  6. $r.y \leftarrow x$
  7. $r.M \leftarrow r.M + 1$
  8. end
  9. StreamingRIS($M$)
  10. Input: Number of candidates $M \geq 1$
  11. begin
  12. $r.w_{\text{sum}} \leftarrow 0$
  13. $r.M \leftarrow 0$
  14. for $m = 1, 2, \ldots, M$ do
  15. 16. Draw $x_m \sim p_c$
  17. 18. UpdateReservoir($r$, $x$, $\hat{p}(x_m)/p_c(x_m)$) ; // Updating $r$
  19. end
  20. return $r$
end
```

Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR  •  214:3
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 processing candidate \( x_m \) inside UpdateReservoir\((r)\), reservoir \( r \) stores the following data: (i) a sum of weights \( r.w_{\text{sum}} = \sum_{i=1}^{m} w(x_i) \); (ii) a count of candidates processed \( r.M = m \); and (iii) a sample \( r.y \) randomly drawn from prior candidates \( x_1, \ldots, x_m \) with selection probability \( \frac{2}{|r.y = x_i|} = \frac{w(x_i)}{r.w_{\text{sum}}} \) for all \( i = 1, 2, \ldots, m \). One can easily evaluate the RIS estimator in Eq. (6) using the reservoir \( r \) output from StreamingRIS\((r)\) using \((f)_m = h(r.y) r.W\).

Reservoir-based spatiotemporal importance resampling. To estimate direct illumination using Eq. (3), after generating the camera ray \( x_2 \rightarrow x_1 \), RIS (Algorithm 1) can be used to select the light vertex \( x_0 \) by setting the target function \( \hat{p} \) to the measurement contribution of the path \((x_0, x_1, x_2)\). 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 \( r.W \) after determining the sample \( r.y \), as expressed in Algorithm 1.\(^1\) 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 \( q_i \). Furthermore, precautions must be taken when evaluating \( \hat{p}^{(q_i)}(r.y) \) (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 \( r.y \) 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 material-form parameterization to facilitate differentiating Eq. (1) with respect to \( \theta \). This parameterization introduces a motion \( \chi \) such that for any \( \theta \), \( \chi(\cdot, \theta) \) is a differentiable one-to-one mapping from some predetermined reference configuration \( \mathcal{B} \) (independent of parameters \( \theta \)) to the scene geometry \( \mathcal{M}(\theta) \). To distinguish points in the ordinary geometry from those in the reference configuration, we term any \( x \in \mathcal{M}(\theta) \) as spatial points and \( p \in \mathcal{B} \) as material points.

The mapping \( \chi(\cdot, \theta) \) induces a path-level map \( \hat{\chi}(\cdot, \theta) \) that takes a one-bounce material path \( \hat{p} = (p_0, p_1, p_2) \) to an ordinary path \( \hat{x} = (x_0, x_1, x_2) \) where \( x_n = \chi(p_{n+1}; \theta) \) for \( n = 0, 1, 2 \).

Applying the change of variable of \( \hat{x} = \hat{\chi}(\hat{p}, \theta) \) to Eq. (1) yields its material-form variant:

\[
I = \int_{\hat{\Omega}} \hat{f}(\hat{p}) \, d\mu(\hat{p}),
\]

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

\[
\hat{f}(\hat{p}) := f(x) \left| \frac{d\mu(x)}{d\mu(\hat{p})} \right|,
\]

with \( \|d\mu(x)/d\mu(\hat{p})\| = \prod_{n=0}^2 \|d\mu(x_n)/d\mu(p_n)\| \) being the Jacobian determinant capturing this change of variable.

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

\[
\frac{df}{d\theta} = \int_{\hat{\Omega}} \frac{\partial}{\partial\theta} \hat{f}(\hat{p}) \, d\mu(\hat{p}) + \int_{\partial\hat{\Omega}} \hat{f}(\hat{p}) \left( \frac{\partial}{\partial\theta} \mu_{\text{int}} \right) \frac{\partial p_K}{\partial\theta} \, d\mu(\hat{p}) + \int_{\partial\hat{\Omega}} \hat{f}(\hat{p}) \left( \frac{\partial}{\partial\theta} \mu_{\text{bdy}} \right) \frac{\partial p_K}{\partial\theta} \, d\mu(\hat{p}),
\]

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 \( \theta \). For direct lighting, \( \partial\hat{\Omega} \) comprises material boundary paths \( \hat{p} = (p_0, p_1, p_2) \) with exactly one boundary segment \( p_K \) \( K \in \{0, 1\} \) such that \( x_K = \chi(p_K; \theta) \) is a jump discontinuity of the mutual visibility function \( \psi(x_K \leftarrow x_{K+1}) \) when fixing \( x_{K+1} = \chi(p_{K+1}; \theta) \). Further,
According to the chain rule, the gradient $\psi_\ell$ being the curve-length measure.

We do this by left-multiplying the gradient $\psi$ essentially the detached copy of its spatial representation $x = \chi(p, \theta)$ at $\theta = \theta_0$.

We note that while the Jacobian $\|d\psi(x)/d\psi(p)\|$ in Eq. (8) is one at $\theta = \theta_0$, its gradient with respect to $\theta$ generally remains nonzero.

3.4 Inverse Rendering and Radiative Backpropagation

Inverse-rendering optimizations. Optimization-based inverse rendering, or by synthesis, infers scene parameters $\theta \in \mathbb{R}^{m_\theta}$ by minimizing a predetermined rendering loss $L$ (e.g., $L_1$ or $L_2$) between rendered images $I \in \mathbb{R}^{m_I}$ with $m_I$ pixels and user-specified targets $I_0 \in \mathbb{R}^{m_I}$:

$$\arg \min_{\theta} L(I(\theta); I_0).$$  (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 $L$ with respect to parameters $\theta$. According to the chain rule, the gradient $d_{\theta} L := \partial L/\partial \theta$ satisfies

$$d_{\theta} L = (\partial L/\partial I) (d_{\theta} I),$$  (11)

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

Radiative backpropagation. When the number of parameters $m_\theta$ and pixels $m_I$ are large, storing the rendered image $I$ 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 $\partial L$ on both sides of Eq. (9), yielding:

$$d_{\theta} L = \int_{\Omega} (\partial L/\partial I)[q] \left( \frac{d}{d\theta} \hat{f}(\hat{\psi}) \right) d\psi(\hat{\psi}) + \int_{\partial I}(\partial L/\partial I)[q] \left( \frac{d}{d\theta} \hat{f}(\hat{\psi}) \right) d\psi(\hat{\psi}),$$  (12)

where $q$ denotes the pixel index where the path $\hat{\psi}$ contributes, and $(\partial L/\partial I)[q] \in \mathbb{R}$ indicates the value in $\partial L/\partial I$ at pixel $q$ (where $\partial L/\partial I$ have the same shape). Also, $\hat{f}(\hat{\psi})$ denotes the material measurement contribution of the path $\hat{\psi}$ to pixel $q$, and $\partial L/\partial \theta$ indicates the difference in $\hat{f}(\hat{\psi})$ 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 REST FOR INTERIOR INTEGRALS

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

$$(d\theta L)_{\text{int}} := \int_{\Omega} (\partial L/\partial I)[q] \left( \frac{d}{d\theta} \hat{f}(\hat{\psi}) \right) d\psi(\hat{\psi}).$$  (13)

When computing gradients at a fixed $\theta = \theta_0$, as discussed in §3.3, we set the reference configuration to $B := M(\theta_0)$. Note that during inverse-rendering optimization, $\theta_0$ 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 $\hat{\psi} = (p_0, p_1, p_2)$ since the material path space $\Omega$ coincides with the ordinary one $\Omega(\theta_0)$, we can sample $\hat{\psi}$ by repurposing methods developed for forward rendering.

We first construct a camera ray from $p_0$ and compute the closest intersection $p_1$ between this ray and the scene (specifically, the reference configuration $B$). With $p_0$ and $p_2$ determined, we apply streaming RIS (Algorithm 1) to sample $p_0$ on a light source by letting $\hat{\psi} = (r, y, p_1, p_2)$ and $r, y$ is the reservoir’s stored sample. In practice, we efficiently compute Eq. (15) by applying reverse-mode automatic differentiation (AD) to

$$\detach \left( \frac{d}{d\theta} \hat{f}(\hat{\psi}) \right) \hat{f}(\hat{\psi}),$$  (16)

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

4.2 Spatiotemporal Reuse

Building on our differentiable 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.

---

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

---

1On the left hand side of Eq. (14), we omit the dependency of $\hat{\psi}$ on $p_1$ because, for perspective pinhole cameras—which is the case we focus on, $p_1$ is constant.
Algorithm 3: Combining reservoirs from spatial and temporal neighbors

Input: Reservoir \( r \); reservoirs \( Q = \{ r_1, \ldots, r_k \} \) to combine
Output: One combined reservoir

begin
for \( i = 1, \ldots, k \) do
    \( \text{UpdateReservoir}(r, r_i, y, \hat{\rho}(r, y; r, p) r.W \ r.M) \)
end
\( r.M \leftarrow \sum_{i=1}^{k} r_i.M \);
\( Z \leftarrow 0; \)
for \( i = 1, \ldots, k \) do
    if \( \hat{\rho}(r, y; r_i, p) > 0 \) then
        \( Z \leftarrow Z + r_i.M \);
end
\( r.W \leftarrow r.wsum/(\hat{\rho}(r, y; r, p) Z); \)
return \( r \);
end

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 \( \rho = (p_0, p_1, p_2) \) that are later used to compute gradients. As discussed in §4.1, we sample the path by: (i) constructing a camera ray from \( p_2 \) on the detector and intersecting the scene at \( p_1 \) (Line 6 of Algorithm 4); and (ii) drawing \( p_0 \) 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 \( p_1 \) (where RIS occurs) in each reservoir as \( r.p \) (Line 8). After obtaining a reservoir \( r \) using RIS, we find neighbor reservoirs \( Q \) such that each \( r' \in Q \) has its shading point \( r'.p \) near \( r.p \) (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 \( \hat{\rho}(::p') \) defined with respect to a shading point \( p' \) (as opposed to a pixel). The light sample \( r.y \) after combining reservoirs completes the path sample \( \rho = (r.y, p_1, p_2) \).

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

To keep \( r.y \) and \( r.p \) updated, we focus on the common inverse-rendering setting that represents scene geometries \( M(\theta_{prev}) \) and \( M(\theta_{cur}) \) with triangle meshes of identical topology (but possibly differing vertex positions). We record indices of triangles containing \( r.y \) and \( r.p \) and corresponding barycentric coordinates. Then, during the updateInteriorReservoirs() on Line 3, we recompute \( r.y \) and \( r.p \) 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).

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

4.4 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 \( r.p \) over the reference surface \( B \). Doing so not
We now show how to apply ReSTIR to estimate the boundary component of Eq. (12) from §3.3:

$$\left(\partial \mathcal{L}^{\text{bnd}}_{\text{res}}\right)_{\text{hid}} = \int_{\partial \Omega} (\partial_2 \mathcal{L}[q] \cdot \hat{\mathbf{n}}_{\mathbf{p}_K}^+ (\mathbf{p}) \cdot \nabla \phi (\mathbf{p}_K) \left(\frac{d \mathbf{p}_K}{d \theta}\right)) d \phi (\mathbf{p}).$$

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 $\mathbf{p}$ 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 $\mathbf{p}_K \mathbf{p}_{K+1}$. Any path with $K = 1$ has endpoint $\mathbf{p}_2$ on the detector and is called a primary boundary path; any with $K = 0$ has endpoint $\mathbf{p}_1$ 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, only allowing 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 $\mathbf{p}_0$ to the shading point $\mathbf{p}_1$ for faster forward rendering:

$$\hat{b}(\mathbf{p}_0) = L_e(\mathbf{p}_0 \rightarrow \mathbf{p}_1) f_r(\mathbf{p}_0 \rightarrow \mathbf{p}_1) G_0(\mathbf{p}_0 \mapsto \mathbf{p}_1),$$

with $G_0$ the geometric term $G$ with visibility $\mathbb{V}(\mathbf{p}_0 \mapsto \mathbf{p}_1)$ 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 $\mathbf{p}_0$ 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, $\mathbf{n}(r', \mathbf{p}) \cdot \mathbf{n}(r, \mathbf{p}) \ll 1$ with $\mathbf{n}(\mathbf{p})$ the surface normal at $\mathbf{p}$. In this case, we follow Bitterli et al. and exclude any reservoirs $r'$ with $\mathbf{n}(r', \mathbf{p}) \cdot \mathbf{n}(r, \mathbf{p})$ 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:

$$\left(\partial \mathcal{L}^{\text{bnd}}_{\text{res}}\right)_{\text{hid}} = \int_{\partial \Omega} (\partial_2 \mathcal{L}[q] \cdot \Delta_{\mathbf{p}_K}^{\hat{r}_q}(\mathbf{p}) \cdot \nabla \phi (\mathbf{p}_K) \left(\frac{d \mathbf{p}_K}{d \theta}\right)) d \phi (\mathbf{p}).$$

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

$$\left(\partial \mathcal{L}^{\text{pri}}_{\text{res}}\right)_{\text{hid}} = \frac{(\partial \mathcal{L}[q] \cdot \Delta_{\mathbf{p}_1}^{\hat{r}_q}(\mathbf{p}) \cdot \nabla \phi (\mathbf{p}_1) \left(\frac{d \mathbf{p}_1}{d \theta}\right))}{p(\mathbf{p}_1) \cdot w} \cdot 1.$$

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

$$\text{detach} \left(\frac{(\partial \mathcal{L}[q] \cdot \Delta_{\mathbf{p}_1}^{\hat{r}_q}(\mathbf{p}) \cdot \nabla \phi (\mathbf{p}_1) \left(\frac{d \mathbf{p}_1}{d \theta}\right)}{p(\mathbf{p}_1) \cdot w} \right) \cdot 1.$$
ALGORITHM 5: Our streaming RIS with spatiotemporal reuse for primary boundary paths

1 EstPrimaryBound(∂f, L, prevReservoirs)
2 begin
3 \((d_\partial L)^{\text{pri}}_\text{bnd} \leftarrow 0;\)
4 /* Sampling boundary paths (non-differentiable) */
5 paths \(\left\{\right\};\)
6 for \(i = 1 \text{ to } N_{\text{ord}}\) do
7 Sample \(p_1\) with pdf \(p(p_1);\) /* MC edge sampling*/
8 Let \(q\) be the index of the pixel intersecting \(p_1, p_2;\)
9 /* Generate initial candidates */
10 Generate reservoir \(r\) using stream RIS; // §4.1
11 /* Spatiotemporal reuse */
12 \(Q \leftarrow \text{PickNeighbors(prevReservoirs, } r);\) /* §4.2
13 \(r \leftarrow \text{SpatiotemporalReUse}(r, Q);\) // Alg. 3
14 /* Store sampled path */
15 paths[\(i\)].pid \(\leftarrow q;\) // Record pixel index
16 paths[\(i\)].vtx \(\leftarrow (r.y, p_1, p_2);\)
17 /* Record the detached factor of Eq. (20) */
18 pdf \(\leftarrow p(p_1) N_{\text{ord}} / r.W;\)
19 paths[\(i\)].vec \(\leftarrow (d_\partial L)[q] \Delta f^r(\hat{p})(p) n_\partial(p_1)/pdf\)
20 end
21 /* Gradient computation (differentiable) */
22 for \(i = 1 \text{ to } N_{\text{ord}}\) do
23 \(q, \hat{p}, v \leftarrow \text{paths[}\!i\!.\!\text{pid}, \text{paths[}\!i\!.\!\text{vtx}, \text{paths[}\!i\!.\!\text{vec};}\)
24 Re-compute \(p_1\) using differentiable ray intersection;
25 \(I \leftarrow \text{detach}(v) : p_1;\) // Eq. (20)
26 /* Back-propagate gradients */
27 \((d_\partial L)^{\text{pri}}_\text{bnd} \leftarrow (d_\partial L)^{\text{pri}}_\text{bnd} + \text{backward}(I);\)
28 end
29 return \((d_\partial L)^{\text{pri}}_\text{bnd};\)
30 end

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 \(p^B\) from a triangle edge with probability \(p(p^B);\) sample light vertex \(p_0\) with conditional probability \(p(p_0 | p^B).\)

**S.2** If \(p_0\) and \(p^B\) are mutually visible, trace a ray from \(p^B\) in direction \(p_0, p^B\) to find intersection \(p_1.\) This guarantees \(p_0, p_1\) is a boundary segment.

**S.3** Complete path \(\hat{p} = (p_0, p_1, p_2)\) with \(p_2\) being the perspective camera’s center of projection.

This sampling process gives an ordinary Monte Carlo estimator:

\[
\frac{(d_\partial L)^{\text{bnd}}_\text{bnd} \leftarrow \frac{f^{\text{bnd}}_\text{bnd}(p^B, p_0, p_2)}{p(p^B) p(p_0 | p^B)} n_\partial(p_0)^\top \frac{dp_0}{d\theta}},
\]

where

\[
f^{\text{bnd}}_\text{bnd}(p^B, p_0, p_2) := (d_\partial L)[q] \Delta f^0(q)(p) f^B(p^B, p_0),
\]

with

\[
f^B(p^B, p_0) := \left\| \frac{dA(p_1)}{d\theta} \frac{dp_0}{d\theta} \right\|.
\]

the Jacobian determinant for changing from joint probability density \(p(p_0, p_1)\) to \(p(p^B, p_0).\) Refer to Zhang et al. [2020] for more details, including the derivation of \(f^B.\)

**Streaming RIS.** The first step (S.1) in multi-directional sampling draws light vertex \(p_0\) given \(p^B\) on a triangle edge. For scenes with complex illumination and visibility, sampling \(p_0\) via streaming RIS (Algorithm 1) improves efficiency. To this end, given \(p^B\) we set \(\hat{p}\) as

\[
\hat{p}(p_0, p^B) = f^{\text{bnd}}_\text{bnd}(p^B, p_0, p_2),
\]

with \(f^{\text{bnd}}_\text{bnd}\) defined in Eq. (22), and standard light sampling with pdf \(p_0.\) Here we neglect gradient term \(n_\partial(p_0)^\top \frac{dp_0}{d\theta}\) as it is difficult to obtain in the path sampling stage. This leads to our RIS estimator:

\[
\left\langle (d_\partial L)^{\text{bnd}}_\text{bnd} \right\rangle_{\text{RIS}}^M \frac{f^{\text{bnd}}_\text{bnd}(p^B, p_0, p_2) r.W}{\frac{n_\partial(p_0)^\top \frac{dp_0}{d\theta}}{p(p^B)}},
\]

which essentially replaces \(p(p_0, p^B)\) in the ordinary Monte Carlo estimator of Eq. (21) with \(r.W / p(p^B),\) as \(p_0\) is now drawn with RIS.

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

\[
\frac{\partial \left\langle (d_\partial L)^{\text{bnd}}_\text{bnd} \right\rangle_{\text{RIS}}^M}{\partial p(p^B)} \frac{f^{\text{bnd}}_\text{bnd}(p^B, p_0, p_2) r.W}{\frac{n_\partial(p_0)^\top \frac{dp_0}{d\theta}}{p(p^B)}}.
\]

**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 \(p^B\) in each reservoir \(r\) (Line 9) and use this point for nearest neighbor searches (Line 10), rather than using shading point \(p_1\) as in our other computations. After merging \(r\) with nearby reservoirs from the previous gradient step, we trace a ray to obtain \(p_1 (S.2)\) to complete the path sample \(\hat{p} = (r.y, p_1, p_2).\)

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 \(p^B.** Before drawing \(p_0\) on a light, Zhang et al.’s [2020] multi-directional sampling picks \(p^B\) from a triangle edge (Line 7). A naïve implementation could (i) select an edge \(e_k\) with probability \(p_k\) proportional to edge length \(\|e_k\|;\) and (ii) draw \(p^B\) uniformly from \(e_k.\) But this loses efficiency in complex scenes if it frequently returns samples \(p^B\) not visible to any light.

To address this problem, we leverage our reservoirs to guide edge sampling as follows. For an edge \(e_k,\) containing \(n_k\) reservoirs (i.e., \(n_k\) reservoirs \(r\) with \(r, p\) lying on \(e_k),\) 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 \(\|e_k\|:

\[
p_k \propto \left\langle \frac{\|e_k\|}{n_k} \sum r.e_k \left( r, w_{\text{sum}} / r.M + \epsilon \right) \|e_k\| \right\rangle.\]

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 \(p^B\) and \(p_0\) but require nontrivial precomputation. Instead, our
**ALGORITHM 6:** Our streaming RIS with spatiotemporal reuse for secondary boundary paths

```
1 EstSecondaryBound(dJL, prevEdgeReservoirs)
2 begin
3   (dJL)_bnd ← 0;
4   /* Sampling boundary paths (non-differentiable) */
5      updateEdgeReservoirs(prevEdgeReservoirs);
6   paths ← {}; edgeReservoirs ← {};
7   for i = 1 to N_bnd do
8      Sample p from an edge with pdf p(p);
9      /* Generate initial candidates */
10     Generate reservoir r using stream RIS; // §5.2
11     r.p ← p;
12     /* Spatiotemporal reuse */
13     Q ← PickNeighbors(prevEdgeReservoirs, r); // §5.2
14     r ← SpatiotemporalReuse(r, Q); // Alg. 3
15     /* Complete the path sample */
16     p0 ← r.y; p1 ← rayIntersect(p0, p0, p);
17     p1 ← p0; // Used by Alg. 8
18     /* Store sampled path */
19     Let q be the index of pixel intersecting P1, P2;
20     paths[i].pid ← q; // Record pixel index
21     paths[i].vtx ← (p0, p1, p2);
22     /* Record the detached factor of Eq. (26) */
23     pdf ← p(p) N_2/r.W;
24     paths[i].vec ← (bnd(p, r, y, p2) n_a(p));
25     /* Store reservoir */
26     edgeReservoirs[i] ← r;
27 end
28 /* Gradient computation (differentiable) */
29 for i = 1 to N_bnd do
30      q, p, r ← paths[i].pid, paths[i].vtx, paths[i].vec;
31      Re-compute p0 using differentiable ray intersection;
32      I ← detach(v) · p0; // Eq. (26)
33     /* Back-propagate gradients */
34     (dJL)_bnd ← (dJL)_bnd + backward(I);
35   return (dJL)_bnd, edgeReservoirs;
```

Edge importance sampling is a by-product of ReSTIR sample reuse and is orthogonal to these methods.

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 them.

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 P1(0) and P1(1) may return different reservoirs Q1(0) and Q1(1) (Line 9, Algorithm 4). This causes the merged reservoirs returned by SpatiotemporalReuse (Line 10) to become less correlated.

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 r(t) be a reservoir generated using streaming RIS (Line 9) for the t-th correlated path, and

\[ Q(t) = \text{PickNeighbors} \left( \text{prevReservoirs}, r(t) \right). \]  

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:

\[ r(t) \leftarrow \text{SpatiotemporalReuse} \left( r(t), Q^* \right), \]  

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

![Image of pixel-level antithetic sampling and correlation](image-url)

**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 p*, Yu et al. [2022] proposed generating three correlated rays through p*, p*+, and p*− 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.

![Image of correlation and reservoir selection](image-url)

**Fig. 5.** Correlating reservoirs: (a) Independent spatiotemporal reuse at antithetic shading points p and p* may reuse different reservoirs. Here, Q = \{r1, r2\} will be used at p and Q* = \{r2, r3\} at 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 p is chosen, so both p and p* reuse Q = \{r1, r2\}. This better correlates antithetic paths, improving sampling efficiency.
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 \( \frac{d \mathcal{L}}{d \theta} \) (Line 6) and update scene parameters (Line 15) via simple gradient descent (i.e., \( \theta \leftarrow \theta - \lambda_t \frac{d \mathcal{L}}{d \theta} \) 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 \( \frac{d \mathcal{L}}{d \theta} \) 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 \( I_0 \) and one rendered image \( I \) 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., \( r_y \) 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 resvr\(^{(t-1)}\) and edgeResvr\(^{(t-1)}\) whose shading points are occluded to all camera views in the current mini-batch by “forwarding” them to resvr\(^{(t)}\) and edgeResvr\(^{(t)}\) (Lines 15–24) for future reuse.

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 Ryzen 9 5900X 12-core CPU, 32GB of RAM, and an NVIDIA RTX 3090 GPU.

All our inverse rendering results use unbiased gradient estimates except 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.

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], 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.

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.

<table>
<thead>
<tr>
<th>Scene</th>
<th>Trg.</th>
<th>Bat.</th>
<th>Param.</th>
<th>Iter.</th>
<th>DR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oxalis (Fig. 1)</td>
<td>1</td>
<td>1</td>
<td>183K</td>
<td>2304</td>
<td>0.45s</td>
</tr>
<tr>
<td>Tree (Fig. 10)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1024</td>
<td>0.9s</td>
</tr>
<tr>
<td>Painting (Fig. 11)</td>
<td>1</td>
<td>1</td>
<td>153K</td>
<td>2048</td>
<td>0.37s</td>
</tr>
<tr>
<td>Kitty (Fig. 12)</td>
<td>32</td>
<td>16</td>
<td>65K</td>
<td>128</td>
<td>14.1s</td>
</tr>
<tr>
<td>Octagon (Fig. 13)</td>
<td>17</td>
<td>17</td>
<td>30K</td>
<td>512</td>
<td>12.2s</td>
</tr>
<tr>
<td>World map (Fig. 14)</td>
<td>6</td>
<td>2</td>
<td>52K</td>
<td>1024</td>
<td>5.2s</td>
</tr>
</tbody>
</table>
Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR

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 \( L(I) = \| I - I_0 \|_2^2 \) with some fixed \( I_0 \), we compare estimates of gradients \( \partial_I L \) and \( \partial_\theta I \)—whose product gives the gradient \( \partial_\theta L \) of the loss \( L \). 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 \( \partial_I L \). 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 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 \( \partial_\theta L \) of rendering loss \( L \) is the product of \( \partial_I L \) (gradient of \( L \) with respect to forward render \( I \)) and \( \partial_\theta I \) (derivative of forward render \( I \) with respect to scene parameters \( \theta \)).

For inverse rendering, one can apply forward-rendering ReSTIR only to improve \( I \) (and its gradient \( \partial_I L \)) while leaving \( \partial_\theta I \) 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 + RIS uses RIS for estimating both \( I \) and \( \partial_\theta I \);
- Ours + RIS uses ReSTIR for estimating \( I \) and \( \partial_\theta I \); and
- Ours + Ours is our full method, using ReSTIR for both \( I \) and \( \partial_\theta I \).

Further, we use \( L(I) = \| I - I_0 \|_2^2 \) with some fixed \( I_0 \) for gradient \( \partial_I L \) (computed via automatic differentiation). As shown in Figures 9, both "Ours + RIS" and "Ours + Ours" use ReSTIR for forward

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 + Ours") allows the optimization to smoothly converge to the groundtruth while the other methods fail due to high-variance estimates of the gradient \( \partial_\theta L \).
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.

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.

Another major benefit for having low-variance forward rendering $I$ is to lower the bias of the gradient $\partial_L I$ when $\partial_L I$ is nonlinear with respect to $I$.
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 of the Kitty, causing over-blurring and color shifts in the reconstructions. (b) Our method, on the other hand, provides significantly more accurate results.

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 uses 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


Amortizing Samples in Physics-Based Inverse Rendering using ReSTIR


ACM Trans. Graph., Vol. 42, No. 6, Article 214. Publication date: December 2023.