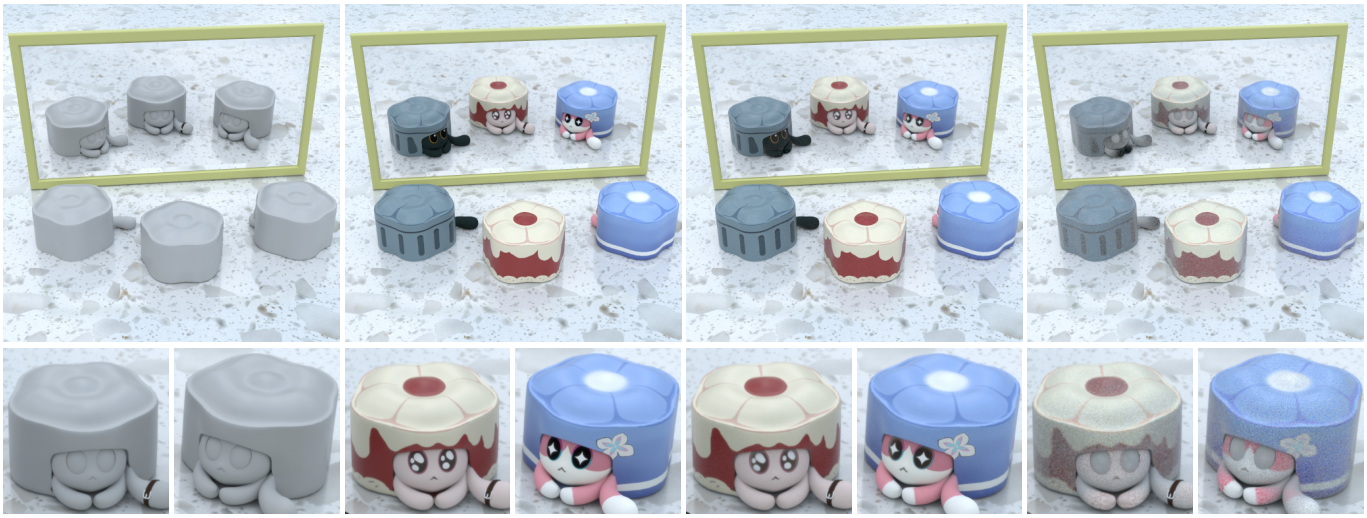
Image-space Adaptive Sampling for Fast Inverse Rendering

KAI YAN, University of California Irvine, USA
CHENG ZHANG, Reality Labs, Meta, USA
SÉBASTIEN SPEIERER, Reality Labs, Meta, Switzerland
GUANGYAN CAI, University of California Irvine, USA
YUFENG ZHU, Reality Labs, Meta, USA
ZHAO DONG, Reality Labs, Meta, USA
SHUANG ZHAO, University of California Irvine, USA



(a) **Initial** (b) **Target** (c) **Ours** (d) **Uniform**

Fig. 1. We introduce an image-space adaptive sampling technique as a form of pixel-level mini-batching for inverse rendering (that only renders a subset of pixels per iteration). Our method is loss- and variance-aware, enabling fast inverse rendering with low per-iteration computation budget. In this example, we jointly optimize the surface albedo of three Catcake objects. At equal time, our technique outperforms uniform mini-batching by allowing faster convergence. Inverse rendering animations generated using our technique and uniform mini-batching can be found in the supplemental material.

Inverse rendering is crucial for many scientific and engineering disciplines. Recent progress in differentiable rendering has led to efficient differentiation of the full image formation process with respect to scene parameters, enabling gradient-based optimization.

However, computational demands pose a significant challenge for differentiable rendering, particularly when rendering all pixels during inverse rendering from high-resolution/multi-view images. This computational cost leads to slow performance in each iteration of inverse rendering. Meanwhile, naively reducing the sampling budget by uniformly sampling pixels to render in each iteration can result in high gradient variance during inverse rendering, ultimately degrading overall performance.

Our goal is to accelerate inverse rendering by reducing the sampling budget without sacrificing overall performance. In this paper, we introduce a novel image-space adaptive sampling framework to accelerate inverse rendering by dynamically adjusting pixel sampling probabilities based on gradient variance and contribution to the loss function. Our approach efficiently handles high-resolution images and complex scenes, with faster convergence and improved performance compared to uniform sampling, making it a robust solution for efficient inverse rendering.

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

Additional Key Words and Phrases: Inverse rendering, differentiable rendering, gradient-based optimization

ACM Reference Format:
Kai Yan, Cheng Zhang, Sébastien Speierer, Guangyan Cai, Yufeng Zhu, Zhao Dong, and Shuang Zhao. 2025. Image-space Adaptive Sampling for Fast Inverse Rendering. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference*

Authors' Contact Information: Kai Yan, University of California Irvine, Irvine, USA, kyan8@uci.edu; Cheng Zhang, Reality Labs, Meta, Redmond, USA, zhangchengee@gmail.com; Sébastien Speierer, Reality Labs, Meta, Zürich, Switzerland, sebastieneps@gmail.com; Guangyan Cai, University of California Irvine, Irvine, USA, gcai3@uci.edu; Yufeng Zhu, Reality Labs, Meta, Redmond, USA, yufengzhu@meta.com; Zhao Dong, Reality Labs, Meta, Redmond, USA, zhaodong@meta.com; Shuang Zhao, University of California Irvine, Irvine, USA, shz@ics.uci.edu.

This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGGRAPH Conference Papers '25, Vancouver, BC, Canada
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1540-2/25/08
https://doi.org/10.1145/3721238.3730627

Papers '25), August 10–14, 2025, Vancouver, BC, Canada. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3721238.3730627

1 Introduction

Physics-based forward rendering concerns with synthesizing physically accurate images of 3D scenes. Decades of research efforts on this topic have led to forward rendering techniques capable of efficiently reproducing complex light transport effects including environmental lighting, soft shadows, and interreflection.

On the contrary, inverse rendering takes input images of a scene and aims to estimate scene parameters, including material properties and object geometries. A widely adopted solution known as *analysis by synthesis* formulates inverse rendering as an optimization problem that seeks for scene parameters that minimize the difference (quantified by some loss) between input images and renderings of the scene. Recent advances in differentiable rendering [Li et al. 2018; Zhang et al. 2020; Bangaru et al. 2020] have enabled the evaluation of forward-rendering derivatives with respect to scene parameters, allowing inverse-rendering optimizations to be solved using gradient-based methods such as stochastic gradient descent (SGD) and Adam [Kingma and Ba 2014].

Many, if not most, practical inverse-rendering configurations use many (e.g., multi-view) input images. Unfortunately, rendering all the corresponding images for every iteration of the optimization process can be can be prohibitively expensive in terms of both compute time and VRAM consumption. To address this problem, pixel-level *mini-batching* is a widely adopted approach in inverse-rendering pipelines [Mildenhall et al. 2020; Yariv et al. 2020; Müller et al. 2022; Nimier-David et al. 2022]: at each iteration, the loss is estimated using only a randomly selected subset of images or pixels.

However, the efficient selection of pixels for mini-batching has received limited attention. Most existing inverse-rendering pipelines simply adopt uniform sampling when selecting images or pixels. Although this simple strategy is unbiased and works adequately in many cases, as we will demonstrate later in this paper, doing so can lead to highly noisy estimates of loss gradients that significantly reduce the convergence rate of inverse-rendering optimizations.

In this paper, we introduce a technique that performs adaptive pixel sampling for inverse rendering. Given a per-iteration sampling budget, our technique allocates samples on each pixel based on not only its contribution to the loss function but also the forward/differentiable rendering variance. By offering efficient and low-variance gradient estimates, our method enables fast inverse rendering using small batch sizes without changing the underlying primal/differentiable rendering procedures.

Technically, our paper makes the following contributions.

- We introduce a technique that adaptively samples pixels to minimize the variance of loss gradient estimates (§3).
- We discuss how the pixel sampling technique can be efficiently implemented and integrated into complete inverse-rendering pipelines (§4).

We demonstrate in §5 the effectiveness of our technique using several synthetic inverse-rendering examples, including SVBRDF reconstruction, shape optimization, and inverse volume rendering.

2 Related Work

Adaptive sampling for rendering. Adaptive sampling has been extensively explored to improve the quality of Monte Carlo rendering in traditional rendering by allocating samples strategically rather than randomly. This area has been the focus of numerous previous works [Ramamoorthi et al. 2007; Overbeck et al. 2009; Rousselle et al. 2012; Zwicker et al. 2015; Hasselgren et al. 2020; Salehi et al. 2022; Firmino et al. 2023]. However, these works primarily focus on enhancing the quality of forward rendering. In contrast, we aim to utilize image-space adaptive sampling to improve the performance of inverse rendering.

Variance-aware forward rendering. Variance reduction is a long-standing topic in Monte Carlo forward rendering. Recent methods leverage variance-aware method to enhance various aspects including multiple importance sampling [Grittmann et al. 2019] and path guiding [Rath et al. 2020]. Our technique, a variance-aware method for enhancing inverse rendering, is largely orthogonal to these methods.

Differentiable rendering. Physics-based differentiable rendering has made significant progress in recent years, enabling gradient-based optimization in realistic scenes. A primary challenge in the development of general-purpose differentiable rendering techniques has been differentiating with respect to scene geometry, which typically requires calculating additional boundary integrals. To address this, multiple methods [Li et al. 2018; Zhang et al. 2020; Yan et al. 2022; Zhang et al. 2023; Xu et al. 2023] have been developed to directly sample discontinuity boundaries. Another class of methods utilize reparameterizes boundary integrals to avoid explicit handling of discontinuity boundaries altogether [Loubet et al. 2019; Bangaru et al. 2020]. These techniques simplify the differentiation process while maintaining accuracy in gradient computation. Our technique is orthogonal to these methods and can benefit inverse-rendering optimizations using the latter class.

Path sampling methods for differentiable rendering also been curial recently. Several approaches have been proposed to allow differentiable rendering scaling out to complex scenes with large numbers of parameters [Nimier-David et al. 2020; Vicini et al. 2021]. [Zhang et al. 2021; Su and Gkioulekas 2024; Belhe et al. 2024] propose differential and antithetic BRDF sampling methods for estimating BRDF parameter derivatives.

Variance-aware differentiable rendering was recently introduced in [Yan et al. 2024]. This technique enables differentiating not only the rendered image with respect to scene parameters but also its variance with respect to sampling probabilities. Our technique is orthogonal to this work, focusing on the estimation of the *variance of derivatives* for adaptive sampling rather than the *derivative of variance*.

Inverse rendering. Inverse rendering aims to estimate the physical attributes of a scene, such as material, geometry, lighting, and volume, from image(s). These pipelines recover scene parameters by minimizing a loss function, often leveraging a large batch of input photos for reconstruction. Neural-based methods [Mildenhall et al. 2020; Wang et al. 2021; Zhang et al. 2022; Kerbl et al. 2023] have gained popularity in recent years. In addition, recent advances in

differentiable rendering have driven the development of physics-based inverse rendering pipelines, as demonstrated by [Munkberg et al. 2022; Cai et al. 2022; Hasselgren et al. 2022; Sun et al. 2023]. However, these advances present a challenge: when processing high-resolution images or large image sets, small batch sizes can result in significant variance in gradient estimates.

Our key insight is that computing every pixel during inverse rendering is unnecessary; by strategically allocating samples, we can effectively reduce rendering costs and enhance overall performance.

3 Image-Space Adaptive Sampling

Consider a typical inverse rendering problem where a scene is controlled by some parameter $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ and $\boldsymbol{I}(\boldsymbol{\theta}) \in \mathbb{R}^{m_{\boldsymbol{I}}}$ be rendering(s) of the scene comprising $m_{\boldsymbol{I}}$ pixels $I_1(\boldsymbol{\theta}), \ldots, I_{m_{\boldsymbol{I}}}(\boldsymbol{\theta})$. The inverse rendering problem then aims at finding the optimal parameter $\boldsymbol{\theta}^*$ minimizing some (differentiable) loss $\mathcal{L}(\boldsymbol{I}; \boldsymbol{I}^{\text{target}})$ that measures the difference between the rendering $\boldsymbol{I}$ and some fixed target $\boldsymbol{I}^{\text{target}}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}\left(\boldsymbol{I}(\boldsymbol{\theta});\ \boldsymbol{I}^{\text{target}}\right). \tag{1}$$

In practice, the inverse-rendering optimization (1) is usually solved using gradient-based methods. To this end, a key ingredient is the derivative $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ of the loss $\mathcal{L}$ with respect to the parameter $\boldsymbol{\theta}$.

According to the chain rule, it holds that

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \underbrace{\frac{\partial \mathcal{L}}{\partial \boldsymbol{I}}}_{=:\,\partial_{\boldsymbol{I}}\mathcal{L}} \frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}. \tag{2}$$

Let $(\partial_{\boldsymbol{I}}\mathcal{L})_j \in \mathbb{R}$ denote the j-th component of $\partial_{\boldsymbol{I}}\mathcal{L}$. Then, Eq. (2) can be rewritten as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \sum_{j=1}^{m_{\boldsymbol{I}}} (\partial_{\boldsymbol{I}}\mathcal{L})_j \frac{\mathrm{d}I_j}{\mathrm{d}\boldsymbol{\theta}}. \tag{3}$$

Since estimating the derivative $\mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta}$ requires *differentiable rendering*, directly computing the sum over all pixels in Eq. (3) is computationally intensive. Replacing the sum with a Monte Carlo estimation (i.e., via mini-batching) can reduce this burden significantly.

Problem specification. We consider the following single-sample estimator $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$ of the loss gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$:

$$\left\langle \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} \right\rangle = \frac{\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle}{p_j} \left\langle \frac{\mathrm{d}I_j}{\mathrm{d}\boldsymbol{\theta}} \right\rangle, \tag{4}$$

where:

- the pixel index j is drawn randomly from $\{1, 2, \ldots, m_{\boldsymbol{I}}\}$ with the probability mass p_j;
- $\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle$ and $\langle \mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta} \rangle$ denote, respectively, Monte Carlo estimators of $(\partial_{\boldsymbol{I}}\mathcal{L})_j$ and $\mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta}$ that we assume are *given*, *independent*, and *unbiased*.

The objective of this section is to derive the probabilities masses $(p_j : j = 1, 2, \ldots, m_{\boldsymbol{I}})$ that minimize the variance of the estimator $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$ given by Eq. (4). In what follows, we revisit the existing solutions before introducing our technique in §3.1 and §3.2.

Previous methods. Conventional inverse-rendering methods have relied mostly on *uniform batching* that sets $p_j \equiv 1/m_{\boldsymbol{I}}$ for all j. Recently, Su and Gkioulekas [2024] have proposed to set

$$p_j \propto |(\partial_{\boldsymbol{I}}\mathcal{L})_j|. \tag{5}$$

Unfortunately, as we will demonstrate in §5, both uniform sampling and Eq. (5) can cause the gradient estimate $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$ in Eq. (4) to suffer from high variance when the estimators $\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle$ and $\langle \mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta} \rangle$ have greatly varying means and/or variances for different pixels j.

3.1 Scalar Case

To derive our technique, we first consider a simple case when the scene parameter θ is a scalar (i.e., $m_{\boldsymbol{\theta}} = 1$). We aim to minimize the variance of the estimator $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\theta \rangle$ expressed in Eq. (4). Since

$$\mathbb{V}\left[\left\langle \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \right\rangle\right] = \mathbb{E}\left[\left\langle \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \right\rangle^2\right] - \mathbb{E}^2\left[\left\langle \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \right\rangle\right], \tag{6}$$

and $\mathbb{E}[\langle \mathrm{d}\mathcal{L}/\mathrm{d}\theta \rangle] = \mathrm{d}\mathcal{L}/\mathrm{d}\theta$ is constant (as long as the estimator is unbiased), minimizing the variance $\mathbb{V}[\langle \mathrm{d}\mathcal{L}/\mathrm{d}\theta \rangle]$ amounts to minimizing the second moment $\mathbb{E}[\langle \mathrm{d}\mathcal{L}/\mathrm{d}\theta \rangle^2]$.

We recall that the estimators $\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle$ and $\langle \mathrm{d}I_j/\mathrm{d}\theta \rangle$ are assumed to be unbiased and independent. It follows that

$$\mathbb{E}\left[\left\langle \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\theta} \right\rangle^2\right] = \sum_{j=1}^{m_{\boldsymbol{I}}} \frac{1}{p_j} \mathbb{E}\left[\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle^2\right] \mathbb{E}\left[\left\langle \frac{\mathrm{d}I_j}{\mathrm{d}\theta} \right\rangle^2\right]. \tag{7}$$

It can be shown using Lagrange multiplier (see the textbook by Boyd and Vandenberghe [2004, §5.1.1]) that Eq. (7) has a global minimizer

$$p_j \propto \sqrt{p_{j1}\, p_{j2}}, \quad \text{where} \quad \begin{aligned} p_{j1} &:= \mathbb{E}[\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle^2], \\ p_{j2} &:= \mathbb{E}[\langle \mathrm{d}I_j/\mathrm{d}\theta \rangle^2], \end{aligned} \tag{8}$$

for $j = 1, 2, \ldots, m_{\boldsymbol{I}}$.

Discussion. Eq. (8) is a generalization of sampling densities used by previous inverse-rendering methods. Specifically, assuming $\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle$ to have zero variance (i.e., $(\partial_{\boldsymbol{I}}\mathcal{L})_j$ is known), it holds that $\mathbb{E}[\langle (\partial_{\boldsymbol{I}}\mathcal{L})_j \rangle^2] \equiv (\partial_{\boldsymbol{I}}\mathcal{L}_j)^2$, and Eq. (8) becomes

$$p_j \propto |(\partial_{\boldsymbol{I}}\mathcal{L})_j| \sqrt{\mathbb{E}\left[\left\langle \frac{\mathrm{d}I_j}{\mathrm{d}\theta} \right\rangle^2\right]}. \tag{9}$$

This relation further reduces to Eq. (5) when the second moments $\mathbb{E}[\langle \mathrm{d}I_j/\mathrm{d}\theta \rangle^2]$ remain constant for all j. However, these assumptions rarely hold in practice, reducing the effectiveness of the simple approach expressed in Eq. (5)—which we will demonstrate in §5.

3.2 General Case

We now consider the general case when the scene parameter $\boldsymbol{\theta} := (\theta_1, \theta_2, \ldots, \theta_{m_{\boldsymbol{\theta}}})$ is a vector. In this case, for each pixel j, the derivative $\mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta}$ is also a vector with its k-th component being the partial derivative $\partial I_j/\partial\theta_k$.

In this case, we consider the problem of minimizing the sum of per-parameter variances:

$$\sum_{k=1}^{m_{\boldsymbol{\theta}}} \mathbb{V}\left[\left\langle \frac{\partial \mathcal{L}}{\partial \theta_k} \right\rangle\right], \tag{10}$$

which is essentially the trace of the covariance matrix of the (vector-valued) estimate $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$.

Similar to the scalar case discussed in §3.1, minimizing Eq. (10) boils down to minimizing the sum of per-parameter second moments:

$$\sum_{k=1}^{m_{\boldsymbol{\theta}}} \mathbb{E}\left[\left\langle \frac{\partial \mathcal{L}}{\partial \theta_k} \right\rangle^2\right] = \sum_{j=1}^{m_{\boldsymbol{I}}} \frac{\mathbb{E}\left[\langle (\partial_{\boldsymbol{I}} \mathcal{L})_j \rangle^2\right]}{p_j} \sum_{k=1}^{m_{\boldsymbol{\theta}}} \mathbb{E}\left[\left\langle \frac{\partial I_j}{\partial \theta_k} \right\rangle^2\right], \tag{11}$$

with the global minimizer

$$p_j \propto \sqrt{p_{j1}\, p_{j2}}, \quad \text{where} \quad \begin{aligned} p_{j1} &:= \mathbb{E}[\langle (\partial_{\boldsymbol{I}} \mathcal{L})_j \rangle^2], \\ p_{j2} &:= \sum_{k=1}^{m_{\boldsymbol{\theta}}} \mathbb{E}\left[\left\langle \frac{\partial I_j}{\partial \theta_k} \right\rangle^2\right]. \end{aligned} \tag{12}$$

Monte Carlo estimation. We compute the sampling probability p_j given by Eq. (12) using a Monte Carlo process.

Specifically, to evaluate the first component p_{j1}, we first obtain the primal renderings $\boldsymbol{I}$ and compute the loss $\mathcal{L}$ and the derivative $\partial_{\boldsymbol{I}} \mathcal{L}$ using automatic differentiation. Then, we obtain p_{j1} by taking the j-th element of component-wise squared $\partial_{\boldsymbol{I}} \mathcal{L}$.

We now focus on estimating the remaining component p_{j2} given by the sum of second moments $\mathbb{E}[\langle \partial I_j/\partial \theta_k \rangle^2]$. Let f_j denote the measurement contribution function for pixel j. We assume that the provided differential estimators $\langle \partial I_j/\partial \theta_k \rangle$ (for all $k = 1, 2, \ldots, m_{\boldsymbol{\theta}}$) share the same path sampling strategy. Precisely,

$$\left\langle \frac{\partial I_j}{\partial \theta_k} \right\rangle = \frac{1}{\mathrm{pdf}_j(\bar{\boldsymbol{x}})} \frac{\partial f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}{\partial \theta_k}, \tag{13}$$

where the light path $\bar{\boldsymbol{x}}$ is drawn randomly with the probability density pdf_j. We omit the potential dependency of pdf_j on the scene parameter $\boldsymbol{\theta}$ for brevity.

Then, as demonstrated recently by Yan et al. [2024], the second moment $\mathbb{E}[\langle \partial I_j/\partial \theta_k \rangle^2]$ can be expressed as a path integral

$$\mathbb{E}\left[\left\langle \frac{\partial I_j}{\partial \theta_k} \right\rangle^2\right] = \int_\Omega \frac{1}{\mathrm{pdf}_j(\bar{\boldsymbol{x}})} \left(\frac{\partial f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}{\partial \theta_k}\right)^2 \mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{14}$$

where Ω denotes the path space with the Lebesgue measure μ. Therefore,

$$p_{j2} = \mathbb{E}\left[\sum_{k=1}^{m_{\boldsymbol{\theta}}} \left\langle \frac{\partial I_j}{\partial \theta_k} \right\rangle^2\right] = \int_\Omega \frac{1}{\mathrm{pdf}_j(\bar{\boldsymbol{x}})} \sum_{k=1}^{m_{\boldsymbol{\theta}}} \left(\frac{\partial f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}{\partial \theta_k}\right)^2 \mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{15}$$

which we estimate using the estimator

$$\langle p_{j2} \rangle = \frac{1}{\mathrm{pdf}_j^2(\bar{\boldsymbol{X}})} \sum_{k=1}^{m_{\boldsymbol{\theta}}} \left(\frac{\partial f_j(\bar{\boldsymbol{X}}; \boldsymbol{\theta})}{\partial \theta_k}\right)^2, \tag{16}$$

where $\bar{\boldsymbol{X}}$ is a light path sampled with the probability density pdf_j. We will discuss the efficient computation of Eq. (16) in §4.3.

Discussion. Computing the probabilities p_j in Eq. (12) requires performing primal and differentiable renderings of all pixels of the images $\boldsymbol{I}$, defeating the purpose of mini-batching. Fortunately, since these probabilities are only used for sampling pixels, we can compute them in an approximated fashion without sacrificing the unbiasedness of the gradient estimate $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$. We will provide more details on this in §4.1.

ALGORITHM 1: Estimating loss gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ using our method

```
1  EstimateLossGradient()
   /* Stage 1                                                    */
2  if p_j needs to be updated then
3      Render I using low resolution and sample count;
4      Compute the loss L and gradient ∂_I L using AD;
5      p_{j1} = (∂_I L) ** 2 ;              // Component-wise square
6      Compute p_{j2} using our custom AD process ;          // §4.3
7      Upsample and denoise p_{j1} and p_{j2};
8      p_j = (p_{j1} * p_{j2}) ** (1/2) ;  // Component-wise operations
9      Normalize p_j;
   /* Stage 2                                                    */
10 J = SamplePixelIndices(p_j);
11 ∂_θ L = 0;
12 for each j ∈ J do
13     Estimate I_j using (forward) path tracing and compute (∂_I L)_j;
14     Sample a camera_ray through pixel j;
15     ∂_θ L += PathReplay(camera_ray, I_j, (∂_I L)_j);
16 end
17 return ∂_θ L;
```

4 Inverse Rendering Using Image-Space Adaptive Sampling

We now explain how our sampling scheme described in §3 can be integrated to allow fast inverse rendering. Specifically, we will detail how loss gradients $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ can be estimated using our adaptive sampling in §4.1, introduce an optional step to better handle objects with greatly varying levels of details in §4.2, and discuss efficient automatic differentiation in §4.3.

4.1 Estimating Loss Gradients

In each iteration of an inverse-rendering optimization, we estimate the loss gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ in two stages, as outlined in Algorithm 1. In the first stage, the pixel sampling probabilities ($p_j : j = 1, 2, \ldots, m_{\boldsymbol{I}}$) are calculated (in an approximated fashion) based on Eq. (12).

In the second stage, we draw a set of pixel indices based on the computed probabilities. For each drawn index j, we first trace a light paths through pixel j to obtain an estimate $\langle (\partial_{\boldsymbol{I}} \mathcal{L})_j \rangle$ of the gradient $\partial_{\boldsymbol{I}} \mathcal{L}_j$. Then, we trace another path through this pixel to obtain $\langle \mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta} \rangle$. In the following, we provide more details for both stages.

Stage 1. A challenge for this stage, as discussed at the end of §3.2, is to compute the probabilities p_j efficiently since evaluating Eq. (12) exactly requires performing differentiable rendering for all pixels, defeating the purpose of adaptive sampling.

To address this problem, we make the key observation that, since the probabilities p_j is only used for sampling pixel indices in Stage 2 (i.e., Line 10 of Algorithm 1), we can compute them approximatedly without breaking unbiasedness (so long as $p_j > 0$ for all j).

In practice, we leverage *upsampling* and *denoising* to greatly reduce the computational overhead for computing p_j. Specifically, we first estimate the components p_{j1} and p_{j2} using low resolution and sample count. Since both of these terms are nonnegative, we treat the results as images and use over-the-shelf tools (such as the OptiX

denoiser [NVIDIA 2024]) and DLSS [NVIDIA 2024] to denoise and upsample them (see Figure 2 for an example).

Additionally, during an inverse rendering optimization, we only recompute p_j every few (e.g., 20) iterations, further reducing the amortized overhead per iteration.

Stage 2. Given the computed probabilities $(p_j : j = 1, 2, \ldots, m_{\boldsymbol{I}})$, the second stage of our pipeline starts with drawing pixel indices based on these probabilities. For each drawn index j, we trace a camera ray through the corresponding pixel and construct a full path using unidirectional path tracing. This path is used to estimate the primal pixel value I_j as well as the derivative $(\partial_{\boldsymbol{I}} \mathcal{L})_j$.

Then, we trace another path independently through the same pixel j. This path is used to compute the derivative $\mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta}$ and, in turn, update the final gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$. To implement this efficiently, we incorporate the path replay backpropagation (PRB) technique [Vicini et al. 2021].

Discussion. In Stage 2, we assume that the gradient $(\partial_{\boldsymbol{I}} \mathcal{L})_j$ of the loss $\mathcal{L}$ can be estimated solely based on I_j since only part of the image $\boldsymbol{I}$ is estimated unbiasedly. In practice, this is the case for many commonly adopted losses. When $\mathcal{L}$ is the L2 loss, for example, it holds that

$$(\partial_{\boldsymbol{I}} \mathcal{L})_j = 2 \left(I_j - I_j^{\text{target}} \right), \tag{17}$$

where I_j^{target} denotes the j-th pixel of the target image $\boldsymbol{I}$. However, this assumption can be violated for more sophisticated losses. We consider efficient estimation of $(\partial_{\boldsymbol{I}} \mathcal{L})_j$ for such losses future work.

Further, since we use the Monte Carlo rendering $\langle I_j \rangle$ to estimate the gradient $\langle (\partial_{\boldsymbol{I}} \mathcal{L})_j \rangle$, the latter is only unbiased if the relation between I_j and $(\partial_{\boldsymbol{I}} \mathcal{L})_j$ is linear—which holds when the loss is $L2$, as shown in Eq. (17), but is not generally the case for other losses. Fortunately, this is hardly a problem in practice since having biased p_j does not affect the unbiasedness of the final gradient $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$.

4.2 Multi-Resolution Modeling

For scenes containing objects with greatly varying levels of details (e.g., a mixture of textured and untextured objects), the effectiveness of our adaptive sampling can slightly degrade as our formulation in Eq. (10) does not fully capture the correlation between pixels due to being controlled by the same set of parameters. In practice, this can cause our method to allocate more samples for low-detail (e.g., untextured) objects, slowing down the convergence of high-detail (e.g., textured) ones.

To address this problem, we introduce an optional step that computes the sampling probabilities $p_{j',d}$ at multiple downsampled levels $d = 1, 2, \ldots, D$ and combine the results to obtain the final p_j. Specifically, let $(p_{j',d} : j' = 1, 2, \ldots)$ be the sampling probabilities computed on a $(2^d \times 2^d)$-downsampled resolution. We then set the final pixel sampling probabilities as

$$p_j \coloneqq \sum_{d=0}^{D} 4^d \cdot p_{j_d,d}, \tag{18}$$

where j_d denotes the index of the $(2^d \times 2^d)$-downsampled pixel that contains the original pixel j (for $d = 0, 1, \ldots, D$).

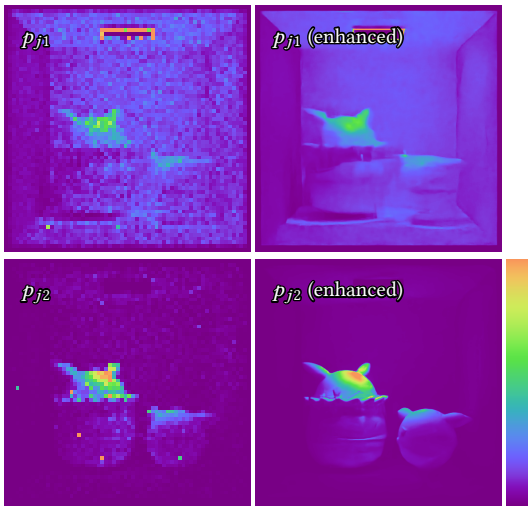


Fig. 2. We compute p_{j1} and p_{j2} defined in Eq. (12) approximately by first obtaining low-resolution estimates using low sample counts, and enhancing them by applying upsampling and denoising.

In this way, a pixel j' affected by many parameters at some high downsampling level d will likely receive some large sampling weight $p_{j',d}$. Then, this weight will be propagated to all the corresponding pixels j at the original resolution via Eq. (18), increasing the probabilities for them to be sampled.

In practice, we compute $p_{j',d}$ by executing Stage 1 of Algorithm 1 multiple times with varying (target) resolutions.

4.3 Custom Derivative Computation

First, we discuss the computation of the sum of partial derivatives

$$\sum_{k=1}^{m_{\boldsymbol{\theta}}} \frac{\partial f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}{\partial \theta_k}, \tag{19}$$

which neglects the squaring operation on the right-hand side of Eq. (15) for each pixel j.

Given a sampled light path $\bar{\boldsymbol{x}}$, a naïve approach is to: (i) acquire the derivative vector $\partial f_j(\bar{\boldsymbol{x}};\boldsymbol{\theta})/\partial \boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ by applying reverse-mode automatic differentiation (AD) to the measurement contribution $f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})$; and (ii) compute the sum of the derivative vector.

Unfortunately, this approach requires back-propagating gradients per pixel—which is usually impractical. Luckily, a special forward-mode AD function forward_to from Dr.Jit [Jakob et al. 2022] can compute this sum—which we explain in the following.

The function forward_to includes two stages: the first **backward enqueue** stage traverses the computation graph backwards from all f_j at once to find potential paths along which gradients can flow to. The second **forward traverse** stage performs a gradient propagation pass along all detected variables θ_k, which accumulates the gradient from each leaf node $\partial f_j(\bar{\boldsymbol{x}};\boldsymbol{\theta})/\partial \theta_k$. Dr.Jit shares the same computational graph on GPU for forward-mode and reverse-mode AD that captures operations for subsequent derivative propagation, which avoid forward propagation from each variables individually.

We now discuss the computation of the sum of squared partial derivatives

$$\sum_{k=1}^{m_{\boldsymbol{\theta}}} \left(\frac{\partial f_j(\bar{\boldsymbol{x}}; \boldsymbol{\theta})}{\partial \theta_k} \right)^2, \tag{20}$$

on the right-hand side of Eq. (15) for each pixel j. We create a custom forward_to function to compute the squared sum in Eq. (20). The process has the identical **backward enqueue** stage to traverse the computation graph. However, in the **forward traverse** stage, it accumulates the square gradient for each leaf node $\partial f_j(\bar{\boldsymbol{x}};\boldsymbol{\theta})/\partial\theta_k$ by squaring its value before the final accumulation in the function accum.

To further reduce the memory footprint and improve efficiency, we implement this method upon PRB [Vicini et al. 2021]: which performs AD for each light path and is fully compatible with our method.

5 Results

We implement our technique described in §3 and §4 on the GPU using the Dr.Jit numerical backend [Jakob et al. 2022]. In the following, we show ablation studies and additional inverse-rendering comparisons in §5.1 and §5.2, respectively.

Experiment configurations. To demonstrate the generality of our method, our experiments involve a wide range of inverse rendering problems, using the Adam optimizer [Kingma and Ba 2014], including SVBRDF reconstruction, shape optimization, and inverse volume rendering. Some of these problems also involve complex light transport effects.

We set the total sample budge (i.e., total number of light paths traced per iteration) to about 100K for all inverse rendering results. For multi-resolution modeling, we set number of depth $D = 4$ for all results. Lastly, as discussed in §4, we update the pixel sampling probabilities p_j every 20 iterations.

Equal-time comparisons. When comparing inverse-rendering results, we use identical losses, initializations, and input images as well as roughly equal execution time per iteration. The total optimization time of our experiments ranges from 10 seconds to 8 minutes (on a workstation with a RTX 4090 GPU).

5.1 Evaluation & Ablation

Variance evaluation. We conduct another experiment to directly evaluate the effectiveness of our adaptive sampling (§3) in reducing the variance of the gradient estimate $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$. As shown in Figure 3, this experiment uses a scene where three diffuse objects are placed in front of a mirror. The experiment simulates a single iteration of an inverse-rendering optimization where the roughness of the mirror is optimized. Given the pixel sampling probabilities p_j, we estimate the loss gradient $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$ at a fixed roughness value using Eq. (4) 1,024 times and examine the variance of individual estimation results. Compared with the simple scheme used by Su and Gkioulekas [2024], our method is capable of reducing the variance by over 3×.

Loss function. We demonstrate the robustness of our technique to different loss functions. Figure 4 shows a JumpyDumpy scene where a glossy textured object is illuminated by an environment map with a bright sun. We optimize the albedo of this object. The bright sun and glossy surface cause drastic changes in brightness across different pixels. For inverse-rendering optimizations driven by both L2 and relative L2 losses, our method outperforms uniform mini-batching.

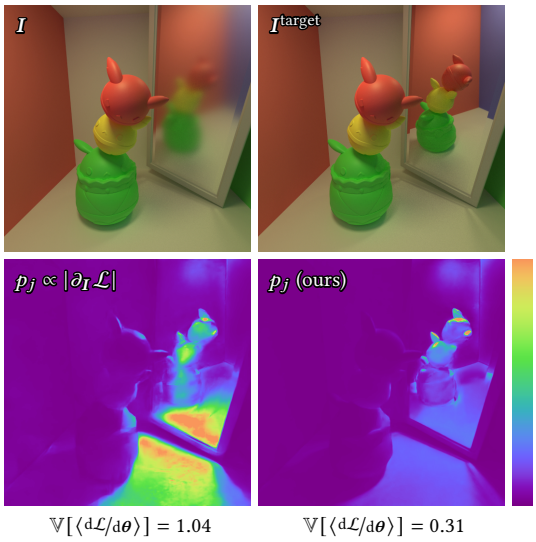


Fig. 3. **Variance comparison:** We directly compare the variance of loss gradient estimates $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$ using two pixel sampling probabilities. In this example, we use a single parameter θ that controls the roughness of the mirror. The variance numbers are acquired by repeatedly evaluating Eq. (4) and examining the statistics of the results.

Multi-resolution scheme. We demonstrate the effectiveness of our multi-resolution scheme described in §4.2 in Figure 5 that uses a Cornell box containing a texture-less trash can and a textured Catcake. Using one input image, we jointly optimize the albedo values for both objects. Without our multi-resolution scheme, the sampling probability p_j distributes samples relatively evenly on both objects, leading to slower convergence for the textured Catcake. In the contrary, without scheme, the probability focuses more on the textured object, leading to more uniform convergence for both objects.

Differential samplers. Our method is not limited to any specific sampling scheme for the estimation of pixel value derivatives $\langle \partial I_j/\partial\theta_k \rangle$. In Figure 6, we show an example that performs differential BSDF sampling [Belhe et al. 2024]. This example involves a Cornell box containing a glossy trash can with a spatially invariant BRDF. We optimize the surface roughness of the trash can. Using one input image and a small sample budget, the uniform-sampling with differential sampler method still stucks at the initial configuration due to high noise. Our method, on the other hand, allows the optimization to converge nicely.

5.2 Inverse-Rendering Comparisons

We now compare inverse-rendering results obtained under equal time. We adjust optimization parameters (e.g., learning rate) for the

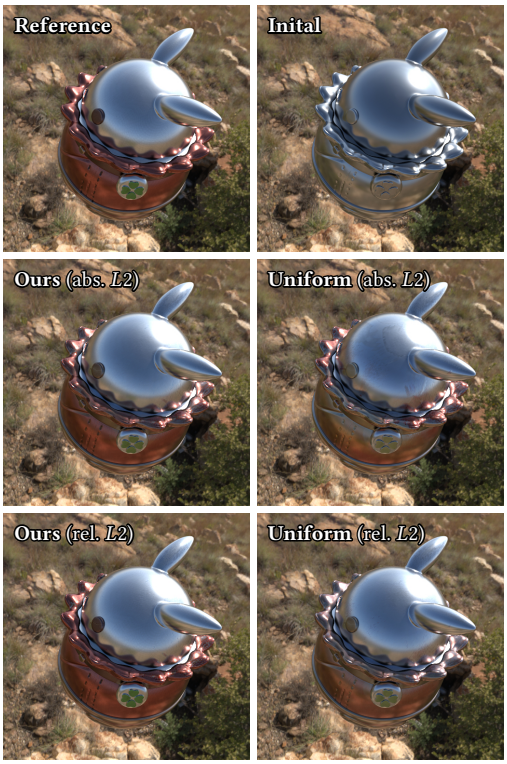


Fig. 4. **Loss functions:** Our technique is robust to the choice of loss functions. In this example, we compare inverse-rendering results generated using our technique and uniform mini-batching at equal time with absolute and relative $L2$ losses. Our method outperforms the baseline in both cases.



Fig. 5. **Multi-resolution modeling:** Our multi-resolution scheme introduced in §4.2 allows uniform convergence on objects with greatly varying levels of details. This example involves an untextured TrashCan and a textured Catcake, and we jointly optimizing the surface albedo of both objects. Without our multi-resolution scheme, the optimization allocates most samples on the trashcan, reducing the convergence rate of the Catcake.

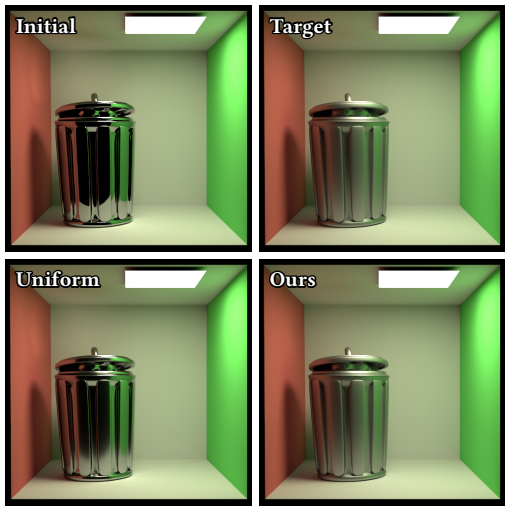


Fig. 6. **Differential BRDF sampling:** Our method makes no assumption on BRDF sampling used by the underlying primal and differentiable rendering processes. In this example, image deriatives $\mathrm{d}I_j/\mathrm{d}\boldsymbol{\theta}$ are estimated using the differential sampling method introduced by Belhe et al. [2024]. At equal time, our method outperforms uniform mini-batching by allowing significantly faster convergence.

baseline methods when needed to ensure that they perform as well as possible under the same time budget.

Material reconstruction. Figure 8 contains inverse-rendering results where surface or volume material properties are optimized. We use the principled BSDF [Burley et al. 2015] for all surface materials. We compare our method to the two baselines discussed in §3: one with constant p_j and the other with $p_j \propto |\partial_{\boldsymbol{I}} \mathcal{L}_j|$.

Each example uses one or more images with 4M pixels, and the optimizations use a budget of about 100K sample paths per iteration.

The Dodoco scene uses a similar setting as Figure 3, and we optimize the roughness of the mirror. The Pottery scene involves a glossy object under environmental lighting with bright sun light, and the spatially varying albedo is optimized using 16 input images. For both examples, our technique outperforms the baselines.

The Earth2Mars scene contains two spheres under environmental lighting. Initialized with the spheres having opposite textures as the target, we optimize the textures (specifying diffuse albedo) of these spheres using 16 input images. Since one of the two spheres resides inside a glass enclosure, the rendering variance of the two spheres differs greatly. Being the only variance-aware technique, our method allows faster convergence than both baseline methods.

The Smoke scene involves a volumetric smoke under environmental lighting. Using 16 input images, we optimize per-voxel density and color of the smoke. By allocating more samples to regions with higher variance, our method again offers the faster convergence than the baselines.

Additional comparisons. Lastly, we provide more inverse-rendering results in Figure 9 where we compare our method to the baseline

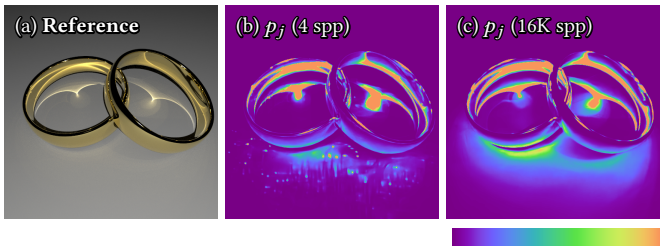


Fig. 7. **Failure case:** For scenes that cannot be efficiently rendered using unidirectional path tracing (a), our estimated sampling probability p_j can be unreliable (b)—unless using impractically high sample counts (c).

with constant p_j. All scenes used by this figure use environmental lighting. Earth uses one image with 16M pixels. Others use multiple images with 262K pixels per image. The optimizations use a budget of about 100K sample paths per iteration.

The Earth scene involves an earth-like object covered by a glass bell. We also observe the back of this object through a mirror. The DodocoA scene contains an object lit by a outdoor environment map. We optimize the spatially varying albedo of this object using 16 multi-view images. The Bowl scene contains a textured glossy bowl, and we optimize albedo using 32 images of the object. We initialize all these examples using a spatially invariant albedo of 0.5.

The Lego scene uses a NeRF-like setting with a lego bulldozer described as an emissive volume (with no scattering). Using 32 input images, we optimize per-voxel density and albedo for this object. In addition, we use a coarse-to-fine scheme that starts with a resolution of 8^3 and gradually increases to 256^3 (by doubling the resolution every 200 iterations).

The DodocoB and Kirby scenes are used for shape optimization. Starting with a sphere with 30K vertices, we optimize the shapes of the objects (described as triangle meshes) using 16 input images. For both examples, we use warped-area sampling [Bangaru et al. 2020] to obtain the shape derivatives and the large-step method [Nicolet et al. 2021] to update the object meshes.

In all cases, our technique outperforms the baseline thanks to low variance in the loss gradient $\langle \mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \rangle$.

6 Discussion and Conclusion

Limitations and future work. Our method assumes unidirectional path tracing. Therefore, for problems that require using adjoint or bidirectional methods (e.g., scenes with strong caustic effects), as shown in Figure. 7, our method would be inapplicable. Our technique performs adaptive sampling in the image space (i.e., for pixels) and currently works only when the underlying primal and differentiable processes use unidirectional path tracing. Generalizing our technique to suppose more sophisticated path sampling strategies such as path-space methods to sample boundary light paths [Zhang et al. 2020; Yan et al. 2022; Zhang et al. 2023] is an important topic for future research.

Also, as discussed in §4.1, developing efficient methods to compute $(\partial_{\boldsymbol{I}} \mathcal{L})_j$ for general losses such as LPIPS [Zhang et al. 2018] is worth exploring in the future.

Conclusion. We introduced an image-space adaptive sampling technique to perform pixel-level mini-batching for inverse rendering while minimizing the variance of loss gradient estimates. By considering both mean and variance of the underlying primal and differentiable rendering processes, our technique is a significant generalization of previous methods. Moreover, we discussed how this sampling technique can be integrated into practical inverse rendering pipelines, allowing smooth convergence with fast rendering per iteration. We demonstrated the effectiveness of our technique by comparing it with previous methods using several synthetic inverse rendering examples.

Acknowledgments

We would like to thank Qianhui Wu for artistic support. This work started when Kai Yan was an intern at Meta.[1] This project was partially funded by NSF grant 2239627.

References

Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. 2020. Unbiased Warped-Area Sampling for Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020).
Yash Belhe, Bing Xu, Sai Praveen Bangaru, Ravi Ramamoorthi, and Tzu-Mao Li. 2024. Importance Sampling BRDF Derivatives. *ACM Transactions on Graphics* (2024).
Stephen Boyd and Lieven Vandenberghe. 2004. *Convex Optimization.* Cambridge University Press.
Brent Burley, Danny Chan, Luca Fascione, Michal Iwanicki, Natty Hoffman, Wenzel Jakob, David Neubelt, Angelo Pesce, and Matt Pettineo. 2015. Physically Based Shading in Theory and Practice. In *ACM SIGGRAPH 2015 Courses.*
G. Cai, K. Yan, Z. Dong, I. Gkioulekas, and S. Zhao. 2022. Physics-Based Inverse Rendering Using Combined Implicit and Explicit Geometries. *Computer Graphics Forum* 41, 4 (July 2022).
Arthur Firmino, Jeppe Revall Frisvad, and Henrik Wann Jensen. 2023. Denoising-Aware Adaptive Sampling for Monte Carlo Ray Tracing. In *ACM SIGGRAPH 2023 Conference Proceedings.* Association for Computing Machinery, Article 32.
Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. 2019. Variance-aware multiple importance sampling. *ACM Trans. Graph.* 38, 6 (2019).
Jon Hasselgren, Nikolai Hofmann, and Jacob Munkberg. 2022. Shape, Light, and Material Decomposition from Images using Monte Carlo Rendering and Denoising. *arXiv:2206.03380* (2022).
J. Hasselgren, J. Munkberg, M. Salvi, A. Patney, and A. Lefohn. 2020. Neural Temporal Adaptive Sampling and Denoising. *Computer Graphics Forum* 39, 2 (2020).
Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. 2022. Dr.Jit: A Just-In-Time Compiler for Differentiable Rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)* 41, 4 (July 2022).
Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics* 42, 4 (July 2023).
Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).
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).
Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.* 38, 6 (2019).
Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV.*
Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. 2022. Instant Neural Graphics Primitives with a Multiresolution Hash Encoding. *ACM Trans. Graph.* 41, 4, Article 102 (July 2022).
Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. 2022. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*
Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 40, 6 (Dec. 2021).

[1] All assets are inspired creations by the author, Kai Yan [Yan 2024]. The assets are not products of Meta, nor are they endorsed by Meta.

Merlin Nimier-David, Thomas Müller, Alexander Keller, and Wenzel Jakob. 2022. Unbiased Inverse Volume Rendering with Differential Trackers. *ACM Trans. Graph.* 41, 4, Article 44 (July 2022).

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).

NVIDIA. 2024. NVIDIA DLSS 4 Introduces Multi Frame Generation & Enhancements For All DLSS Technologies.

NVIDIA. 2024. NVIDIA OptiX 8.1 SDK.

Ryan S. Overbeck, Craig Donner, and Ravi Ramamoorthi. 2009. Adaptive wavelet rendering. *ACM Trans. Graph.* 28, 5 (Dec. 2009).

Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. 2007. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph.* 26, 1 (Jan. 2007).

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Petr Vévoda, Philipp Slusallek, and Jaroslav Křivánek. 2020. Variance-aware path guiding. *ACM Trans. Graph.* 39, 4 (2020).

Fabrice Rousselle, Claude Knaus, and Matthias Zwicker. 2012. Adaptive rendering with non-local means filtering. *ACM Trans. Graph.* 31, 6, Article 195 (2012).

Farnood Salehi, Marco Manzi, Gerhard Roethlin, Romann Weber, Christopher Schroers, and Marios Papas. 2022. Deep Adaptive Sampling and Reconstruction Using Analytic Distributions. *ACM Trans. Graph.* 41, 6 (2022).

Tanli Su and Ioannis Gkioulekas. 2024. Path sampling methods for differentiable rendering. In *Eurographics Symposium on Rendering*.

Cheng Sun, Guangyan Cai, Zhengqin Li, Kai Yan, Cheng Zhang, Carl Marshall, Jia-Bin Huang, Shuang Zhao, and Zhao Dong. 2023. Neural-PBIR reconstruction of shape, material, and illumination. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2021. Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time. *Transactions on Graphics (Proceedings of SIGGRAPH)* 40, 4 (Aug. 2021).

Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. 2021. NeuS: Learning Neural Implicit Surfaces by Volume Rendering for Multi-view Reconstruction. *NeurIPS* (2021).

Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* 42, 6 (2023).

Kai Yan. 2024. Artistic support for rendering. https://yank.ai/.

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).

Kai Yan, Vincent Pegoraro, Marc Droske, Jiří Vorba, and Shuang Zhao. 2024. Differentiating Variance for Variance-Aware Inverse Rendering. In *ACM SIGGRAPH Asia 2024 Conference Proceedings*.

Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. 2020. Multiview Neural Surface Reconstruction by Disentangling Geometry and Appearance. *Advances in Neural Information Processing Systems* 33 (2020).

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

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

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

Yuanqing Zhang, Jiaming Sun, Xingyi He, Huan Fu, Rongfei Jia, and Xiaowei Zhou. 2022. Modeling Indirect Illumination for Inverse Rendering. In *CVPR*.

Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Trans. Graph.* 42, 6 (2023).

Matthias Zwicker, Wojciech Jarosz, Jaakko Lehtinen, B. Moon, Ravi Ramamoorthi, Fabrice Rousselle, Pradeep Sen, Cyril Soler, and Sung-eui Yoon. 2015. Recent Advances in Adaptive Sampling and Reconstruction for Monte Carlo Rendering. *Computer Graphics Forum* 34 (05 2015).

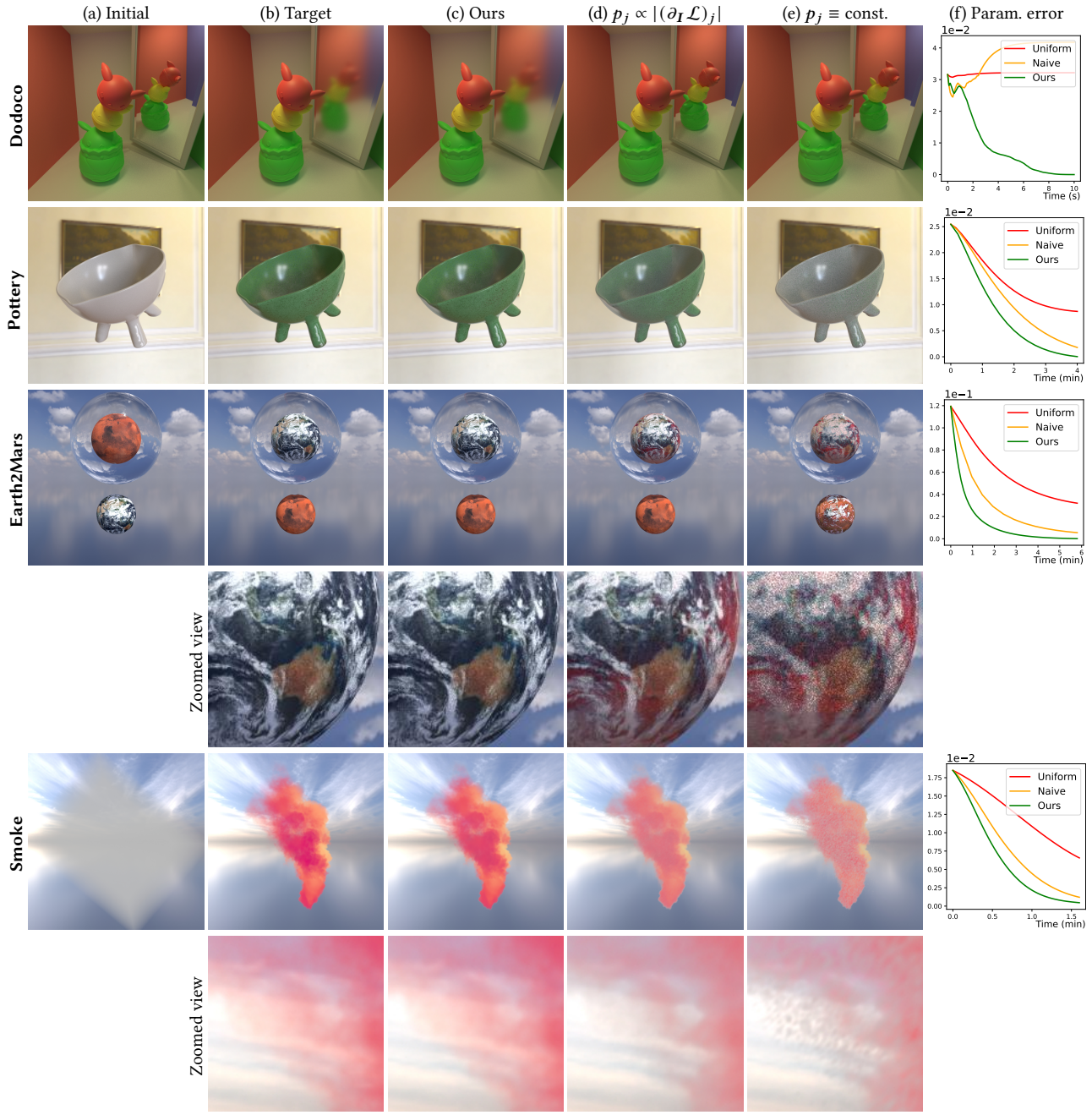


Fig. 8. **Material reconstruction:** We compare inverse-rendering results where object material properties are optimized. For each example, we show results obtained in equal optimization time using three pixel sampling strategies: (c) our technique (12); (d) previous method (5); and (e) uniform sampling. The parameter error (f) measures the difference (in $L2$) between the optimized parameters and the groundtruth. It is only used for evaluation (i.e., not for optimization).

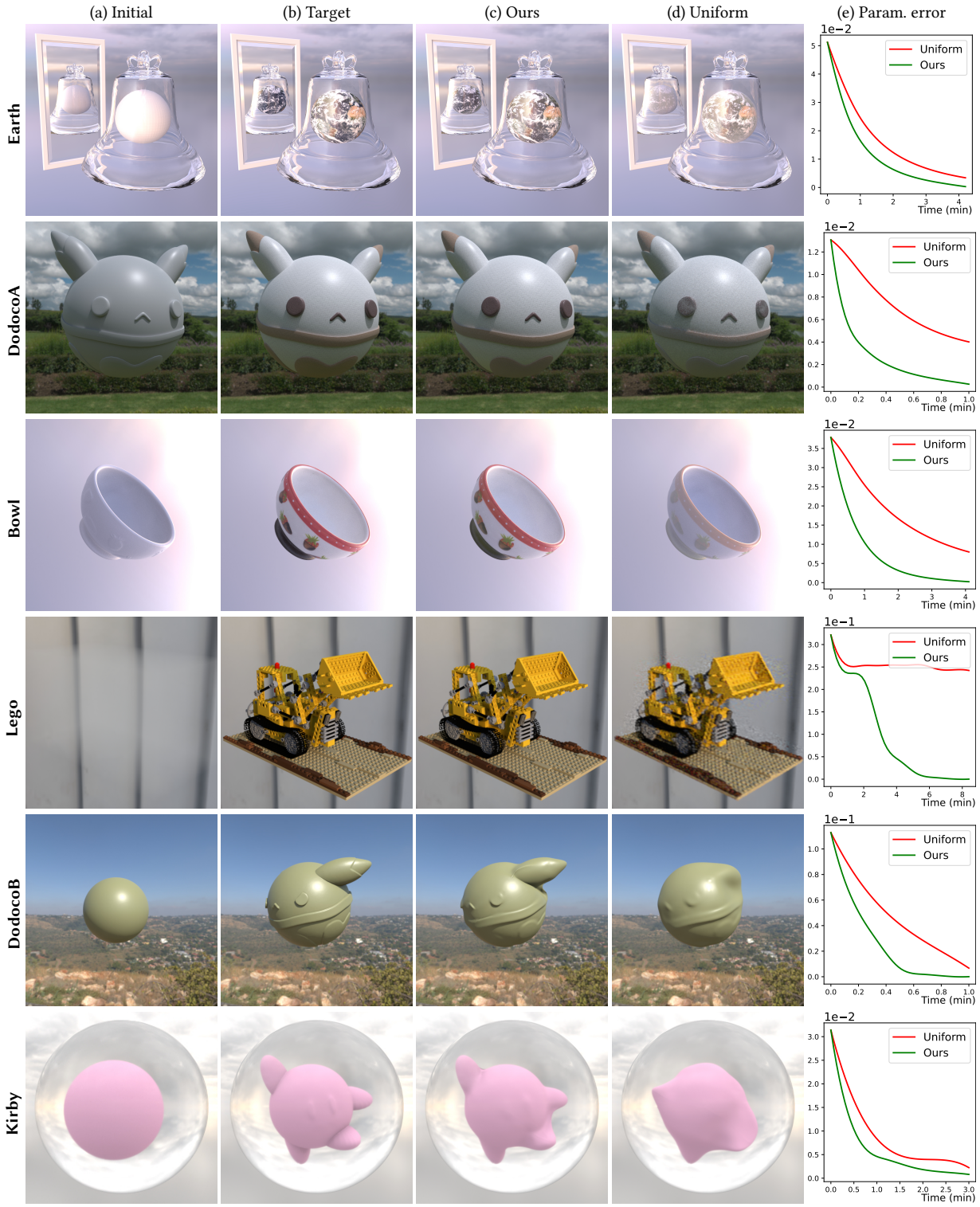


Fig. 9. **Additional inverse-rendering comparison:** We compare inverse-rendering results where material properties are optimized for DodocoA, Earth, and Bowl, and object geometries are optimized for Dodocob and Kirby. For each example, we show results obtained in equal optimization time using two pixel sampling strategies: (c) our technique (12); and (d) uniform sampling. The parameter error is only used for evaluation (i.e., not for optimization).