Markov-Chain Monte Carlo Sampling of Visibility Boundaries for Differentiable Rendering

ANONYMOUS AUTHOR(S)
SUBMISSION ID: 551

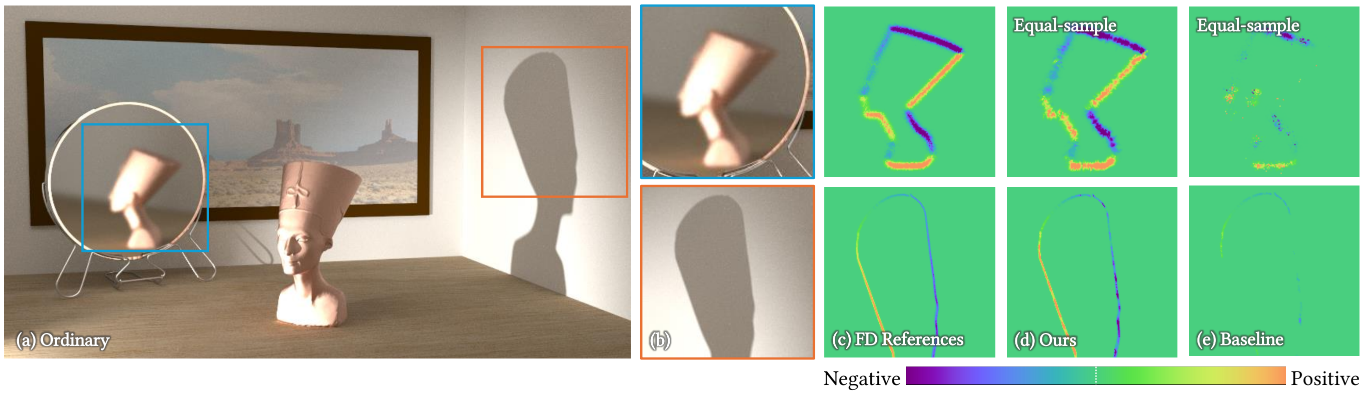

Fig. 1. We introduce a Markov-Chain-Monte-Carlo (MCMC) differentiable rendering method to sample boundary paths that are crucial for estimating derivatives with respect to object geometries. This example includes a highly tessellated Nefertiti model (**with 1 million vertices**) lit by one area lights from the bottom left corner (a), casting a shadow on the right wall, and observed both directly and indirectly through a mirror. We compute derivatives with respect to the translation of the Nefertiti model. The highly tessellated mesh and small area emitter make visibility boundary sampling extremely difficult—even with primary-sample-space guiding. At equal sample and lower render time, our technique (d) produces significantly cleaner derivative estimates than the state-of-the-art baseline by Zhang et al. [2023] (e).

Physics-based differentiable rendering requires estimating boundary path integrals emerging from the shift of discontinuities (e.g., visibility boundaries). Previously, although the mathematical formulation of boundary path integrals has been established, efficient and robust estimation of these integrals has remained challenging. Specifically, state-of-the-art boundary sampling methods all rely on primary-sample-space guiding precomputed using sophisticated data structures—whose performance tends to degrade for finely tessellated geometries.

In this paper, we address this problem by introducing a new Markov-Chain-Monte-Carlo (MCMC) method. At the core of our technique is a local perturbation step capable of efficiently exploring highly fragmented primary sample spaces via specifically designed jumping rules. We compare the performance of our technique with several state-of-the-art baselines using synthetic differentiable-rendering and inverse-rendering experiments.

1 INTRODUCTION

Differentiable rendering computes gradients of detector responses with respect to differential changes of a virtual scene. Being an active research topic in computer graphics, differentiable rendering is a key ingredient for integrating the rendering processes into probabilistic inference and machine learning pipelines, leading to applications in a wide array of areas including computer vision, computational imaging, and computational fabrication.

Recently, great progress has been made in physics-based differentiable rendering theory and algorithms [Li et al. 2018; Zhang et al. 2019, 2020; Bangaru et al. 2020; Xu et al. 2023]. These advances have enabled the capability of differentiating renderings with complex light-transport effects (e.g., interreflection) with respect to arbitrary scene parameters including those controlling global object geometry (e.g., the positions of mesh vertices). Mathematically, it has been demonstrated that physics-based differentiable rendering generally amounts to evaluating *interior* and *boundary* integrals.

The *interior* integrals for differentiable rendering share the same domain as those for forward rendering. Besides repurposing existing sampling strategies developed for forward rendering, several techniques specialized for differentiable rendering have been introduced recently. Utilizing new importance sampling strategies [Zeltner et al. 2021], antithetic sampling [Zhang et al. 2021], or spatiotemporal reuse [Chang et al. 2023; Wang et al. 2023], these techniques have advanced the efficiency for estimating the *interior* integrals.

The *boundary* integrals are unique to differentiable rendering and emerge from discontinuities (e.g., visibility boundaries) evolving with the differentiation parameter. Previously, these integrals are usually estimated in two ways—either *directly* [Li et al. 2018; Zhang et al. 2020; Yan et al. 2022] or after being *reparameterized* into *interior* integrals [Loubet et al. 2019; Bangaru et al. 2020; Xu et al. 2023].

Compared with the reparameterization-based methods, the direct ones have the benefit of being unbiased and, when importance sampled properly, more efficient. To this end, several guiding-based methods [Yan et al. 2022; Zhang et al. 2023] that operate under the primary-sample space have been introduced. Unfortunately, while although these techniques work adequately for scenes with relatively simple geometries, their performance tends to degrades for highly tessellated scenes—which can cause the primary-sample space to become extremely fragmented, overwhelming accelerate structures (e.g., octrees) used by the guiding-based methods.

In this paper, we develop a new technique for sampling of *boundary* paths. Instead of relying on guiding, our technique leverages Markov-Chain-Monte-Carlo (MCMC) for efficient exploration of the primary-sample space.

Concretely, our contributions include:

- Introducing MCMC to physics-based differentiable rendering;
- Devising new local perturbation strategies to enable robust exploration of highly fragmented primary-sample spaces.

We demonstrate the effectiveness of our technique using several differentiable and inverse rendering examples (Figures 7–9).

2 RELATED WORKS

Markov chain Monte Carlo (MCMC). MCMC techniques allow drawing (potentially high-dimensional) samples proportional to arbitrary nonnegative *target functions*, enabling various statistical inference tasks (e.g., [Brooks et al. 2011]). One of the most well known MCMC techniques is the Metropolis-Hastings algorithm [Hastings 1970; Metropolis et al. 1953]. The idea is to start from an initial sample, and mutate the sample using a *proposal* distribution. The algorithm then randomly accepts or rejects the sample based on the target function to create a Markov chain. The sample sequence forms a distribution that asymptotically converges to the target distribution.

State-of-the-art techniques for designing proposal distributions include *Hamiltonian Monte Carlo* (HMC) [Duane et al. 1987; Neal et al. 2011; Betancourt 2017] and *Langevin Monte Carlo* (LMC) [Roberts and Tweedie 1996]. The key feature of these methods lies in their use of the gradients of the target function with respect to the sampling space. This allows them to adapt the proposal distribution to the shape of the target function, improving sampling efficiency.

MCMC forward rendering. MCMC techniques [Sik and Křivánek 2020] were introduced to physics-based rendering by Veach and Guibas [1997]. Their algorithm, termed Metropolis Light Transport, operates in *path space*, generating new path proposals by directly modifying the vertices of previously sampled paths. Subsequent path-space algorithms have introduced proposal strategies targeting hard-to-sample specular or near-specular paths [Jakob and Marschner 2012; Kaplanyan et al. 2014; Hanika et al. 2015], dealing with complex visibility [Otsu et al. 2018], or focusing on spatial image gradients [Lehtinen et al. 2013].

Alternatively, Kelemen et al. [2002] have proposed to operate MCMC rendering in the *primary-sample space* by generating new proposals of primary samples (i.e., random numbers used to construct light paths via predetermined procedures). Although sometimes less efficient than path-space MCMC methods, primary-sample-space techniques are significantly simpler to design and analyze.

Using the primary-sample space formulation, HMC and LMC have been introduced to physics-based rendering by Li et al. [2015] and Luan et al. [2020], respectively. By using the gradients of the light path contribution with respect to the sampling space, these methods enable a general way to adapt the proposal distribution to sparse and anisotropic target functions. Our technique is built upon the PSSMLT rendering framework by Kelemen et al. [2002] but operates over a different space corresponding to boundary light paths, and we optionally use techniques from LMC [Luan et al. 2020] to improve the sampling efficiency.

Monte Carlo sampling of boundary paths. The first general method that samples light paths through boundaries has been introduced by Li et al. [2018]. This technique requires detecting object silhouettes with respect to arbitrary shading points and can be prohibitively expensive for objects with finely detailed geometries.

To address this problem, Zhang et al. [2020] introduced the formulation of differential path integrals—which we also use in this paper. Being a path-space technique, this formulation allows boundary paths to be sampled using a multi-directional process without performing explicit silhouette detection.

To further improve the efficiency of the sampling of boundary paths, Yan et al. [2022] have proposed to utilize hierarchical data structures to guide the sampling. Recently, Zhang et al. [2023] have introduced a technique that utilizes a specialized projection process to guide the sampling process. By leveraging MH-based sampling, our method outperforms these methods on challenging scenes as we will demonstrate in §5.

Reparameterizing boundary integrals. Another class of differentiable rendering methods [Loubet et al. 2019; Bangaru et al. 2020; Vicini et al. 2022; Bangaru et al. 2022; Xu et al. 2023] estimate boundary path integrals after reparameterizing them into interior ones and, thus, do not require directly sampling boundary paths. On the other hand, importance sampling of the reparameterized integrals—which involves hard-to-sample quantities like divergences—remains challenging. As we will demonstrate in §5, our method can offer better efficiency at considerably lower computational cost.

3 PRELIMINARIES

Our contribution is to introduce a Langevin-Monte-Carlo-based sampler for handling discontinuities in differential path integrals. We provide a brief recap of related techniques and definitions before discussing our method.

3.1 Differential Path Integral

3.1.1 Path integrals. Introduced by Veach [Veach 1997], the response I of a radiometric detector can be expressed as a **path integral**:

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

where $\Omega := \bigcup_{N=1}^{\infty} \mathcal{M}^{N+1}$ is the **path space** (with $\mathcal{M}$ denoting the union of all object surfaces) comprising **light paths** $\bar{x} := (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ with $\boldsymbol{x}_0$ on a light source and $\boldsymbol{x}_N$ on the detector, and μ is the corresponding area-product measure. Further, the integrand f is the **measurement contribution function** given by

$$f(\bar{x}) := L_\mathrm{e}(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, W_\mathrm{e}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N) \left[\prod_{n=1}^{N-1} f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})\right] \left[\prod_{n=1}^{N} G(\boldsymbol{x}_{n-1} \leftrightarrow \boldsymbol{x}_n)\right], \tag{2}$$

where L_e and W_e are the **source emission** and **detector importance** (or response), f_s is the **bidirectional scattering distribution function** (BSDF), and G is the (visibility-aware) **geometric term**.

Material-form reparameterization. When the object surfaces $\mathcal{M}$—and hence the path space Ω—evolve with some parameter $\theta \in \mathbb{R}$, differentiating the path integral of Eq. (1) becomes more complicated. To address this problem, Zhang et al. [Zhang et al. 2020] have proposed to reparameterize the evolving object surfaces $\mathcal{M}$ using a one-to-one mapping $\mathsf{X}(\cdot, \theta)$ that, for any θ, transforms a fixed **reference surface** $\mathcal{B}$ to the evolving $\mathcal{M}(\theta)$. Further, to distinguish points on the reference and the evolving surfaces, we call any $\boldsymbol{p} \in \mathcal{B}$ a **material point** and $\boldsymbol{x} \in \mathcal{M}(\theta)$ a **spatial point**.

Let $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \dots, \boldsymbol{p}_N)$ be a **material light path** with each vertex $\boldsymbol{p}_n$ on the reference surface $\mathcal{B}$. Then, the mapping $\mathsf{X}(\cdot, \theta) : \mathcal{B} \mapsto \mathcal{M}(\theta)$ induces a change of variable from $\bar{\boldsymbol{p}}$ to a (spatial) light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_N)$ with $\boldsymbol{x}_n = \mathsf{X}(\boldsymbol{p}_n, \theta)$ for all $0 \le n \le N$. Applying this change of variable to the path integral (1) yields

$$I = \int_{\hat{\Omega}} \underbrace{f(\bar{\boldsymbol{x}}) \prod_{n=0}^{N} \left\| \frac{\mathrm{d}A(\boldsymbol{x}_n)}{\mathrm{d}A(\boldsymbol{p}_n)} \right\|}_{=: \hat{f}(\bar{\boldsymbol{p}})} \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{3}$$

where the domain of integration is the **material path space** $\hat{\Omega} := \bigcup_{N=1}^{\infty} \mathcal{B}^{N+1}$ (which is independent of the parameter θ), and the integrand $\hat{f}$ is called the **material measurement contribution**.

3.1.2 Differential path integrals. [Zhang et al. 2020] have differentiated the material-form path integral of Eq. (3) with respect to the parameter θ using Reynolds transport theorem [1903]. The result can generally be expressed as **material-form differential path integrals**:

$$\frac{\mathrm{d}I}{\mathrm{d}\theta} = \overset{\textbf{interior}}{\boxed{\int_{\hat{\Omega}} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \,\mathrm{d}\mu(\bar{\boldsymbol{p}})}} + \overset{\textbf{boundary}}{\boxed{\int_{\partial\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\, V(\boldsymbol{p}_K)\, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}}. \tag{4}$$

In this equation, the *interior* component is over the same material path space $\hat{\Omega}$ as the material path integral of Eq. (3). The *boundary* component, on the other hand, captures the shift of discontinuous boundaries of the material measurement contribution $\hat{f}$ with respect to the parameter θ.

Boundary path integral. In this paper, we focus on the estimation of the *boundary* component of Eq. (4)—which we explain in more details below.

The domain of this integral is the **material boundary path space** $\partial\hat{\Omega}$. The elements of this space are **material boundary paths** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \dots, \boldsymbol{p}_N) \in \hat{\Omega}$ where exactly one vertex $\boldsymbol{p}_K$ (for some $0 \le K < N$) is constrained over the boundary (i.e., jump discontinuities) of the *mutual visibility* $\mathbb{V}(\boldsymbol{x}_K \leftrightarrow \boldsymbol{x}_{K+1})$ between the spatial points $\boldsymbol{x}_K = \mathsf{X}(\boldsymbol{p}_K, \theta)$ and $\boldsymbol{x}_{K+1} = \mathsf{X}(\boldsymbol{p}_{K+1}, \theta)$ when $\boldsymbol{p}_{K+1}$ is fixed. We call the spatial line segment $\overline{\boldsymbol{x}_K\, \boldsymbol{x}_{K+1}}$—which is tangent to the scene geometry $\mathcal{M}(\theta)$ at a single point $\boldsymbol{x}^{\mathrm{B}}$—a **boundary segment** (see Figure 2).

Further, the measure $\dot{\mu}$ associated with the material boundary path space $\partial\hat{\Omega}$ satisfies that $\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}) = \mathrm{d}\ell(\boldsymbol{p}_K) \prod_{n \neq K} \mathrm{d}A(\boldsymbol{p}_n)$, where ℓ denotes the curve-length measure.

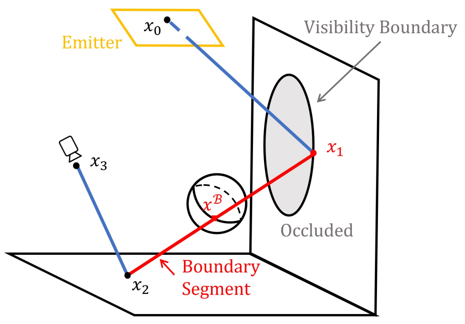

Fig. 2. **Boundary path and visibility boundary:** $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3)$ is (the spatial representation of) a boundary path and contains exactly one boundary segment $\overline{\boldsymbol{x}_1 \boldsymbol{x}_2}$ (illustrated in red) such that one endpoint $\boldsymbol{x}_1$ is constrained on the visibility boundary with respect to the other endpoint $\boldsymbol{x}_2$. This segment is tangent to the scene geometry $\mathcal{M}$ at one other point $\boldsymbol{x}^{\mathrm{B}}$ (which is not a vertex of this path).

Lastly, $V(\boldsymbol{p}_K)$ is the **scalar normal velocity** defined as

$$V(\boldsymbol{p}_K) = \boldsymbol{n}^{\partial}(\boldsymbol{p}_K) \cdot \frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\theta}, \tag{5}$$

where $\boldsymbol{n}^{\partial}(\boldsymbol{p}_K)$ denotes the unit normal of the aforementioned visibility boundary at $\boldsymbol{p}_K$—which we assume without loss of generality to point toward the occluded side (as shown in Figure 2).

3.1.3 Multi-directional sampling of boundary paths. Numerically estimating the *boundary* component in Eq. (4) is challenging. To avoid explicit silhouette detection—which can be prohibitively expensive for geometrically complex scenes—[Zhang et al. 2020] have proposed to sample boundary paths in a *multi-directional* fashion by first obtaining the boundary segment. Then, a *source subpath* and a *detector subpath* are built from the two endpoints of the boundary segment, respectively.

To efficiently importance sample the boundary segment, various guiding methods have been proposed previously [Yan et al. 2022]. Unfortunately, as we will demonstrate in §4, the effectiveness of these methods can degrade for scenes with complex light transport effects such as specular reflection.

In §4, we will introduce a Markov-Chain Monte Carlo (MCMC) method for robust and efficient sampling of boundary paths.

3.2 Markov-Chain-Monte-Carlo Rendering

The basis of our method is Markov-Chain Monte Carlo (MCMC) introduced to rendering by Veach and Guibas [Veach and Guibas 1997]. Unlike ordinary Monte Carlo methods that use independent samples, MCMC methods generate Markov chains—sequences of correlated samples.

Metropolis-Hastings. A commonly used MCMC algorithm is the Metropolis-Hastings (MH) method. Given a non-negative **target function** F, this method allows generating a chain of samples $\{\boldsymbol{u}^t \in \mathcal{U} : t = 1, 2, \dots\}$ distributed asymptotically proportional to F.

The MH method works by iteratively proposing new samples and determining if the proposed samples are accepted. Precisely, after predetermining a **proposal distribution** $\mathcal{T} : \mathcal{U}^2 \mapsto \mathbb{R}_{\ge 0}$, for each the sample $\boldsymbol{u}^t$ in the chain, a new sample $\boldsymbol{v}$ is drawn with a probability proportional to $\mathcal{T}(\boldsymbol{u}^t \to \cdot)$. Then, with the probability

$$\alpha(\boldsymbol{v} \,|\, \boldsymbol{u}^t) := \min\left(1,\ \frac{F(\boldsymbol{v})\,\mathcal{T}(\boldsymbol{u}^t \to \boldsymbol{v})}{F(\boldsymbol{u}^t)\,\mathcal{T}(\boldsymbol{v} \to \boldsymbol{u}^t)}\right), \tag{6}$$

the proposed sample $\boldsymbol{v}$ is accepted by having $\boldsymbol{u}^{t+1} = \boldsymbol{v}$. Otherwise, the current sample $\boldsymbol{u}^t$ is preserved by setting $\boldsymbol{u}^{t+1} = \boldsymbol{u}^t$.

Global mutation and local perturbation. Kelemen et al. [2002] have demonstrated that, for complex target functions, it is beneficial to combine two types of proposal distributions to generate samples efficiently. Specifically, at any $\boldsymbol{u}^t$, one can use the proposal distribution

$$\mathcal{T}(\boldsymbol{u}^t \to \boldsymbol{v}) = \pi\, \mathcal{T}_{\text{global}}(\boldsymbol{u}^t \to \boldsymbol{v}) + (1-\pi)\, \mathcal{T}_{\text{local}}(\boldsymbol{u}^t \to \boldsymbol{v}), \tag{7}$$

where:

- $\mathcal{T}_{\text{global}}(\boldsymbol{u}^t \to \boldsymbol{v})$ is the **global mutation** proposal that makes large modifications and helps jumping among distant peaks of the target function;
- $\mathcal{T}_{\text{local}}(\boldsymbol{u}^t \to \boldsymbol{v})$ is the **local perturbation** proposal that applies small changes and allows efficient local exploration around $\boldsymbol{u}^t$;
- $\pi \in (0, 1)$ is a hyperparameter controlling the probability for the *local* kernel to be used.

4 OUR METHOD

The main objective of this paper is to estimate the *boundary* component of the differential path integral of Eq. (4):

$$I_{\text{bnd}} := \int_{\partial\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\, V(\boldsymbol{p}_K)\, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}). \tag{8}$$

In what follows, we first rewrite this integral as one over the primary-sample space in §4.1. Then, we discuss how this integral can be estimated efficiently using Markov-Chain Monte Carlo (MCMC) in §4.2 and §4.3.

4.1 Primary-Sample-Space Formulation

A material boundary path $\bar{\boldsymbol{p}} = (\boldsymbol{p}^{\text{S}}_s, \ldots, \boldsymbol{p}^{\text{S}}_0, \boldsymbol{p}^{\text{D}}_0, \ldots, \boldsymbol{p}^{\text{D}}_t)$, as stated in §3.1.3, is typically sampled in a *multi-directional* fashion via the following two steps.

S.1 Drawing the boundary segment $\overline{\boldsymbol{p}^{\text{S}}_0 \boldsymbol{p}^{\text{D}}_0}$ by first drawing a spatial point $\boldsymbol{x}^{\text{B}} \in \mathcal{M}$ and a unit vector $\boldsymbol{\omega}^{\text{B}} \in \mathbb{S}^2$. Then, the spatial representation $\overline{x^{\text{S}}_0 x^{\text{D}}_0}$ of the boundary segment $\overline{\boldsymbol{p}^{\text{S}}_0 \boldsymbol{p}^{\text{D}}_0}$ is obtained using ray tracing via

$$x^{\text{S}}_0 = \mathsf{rayIntersect}(\boldsymbol{x}^{\text{B}}, -\boldsymbol{\omega}^{\text{B}}),\ x^{\text{D}}_0 = \mathsf{rayIntersect}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}^{\text{B}}). \tag{9}$$

This step consumes three primary samples (i.e., uniform random numbers) $\boldsymbol{u}^{\text{B}} \in [0, 1)^3$. When the scene geometry is described using polygonal meshes—which is the case we focus on—one random number is used to draw $\boldsymbol{x}^{\text{B}}$ (from an edge of a polygonal face) and two for the direction $\boldsymbol{\omega}^{\text{B}}$. In the rest of this paper, we refer to the point-vector pair $(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}^{\text{B}})$ as a **boundary ray**.

S.2 Building a **source subpath** $(\boldsymbol{p}^{\text{S}}_s, \ldots, \boldsymbol{p}^{\text{S}}_0)$ from $\boldsymbol{p}^{\text{S}}_0$ with $\boldsymbol{p}^{\text{S}}_s$ on a light source and a **detector subpath** $(\boldsymbol{p}^{\text{D}}_0, \ldots, \boldsymbol{p}^{\text{D}}_t)$ from $\boldsymbol{p}^{\text{D}}_0$ with $\boldsymbol{p}^{\text{D}}_t$ on the detector using standard path sampling methods that consume two sequence of random numbers $\boldsymbol{u}^{\text{S}}$ and $\boldsymbol{u}^{\text{D}}$ for the two subpaths, respectively. In practice, we use unidirectional path and particle tracing to construct the source and detector subpaths. For the former, we rely solely on BSDF sampling—That is, without next-event estimation (NEE).

This two-step sampling process allows the *boundary* integral of Eq. (8) to be rewritten as one over the primary-sample space:

$$I_{\text{bnd}} = \int_{[0,1)^3} \int_{\mathcal{U}^2} \hat{f}(\bar{\boldsymbol{p}})\, V(\boldsymbol{p}^{\text{D}}_0)\, J_{\boldsymbol{u}}(\boldsymbol{u}^{\text{B}}, \boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}})\, \mathrm{d}\boldsymbol{u}^{\text{S}}\, \mathrm{d}\boldsymbol{u}^{\text{D}}\, \mathrm{d}\boldsymbol{u}^{\text{B}}, \tag{10}$$

where $\mathcal{U} := \bigcup_{n=0}^{\infty} [0, 1)^n$ is the space of random-number sequences. In addition,

$$J_{\boldsymbol{u}}(\boldsymbol{u}^{\text{B}}, \boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}}) := \left\| \frac{\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{u}^{\text{S}}\, \mathrm{d}\boldsymbol{u}^{\text{D}}\, \mathrm{d}\boldsymbol{u}^{\text{B}}} \right\|, \tag{11}$$

is the Jacobian term that captures the change of variable from material bound path $\bar{\boldsymbol{p}} \in \partial\hat{\Omega}$ to random numbers $(\boldsymbol{u}^{\text{B}}, \boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}}) \in [0, 1)^3 \times \mathcal{U}^2$ and equals the reciprocal of the probability density for drawing the material boundary path $\bar{\boldsymbol{p}}$ (using the aforementioned two-step sampling process).

Computational challenges. As observed by prior works [Yan et al. 2022; Zhang et al. 2023], the mapping from the primary sample $\boldsymbol{u}^{\text{B}}$ to the boundary ray $\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}^{\text{B}}$ can contain many jump discontinuities, causing the importance sampling of $\boldsymbol{u}^{\text{B}}$ challenging. This primarily emerges from the sampling of the point $\boldsymbol{x}^{\text{B}}$ over mesh edges—for which globally continuous parameterizations are hard to obtain.

To address this problem, prior methods [Yan et al. 2022; Zhang et al. 2023] rely on hierarchical structures (e.g, kdtrees or octrees) to guide the sampling. Unfortunately, the performance of these methods degrades when the hierarchical structures fail to accurately capture rapid changes caused by the discontinuities—which tends to occur for highly tessellated scenes (even with techniques like edge sorting [Yan et al. 2022] applied).

In the following, we propose an efficient sampling algorithm that does not require complex guiding data structures. Instead, our technique applies Markov-Chain Monte Carlo (MCMC) with specialized local perturbation strategies that allow a sample to move across discontinuity boundaries in the primary-sample space. Consequently, our method remains robust and efficient even for highly tessellated meshes—which we will demonstrate in §5.

4.2 Basic MCMC Sampler

With the primary-sample-space integral of Eq. (10) established, we apply Markov-Chain Monte Carlo (MCMC) described in §3.2 to efficiently draw the primary samples $\boldsymbol{u}^{\text{B}} \in [0, 1)^3$ and $\boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}} \in \mathcal{U}$.

Target function. We set the target function as

$$F(\boldsymbol{u}^{\text{B}}, \boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}}) := \hat{f}(\bar{\boldsymbol{p}})\, J_{\boldsymbol{u}}(\boldsymbol{u}^{\text{B}}, \boldsymbol{u}^{\text{S}}, \boldsymbol{u}^{\text{D}}). \tag{12}$$

We do not include the scalar normal velocity $V(\boldsymbol{p}^{\text{D}}_0)$ in this function because: (i) it can take negative values; (ii) computing it requires evaluating the derivative $\mathrm{d}\boldsymbol{p}^{\text{D}}_0/\mathrm{d}\theta$—which can be expensive to obtain at render time when using reverse-mode automatic differentiation; and (iii) when differentiating with respect to multiple parameters—a common scenario in inverse rendering—the normal velocity $V(\boldsymbol{p}^{\text{D}}_0)$

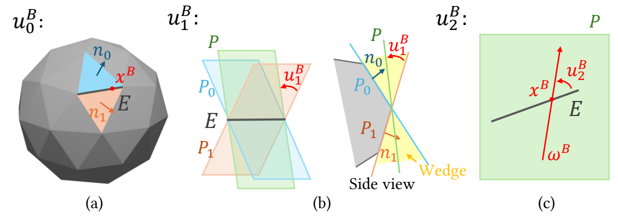

Fig. 3. **Boundary ray sampling:** A boundary ray $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ is typically drawn with random numbers $(u_0^{\mathrm{B}}, u_1^{\mathrm{B}}, u_2^{\mathrm{B}}) \in [0,1)^3$ in three steps as follows. (a) u_0^{B} is used to randomly select an edge E and a point $\boldsymbol{x}^{\mathrm{B}}$ on the edge. (b) The two planes P_0 and P_1 containing the two faces sharing the selected edge E determines a "wedge". u_1^{B} is then used to sample a plane P within this wedge. (c) Lastly, u_2^{B} is used to sample the direction $\boldsymbol{\omega}^{\mathrm{B}}$ within the plane P.

becomes vector-valued (i.e., one component per parameter), making it nontrivial to define a scalar-valued target function F.

Local perturbation. Our local perturbation is based on the scheme introduced by Kelemen et al. [2002]. At each state $\boldsymbol{u}^t$, a new proposal is drawn from an isotropic Gaussian distribution centered at $\boldsymbol{u}^t$:

$$\mathcal{T}_{\text{local}}(\boldsymbol{u}^t \to \boldsymbol{v}) = \mathcal{N}\left(\boldsymbol{v};\ \boldsymbol{u}^t, \epsilon \boldsymbol{I}\right), \tag{13}$$

where $\epsilon \in \mathbb{R}_{>0}$ is the step length of the local perturbation.

Unfortunately, as discussed in §4.1, the target function of Eq. (12) can be extremely fragmented, causing the proposal $\boldsymbol{v}$ obtained using Eq. (13) to have a high chance of being rejected by the Metropolis-Hasting step using Eq. (6). To address this problem, we introduce several specialized operations—which we will discuss in §4.3.

Global mutation. As discussed in §3.2, it is generally beneficial to use a linear combination of local perturbations and global mutations for the full proposal distribution. This is also the case for us, due to the highly discontinuous nature of the target function (12). In theory, we can use any existing (non-MCMC) sampling methods—including the previous guiding-based methods [Yan et al. 2022; Zhang et al. 2023]—for global mutations. However, our experiments indicate that uniformly (re)sampling the primary samples $\boldsymbol{u}^{\mathrm{B}}$, $\boldsymbol{u}^{\mathrm{S}}$ and $\boldsymbol{u}^{\mathrm{D}}$ works adequately without introducing computational overhead from the guiding procedures.

4.3 Improving Local Perturbation

As discussed in §4.2, the vanilla local perturbation in Eq. (13) can be inadequate—especially when the target function is highly fragmented (i.e., contains many disconnected pieces). To address this problem, we introduce two major improvements to the local perturbation in the following.

4.3.1 Rolling. Before introducing the first improvement, we first detail how a boundary ray $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ is drawn using the primary sample $\boldsymbol{u}^{\mathrm{B}} := (u_0^{\mathrm{B}}, u_1^{\mathrm{B}}, u_2^{\mathrm{B}}) \in [0,1)^3$ in Step **S.1**.

When the scene geometry is expressed with polygonal meshes—which is the case we focus on in this paper—the spatial point $\boldsymbol{x}^{\mathrm{B}}$ is drawn from the edges of the mesh using the first component $u_0^{\mathrm{B}} \in [0,1)$. The sampling process—when implemented by selecting first an edge and then a point on that edge—bijectively maps each edge E to a sub-interval $U(E) \subseteq [0,1)$. After drawing $\boldsymbol{x}^{\mathrm{B}}$ from some edge E, the direction $\boldsymbol{\omega}^{\mathrm{B}}$ is then sampled using the last two components $(u_1^{\mathrm{B}}, u_2^{\mathrm{B}})$ from a wedge determined by the face(s) associated with the edge E. Precisely, as illustrated in Figure 3, the second component u_1^{B} is used for drawing a tangent plane from the wedge, and the third component u_2^{B} is consumed to select a direction within this plane.

Given this sampling process, it holds that, when restricting the primary sample $\boldsymbol{u}^{\mathrm{B}}$ inside the axis-aligned box $\mathcal{U}(E) := U(E) \times [0,1)^2$ for any edge E, the mapping from $\boldsymbol{u}^{\mathrm{B}}$ to the boundary ray $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ is continuous. On the contrary, when $\boldsymbol{u}^{\mathrm{B}}$ moves across the boundary of a box $\mathcal{U}(E)$, the corresponding boundary ray, if exists, can change drastically.

Based on this observation, we introduce additional rules to handle situations when a local perturbation attempts to move across the boundary of some box $\mathcal{U}(E)$. Precisely, let the current state be $\boldsymbol{u}^t := (\boldsymbol{u}^{\mathrm{B}}, \boldsymbol{u}^{\mathrm{S}}, \boldsymbol{u}^{\mathrm{D}})$ with $\boldsymbol{u}^{\mathrm{B}} \in \mathcal{U}(E)$ for some edge E and $\boldsymbol{v} := (\boldsymbol{v}^{\mathrm{B}}, \boldsymbol{v}^{\mathrm{S}}, \boldsymbol{v}^{\mathrm{D}})$ be a proposal drawn from Eq. (13). When $\boldsymbol{v}^{\mathrm{B}}$ belongs to the same box $\mathcal{U}(E)$, we proceed as normal by applying the Metropolis-Hasting step in Eq. (6) to the full proposal $\boldsymbol{v}$. Otherwise, we call the (primary-sample-space) intersection between the boundary of the box $\mathcal{U}(E)$ and the line segment connecting $\boldsymbol{u}^{\mathrm{B}}$ and $\boldsymbol{v}^{\mathrm{B}}$ an "*event point*", and attempt to update $\boldsymbol{v}^{\mathrm{B}}$ deterministically based on where the event point is as follows.

Case 1. When the event point resides on the lower/upper bounds of the box $\mathcal{U}(E)$'s first dimension, it holds that the local perturbation attempts to move $\boldsymbol{x}^{\mathrm{B}}$ off the current edge E (see Figure 4-a). In this case, let U be the vertex corresponding to the face containing the event point, we update $\boldsymbol{v}^{\mathrm{B}} = (v_0^{\mathrm{B}}, v_1^{\mathrm{B}}, v_2^{\mathrm{B}}) \in [0,1)^3$ of the proposal $\boldsymbol{v}$ in three steps as follows:

(1) We select another edge E' (uniformly at random) associated with the vertex U;
(2) We "snap" the perturbed point $\boldsymbol{x}^{\mathrm{B}}$ to E' and update the primary sample v_0^{B} accordingly;
(3) We recompute $(v_1^{\mathrm{B}}, v_2^{\mathrm{B}})$ so that the direction $\boldsymbol{\omega}^{\mathrm{B}}$ remains unchanged.

To determine the set of valid edges for the first step, we use the perturbed direction $\boldsymbol{\omega}^{\mathrm{B}}$ (given by $\boldsymbol{v}^{\mathrm{B}}$). Specifically, an edge E' is valid only when $\boldsymbol{\omega}^{\mathrm{B}}$ resides inside the wedge associated with E', which can be determined by testing if

$$(\boldsymbol{n}_1 \cdot \boldsymbol{\omega}^{\mathrm{B}})(\boldsymbol{n}_2 \cdot \boldsymbol{\omega}^{\mathrm{B}}) < 0, \tag{14}$$

where $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are the (outward) normals of the two faces sharing the edge. If there is no valid candidate, we simply reject entire proposal $\boldsymbol{v}$ (and set $\boldsymbol{u}^{t+1} = \boldsymbol{u}^t$).

Case 2. When the event point lands the bounds of the box $\mathcal{U}(E)$'s second dimension, the local perturbation would cause $\boldsymbol{x}^{\mathrm{B}}$ to move out of the wedge determined by E (as illustrated in Figure 3-b). In this case, we "roll" the boundary ray around a face associated with E to obtain a new ray located at some $\boldsymbol{x}^{\mathrm{B}}_{\text{new}}$ on another edge of this face (see Figure 4-b).

Case 3. The last dimension v_2^{B} of the perturbed primary sample $\boldsymbol{v}^{\mathrm{B}}$ is used to draw the direction $\boldsymbol{\omega}^{\mathrm{B}}$ of the perturbed boundary

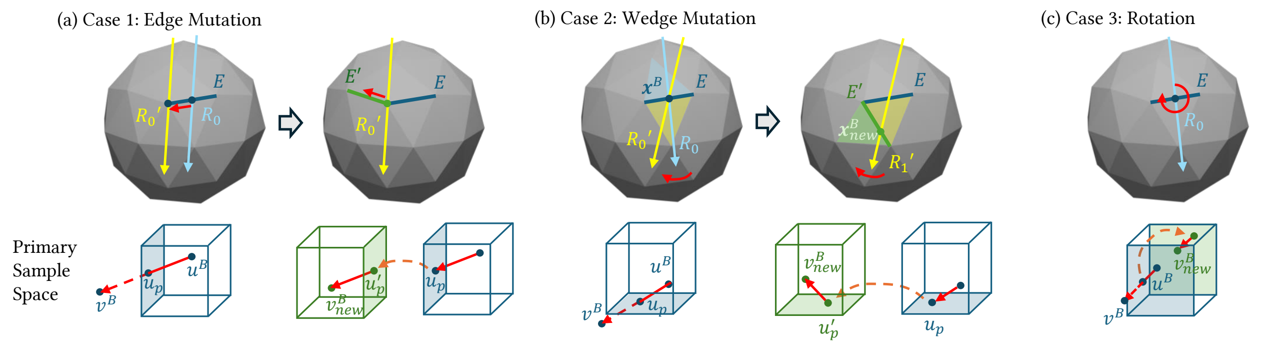

Fig. 4. **Improved local perturbation:** When a local perturbation attempts to move a primary sample $\boldsymbol{u}^{\mathrm{B}}$ to a point $\boldsymbol{v}^{\mathrm{B}}$ outside the box $\mathcal{U}(E)$ illustrated in blue, we move $\boldsymbol{u}^{\mathrm{B}}$ to a new location $\boldsymbol{v}^{\mathrm{B}}_{\mathrm{new}}$ deterministically based on the following rules. Let $\boldsymbol{u}_{\mathrm{p}}$ be the *event point* (i.e., the intersection between the boundary of the box $\mathcal{U}(E)$ and the segment connecting $\boldsymbol{u}^{\mathrm{B}}$ and $\boldsymbol{v}^{\mathrm{B}}$). (a) When $\boldsymbol{u}_{\mathrm{p}}$ is located on the bounds of the first dimension of the box, the perturbation would cause the boundary ray to move off the edge E. In this case, we compute the boundary ray R_0' corresponding to the event point $\boldsymbol{u}_{\mathrm{p}}$ and select an adjacent edge E' such that the point $\boldsymbol{u}_{\mathrm{p}}'$ on the boundary of the box $\mathcal{U}(E')$ also produced the ray R_0'. Then, we continue the perturbation (based on the offset from $\boldsymbol{u}_{\mathrm{p}}$ to v^{B}) to obtain the point $\boldsymbol{v}^{\mathrm{B}}_{\mathrm{new}}$. (b) When $\boldsymbol{u}_{\mathrm{p}}$ resides on the bounds of the second dimension, the perturbation would rotate the boundary ray R_0 out of the wedge associated with the edge E. In this case, the event point $\boldsymbol{u}_{\mathrm{p}}$ produces a boundary ray R_0' that resides inside the plane containing a face associated with E (the yellow triangle in this example). We compute another boundary ray R_1' that coincides with R_0' except for having the position $\boldsymbol{x}^{\mathrm{B}}_{\mathrm{new}}$ (instead of $\boldsymbol{x}^{\mathrm{B}}$). Then, we find the primary-space point $\boldsymbol{u}_{\mathrm{p}}'$ corresponding to R_1' and continue the perturbation to obtain $\boldsymbol{v}^{\mathrm{B}}_{\mathrm{new}}$. (c) When $\boldsymbol{u}_{\mathrm{p}}$ is located on the bounds of the third dimension, the perturbation would rotate the boundary ray R_0 around $\boldsymbol{x}^{\mathrm{B}}$ with an angle beyond the range $[0, 2\pi)$. In this case, we simply map the point $\boldsymbol{v}^{\mathrm{B}}$ back into the box (by entering from the opposite face).

ray $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ within a predetermined plane (see Figure 3-c). Since the mapping from $[0, 1)$ to directions on a plane is cyclic, we simply make the point to re-enter the box $\mathcal{U}(E)$ from the opposite (see Figure 4-c).

Additional details. When jumping between boxes based on the three rules above, when entering a new box and exiting the current one from the same side (e.g., Figure 4-b), we flip the perturbation direction (in the primary-sample space) accordingly. Lastly, we allow a single perturbation to travel across multiple boxes.

Reversibility. It is easy to verify that, based the aforementioned rules, our local perturbation operation is fully reversible. Therefore, the mapping from the current state to the proposal state is one-to-one, which ensures the unbiasedness of our method.

4.3.2 *Langevin Monte Carlo.*

To further improve the local perturbation efficiency, we adopt the Langevin Monte Carlo technique similar to the work by Luan et al. [2020]. Specifically, we update the local perturbation in Eq. (13) to:

$$\mathcal{T}_{\mathrm{local}}(\boldsymbol{u}^t \to \boldsymbol{v}) = \mathcal{N}\left(\boldsymbol{v};\ \boldsymbol{u}^t + \frac{1}{2}\epsilon\,\nabla_{\boldsymbol{u}}\left(\log F(\boldsymbol{u}^t)\right),\ \epsilon \boldsymbol{I}\right). \tag{15}$$

We forego the Adam-based preconditioning introduced by Luan et al. [2020] since it provides little benefits in our experiments—presumably due to the highly fragmented nature of our target function.

After obtaining the proposal $\boldsymbol{v}$, we update it using the same rules discussed in §4.3.

Scaling the step size. To further improve the robustness of our technique, we apply an non-uniform scaling to the step size. Specifically, at $\boldsymbol{u}^t = (\boldsymbol{u}^{\mathrm{B}}, \boldsymbol{u}^{\mathrm{S}}, \boldsymbol{u}^{\mathrm{D}})$ with $\boldsymbol{u}^{\mathrm{B}} := (u^{\mathrm{B}}_0, u^{\mathrm{B}}_1, u^{\mathrm{B}}_2)$, we replace $\epsilon \in \mathbb{R}_{>0}$ in Eq. (15) with a 3×3 diagonal matrix:

$$\epsilon \begin{pmatrix} \|\mathrm{d}u^{\mathrm{B}}_0/\mathrm{d}(\boldsymbol{x}^{\mathrm{B}},\boldsymbol{\omega}^{\mathrm{B}})\|_1 & & \\ & \|\mathrm{d}u^{\mathrm{B}}_1/\mathrm{d}(\boldsymbol{x}^{\mathrm{B}},\boldsymbol{\omega}^{\mathrm{B}})\|_1 & \\ & & \|\mathrm{d}u^{\mathrm{B}}_2/\mathrm{d}(\boldsymbol{x}^{\mathrm{B}},\boldsymbol{\omega}^{\mathrm{B}})\|_1 \end{pmatrix}, \tag{16}$$

where $\mathrm{d}u^{\mathrm{B}}_i/\mathrm{d}(\boldsymbol{x}^{\mathrm{B}},\boldsymbol{\omega}^{\mathrm{B}})$ is obtained by differentiating the inverse of the mapping from $\boldsymbol{u}^{\mathrm{B}}$ to the boundary ray $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$. This operation essentially re-scales the primary-sample-space mutation so that changes become more even in the path space.

4.4 Full MCMC Estimator

We now present our full MCMC estimator—which involves two main stages—for the *boundary* integral in Eq. (8).

In the first stage, we estimate the *normalization factor* c of the target function F defined as

$$c := \int_{[0,1)^3} \int_{\mathcal{U}^2} F(\boldsymbol{u}^{\mathrm{B}}, \boldsymbol{u}^{\mathrm{S}}, \boldsymbol{u}^{\mathrm{D}})\,\mathrm{d}\boldsymbol{u}^{\mathrm{S}}\,\mathrm{d}\boldsymbol{u}^{\mathrm{D}}\,\mathrm{d}\boldsymbol{u}^{\mathrm{B}} = \int_{\partial\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\,\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}), \tag{17}$$

independently using ordinary Monte Carlo methods (e.g., [Yan et al. 2022; Zhang et al. 2023]). This factor gives the full probability density for our MH-based sampler described in §4.2 and §4.3:

$$\mathrm{pdf}(\boldsymbol{u}^{\mathrm{B}}, \boldsymbol{u}^{\mathrm{S}}, \boldsymbol{u}^{\mathrm{D}}) = F(\boldsymbol{u}^{\mathrm{B}}, \boldsymbol{u}^{\mathrm{S}}, \boldsymbol{u}^{\mathrm{D}})/c. \tag{18}$$

As we estimate the normalization factor c, we store the path samples and their corresponding contributions as "seeds". For more efficient path seeding, similar to the method proposed by Bitterli and Jarosz [2019], we sample an additional set of paths where the source sub-paths are constructed using next-event estimation (NEE). These paths are then mapped backed onto the primary-sample space.

In the second stage, we re-sample a subset of n_{chains} seed paths (using resampled importance sampling [Talbot et al. 2005]), each of which then serves as the initial state that is expanded into a full Markov chain using our MCMC sampler described in §4.2 and §4.3.

Let $\left\{(\boldsymbol{u}_i^{\mathrm{B}}, \boldsymbol{u}_i^{\mathrm{S}}, \boldsymbol{u}_i^{\mathrm{D}}) \,:\, i = 1, 2, \ldots, N\right\}$ be all the samples generated by these Markov chains. Our full estimator of the *boundary* integral then takes the form

$$\langle I_{\mathrm{bnd}} \rangle_{\mathrm{MCMC}} = \frac{1}{N} \sum_{i=1}^{N} \frac{F(\boldsymbol{u}_i^{\mathrm{B}}, \boldsymbol{u}_i^{\mathrm{S}}, \boldsymbol{u}_i^{\mathrm{D}})\, V(\boldsymbol{p}_{0,i}^{\mathrm{D}})}{\mathrm{pdf}(\boldsymbol{u}_i^{\mathrm{B}}, \boldsymbol{u}_i^{\mathrm{S}}, \boldsymbol{u}_i^{\mathrm{D}})} = \frac{1}{N} \sum_{i=1}^{N} c\, V(\boldsymbol{p}_{0,i}^{\mathrm{D}}), \tag{19}$$

where $\boldsymbol{p}_{0,i}^{\mathrm{D}}$ is an endpoint of the boundary segment generated using primary samples $\boldsymbol{u}_i^{\mathrm{B}}$.

In practice, given the definition of the scalar normal velocity V in Eq. (5), our full estimator in Eq. (19) can be evaluated by applying automatic differentiation to

$$\frac{1}{N} \sum_{i=1}^{N} \mathrm{detach}\left(c\, \boldsymbol{n}^{\partial}\left(\boldsymbol{p}_{0,i}^{\mathrm{D}}\right)\right) \cdot \boldsymbol{p}_{0,i}^{\mathrm{D}}, \tag{20}$$

where $\boldsymbol{n}^{\partial}(\boldsymbol{p}_{0,i}^{\mathrm{D}})$ denotes the unit normal of the visibility boundary curve at $\boldsymbol{p}_{0,i}^{\mathrm{D}}$ (pointing toward the occluded side).

5 RESULTS

We implement our MCMC-based estimator described in §4 on the CPU-based system by Xu et al. [2023] that uses the Enzyme automatic differentiation framework [Moses and Churavy 2020]. Our MCMC estimator does not requi

Baselines. We compare our technique to two state-of-the-art baselines: primary-sample-space projective sampling by Zhang et al. [2023] (indicated as "PSDR_proj"), and path-space warped-area sampling by Xu et al. [2023] (denoted as "PSDR_was").

Ablations. We evaluate the effectiveness of our rolling scheme (§4.3) in Figure 5 using a scene containing an object lit by a large area light viewed through a mirror. The differentiable rendering results are with respect to the vertical displacement of the object (visualized using the same colormap as Figure 1). The inverse rendering results—where the shape of the object is optimized—use the same number n_{chains} of chains and varying sample counts for all configurations to conduct equal-time comparisons.

Additionally, we evaluate how correlation affects inverse rendering performance in Figure 6, which uses a scene containing an object inside a glass box. Using 50 multi-view images, we optimize the shape of object (initialized as a sphere). In this experiment, we keep the MCMC step length ϵ and total sample counts fixed, and gradually increase the number n_{chains} of chains. By examining mesh error after 200, 400, and 600 iterations (see Figure 6-c), we observe that using 500–1000 chains provides the best overall efficiency—which we find to be the case for all of our inverse-rendering experiments.

Inverse-rendering comparisons. To further demonstrate the effectiveness of our technique, we show differentiable and inverse rendering results in Figures 7–9. For all experiments, our method use 500–1000 chains with around one million samples in total. We manually tweak the step size ϵ in Eq. (15) so that the acceptance rate is between 0.1 and 0.6. One can also use adaptive MCMC [Andrieu and Thoms 2008] to tune the step size, but we leave this feature out for simplicity. We also skip burn-in and thinning as they have little effect in our experiments.

Table 1. We show the configuration and performance data for our inverse rendering experiments. "# of vert." measures how many vertices are used for the optimization. "spp" shows the average sample per pixel used to evaluate the *boundary* term during training. The last three columns show the time taken (in seconds) for each algorithm to evaluate the *boundary* term.

Scene	# of vert.	spp	Ours	PSDR_proj	PSDR_was
Bunny	40,000	13	2.15	**1.95**	5.54
Dodoco	100,000	23	**1.08**	3.30	8.38
Lucy	50,000	9	**1.05**	2.89	5.40

To compare with PSDR_was and PSDR_proj, we use equal-sample configurations because PSDR_proj uses a different system (i.e., Mitsuba 3's CPU backend) from PSDR_was and our implementation. At equal sample, our technique normally runs faster than the two baselines. Please see Table 1 for detailed performance data.

Since all three techniques estimate only the *boundary* integral of Eq. (4), we use the same method for the *interior* component and configure the scenes so that the gradients are dominated by the *boundary* term.

Figure 7 shows a Lucy scene with a glossy object, two mirrors, and a large area light. We optimize the shape of the object using a single target image. Figure 8 contains the **Bunny** scene where an dark object is encapsulated within a glass container (with low surface roughness). Figure 9 shows the **Dodoco** scene where a diffuse object is viewed from a mirror.

All three examples use moderate to high tessellations, and specular light transport is crucial for accurate reconstructions. As demonstrated by the derivative images shown at the top of each figure, our technique produces clean gradients in equal sample, leading to better shape reconstructions.

6 DISCUSSION AND CONCLUSION

Limitations and future work. Our technique only estimates *boundary* path integrals. Leveraging Markov-Chain-Monte-Carlo (MCMC) and/or Langevin-Monte-Carlo (LMC) to efficiently estimate *interior* path integrals in differentiable rendering is a topic worth investigating. Also, although we have taken a first step analyzing how correlation affects inverse rendering, more in-depth explorations are still needed.

Conclusion. In this paper, we introduced a new Markov-Chain-Monte-Carlo (MCMC) method—which operates in the primary-sample space—to efficiently sample *boundary* light paths. At the core of our technique is a specialized local perturbation scheme that allows efficient MCMC sampling of extremely fragmented target functions. We evaluated the effectiveness of our technique by comparing with state-of-the-art baselines [Zhang et al. 2023; Xu et al. 2023] using several synthetic differentiable and inverse rendering examples.

REFERENCES

Christophe Andrieu and Johannes Thoms. 2008. A Tutorial on Adaptive MCMC. *Statistics and Computing* 18, 4 (2008), 343–373.

Sai Bangaru, Michael Gharbi, Tzu-Mao Li, Fujun Luan, Kalyan Sunkavalli, Milos Hasan, Sai Bi, Zexiang Xu, Gilbert Bernstein, and Fredo Durand. 2022. Differentiable Rendering of Neural SDFs through Reparameterization. In *ACM SIGGRAPH Asia 2022 Conference Proceedings* (Daegu, Republic of Korea) *(SIGGRAPH Asia '22)*. Association for Computing Machinery, New York, NY, USA, Article 22, 9 pages. https://doi.org/10.1145/3550469.3555397

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

Michael Betancourt. 2017. A conceptual introduction to Hamiltonian Monte Carlo. *arXiv preprint arXiv:1701.02434* (2017).

Benedikt Bitterli and Wojciech Jarosz. 2019. Selectively metropolised Monte Carlo light transport simulation. *ACM Trans. Graph.* 38, 6 (2019), 153:1–153:10.

Steve Brooks, Andrew Gelman, Galin Jones, and Xiao-Li Meng. 2011. *Handbook of markov chain monte carlo*. CRC press.

Wesley Chang, Venkataram Sivaram, Derek Nowrouzezahrai, Toshiya Hachisuka, Ravi Ramamoorthi, and Tzu-Mao Li. 2023. Parameter-space ReSTIR for Differentiable and Inverse Rendering. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*. Association for Computing Machinery, New York, NY, USA, 18:1–18:10.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. 1987. Hybrid monte carlo. *Physics letters B* (1987).

Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. 2015. Improved half vector space light transport. In *CGF*.

W Keith Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. (1970).

Wenzel Jakob and Steve Marschner. 2012. Manifold exploration: A Markov chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.* 31, 4 (2012), 58:1–58:13.

Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. 2014. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Trans. Graph.* 33, 4 (2014), 102:1–102:13.

Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. 2002. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, Vol. 21. Wiley Online Library, 531–540.

Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. 2013. Gradient-domain metropolis light transport. *ACM Trans. Graph.* 32, 4 (2013), 95:1–95:12.

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

Tzu-Mao Li, Jaakko Lehtinen, Ravi Ramamoorthi, Wenzel Jakob, and Frédo Durand. 2015. Anisotropic Gaussian mutations for metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Trans. Graph.* 34, 6 (2015), 209:1–209:13.

Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. 2019. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 1–14.

Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. 2020. Langevin Monte Carlo rendering with gradient-based adaptation. *ACM Trans. Graph.* 39, 4 (2020), 140:1–140:16.

Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* (1953).

William Moses and Valentin Churavy. 2020. Instead of Rewriting Foreign Code for Machine Learning, Automatically Synthesize Fast Gradients. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 12472–12485.

Radford M Neal et al. 2011. MCMC using Hamiltonian dynamics. *Handbook of markov chain monte carlo* (2011).

Hisanari Otsu, Johannes Hanika, Toshiya Hachisuka, and Carsten Dachsbacher. 2018. Geometry-aware metropolis light transport. *ACM Trans. Graph.* 37, 6 (2018), 278:1–278:11.

Osborne Reynolds. 1903. *Papers on mechanical and physical subjects: the sub-mechanics of the universe*. Vol. 3. The University Press.

Gareth O Roberts and Richard L Tweedie. 1996. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli* (1996).

Martin Sik and Jaroslav Křivánek. 2020. Survey of Markov Chain Monte Carlo Methods in Light Transport Simulation. *IEEE Transactions on Visualization and Computer Graphics* 26, 4 (2020), 1821–1840. https://doi.org/10.1109/TVCG.2018.2880455

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

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

Eric Veach and Leonidas J. Guibas. 1997. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 97)*. ACM Press/Addison-Wesley Publishing Co., 65–76.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. 2022. Differentiable Signed Distance Function Rendering. *ACM Trans. Graph.* 41, 4 (2022), 125:1–125:18.

Yu-Chen Wang, Chris Wyman, Lifan Wu, and Shuang Zhao. 2023. Amortizing Samples in Physics-Based Inverse Rendering Using ReSTIR. *ACM Trans. Graph.* 42, 6 (2023), 214:1–214:17.

Peiyu Xu, Sai Bangaru, Tzu-Mao Li, and Shuang Zhao. 2023. Warped-Area Reparameterization of Differential Path Integrals. *ACM Trans. Graph.* 42, 6 (2023), 213:1–213:18.

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

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

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

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

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

Ziyi Zhang, Nicolas Roussel, and Wenzel Jakob. 2023. Projective Sampling for Differentiable Rendering of Geometry. *ACM Trans. Graph.* 42, 6 (2023), 212:1–212:14.

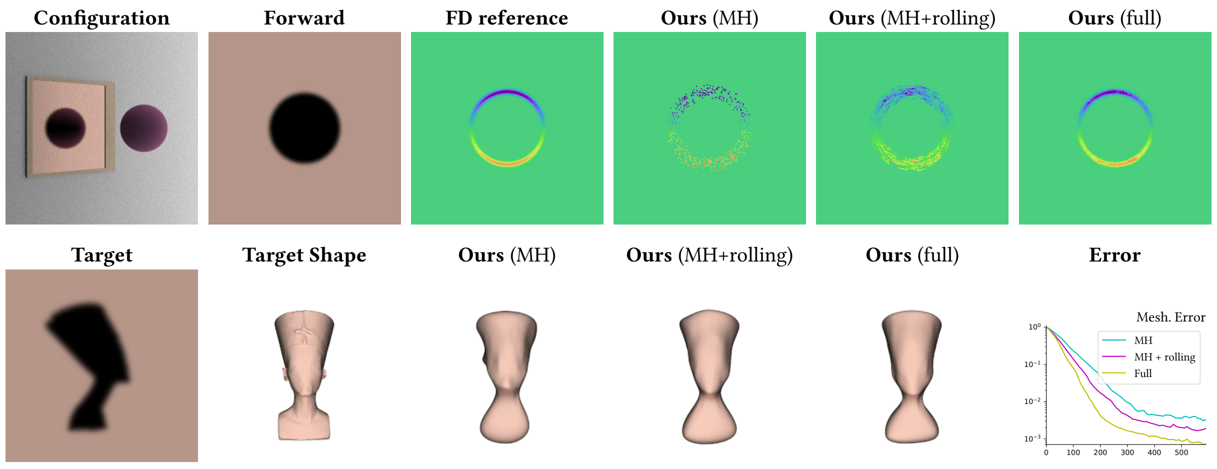

Fig. 5. **Ablation:** We compare differentiable and inverse rendering performance of naive Metropolis Hastings with enabling the local perturbation to roll across face edges, and our full estimator with Langevin Monte Carlo and gradient re-scaling.

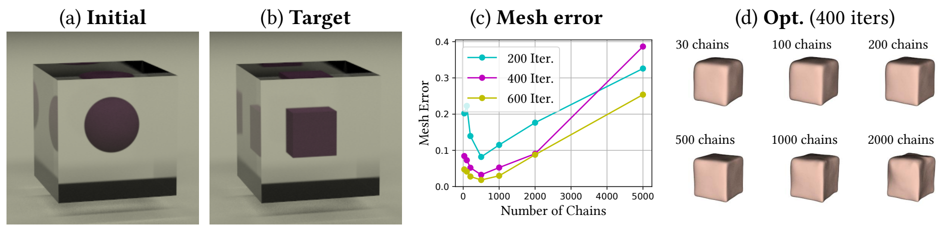

Fig. 6. **Ablation:** We compare inverse-rendering performance of our MCMC estimator with fixed total number of samples and varying number of chains.

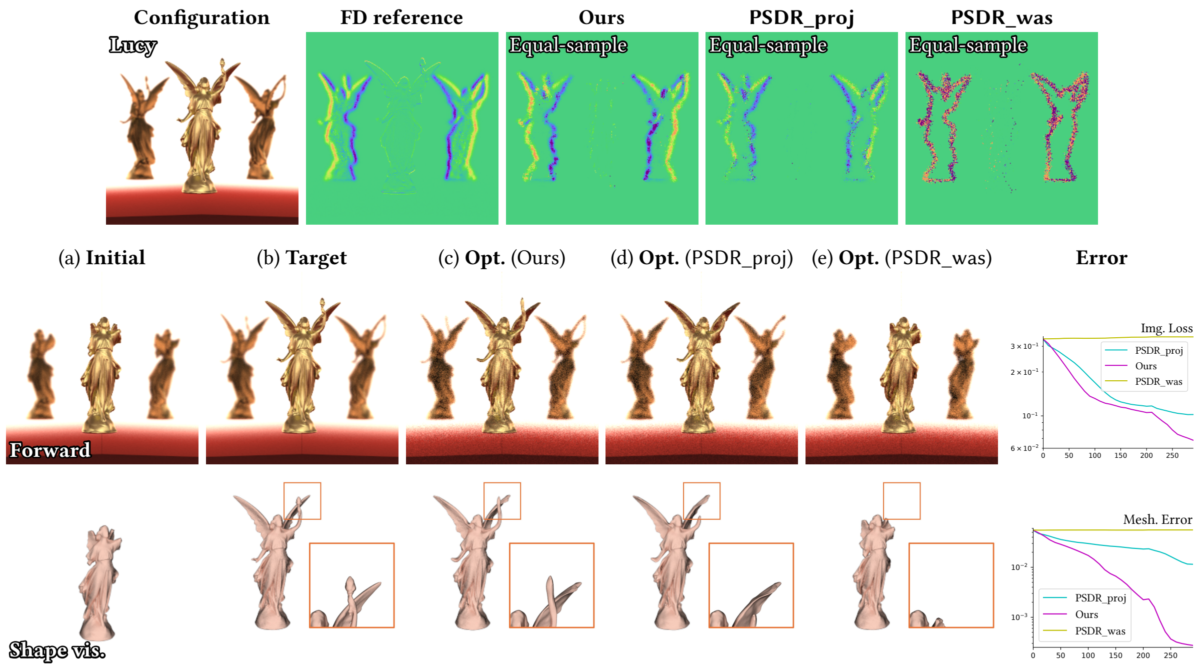

Fig. 7. We compare **differentiable and inverse rendering results** generated using PSDR_proj, PSDR_was, and our technique.

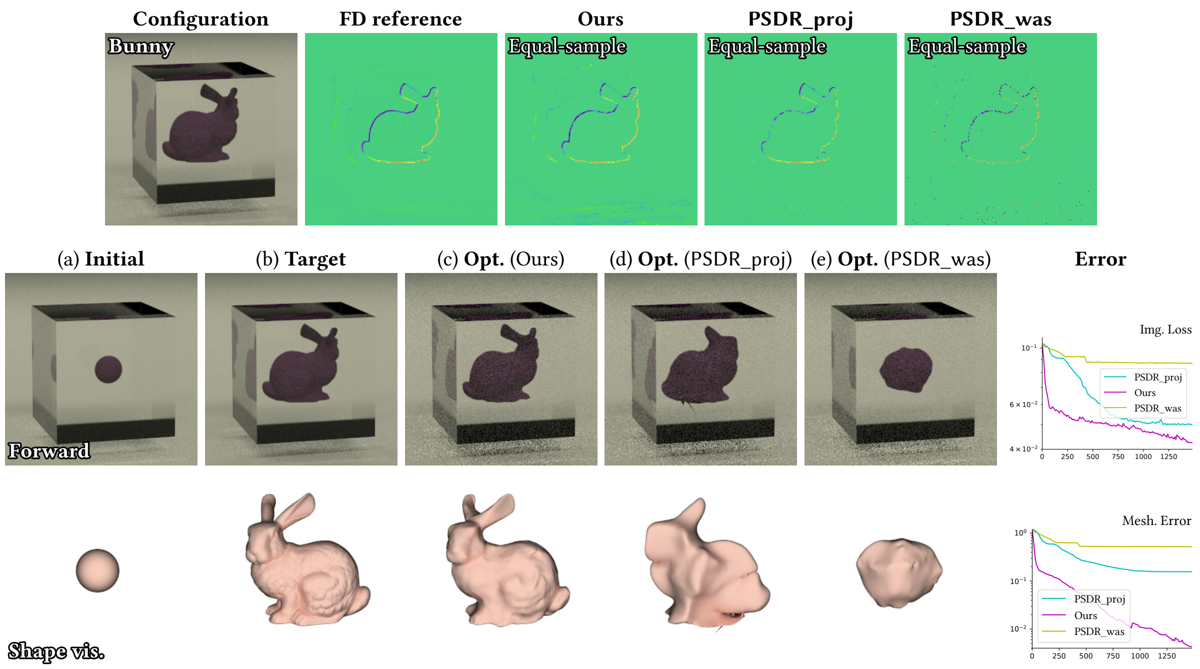

Fig. 8. We compare **differentiable and inverse rendering results** generated using PSDR_proj, PSDR_was, and our technique.

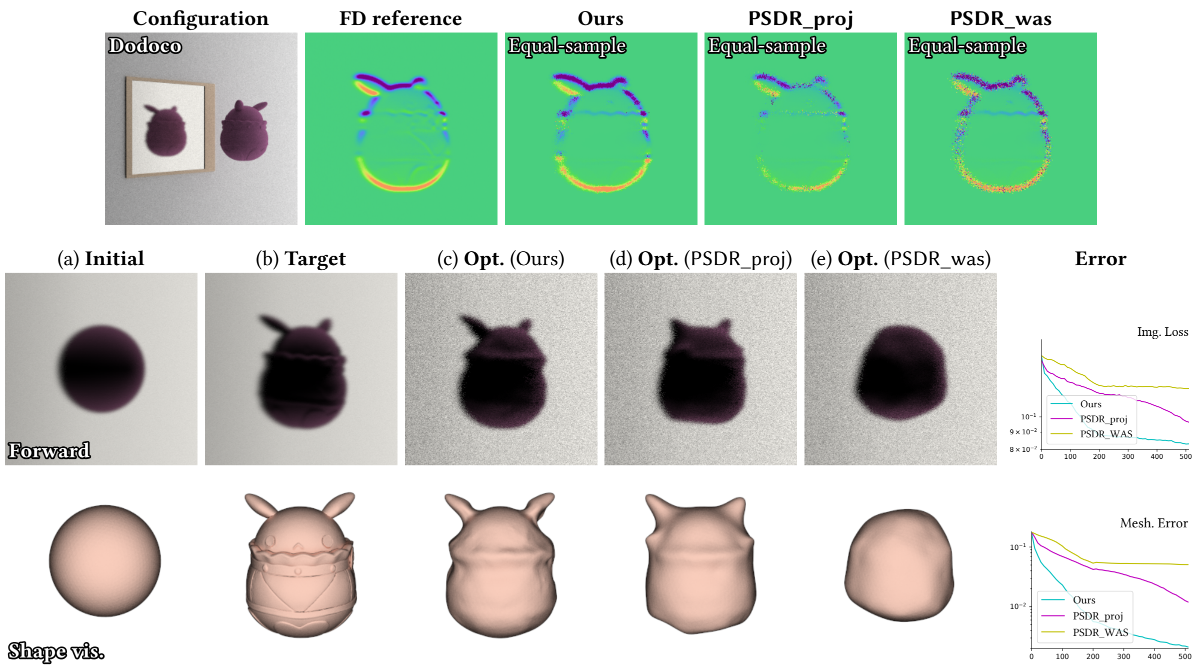

Fig. 9. We compare **differentiable and inverse rendering results** generated using PSDR_proj, PSDR_was, and our technique.