Efficient Estimation of Boundary Integrals for Path-Space Differentiable Rendering

KAI YAN, University of California, Irvine, USA and Meta Reality Labs, USA
CHRISTOPH LASSNER, BRIAN BUDGE, and ZHAO DONG, Meta Reality Labs, USA
SHUANG ZHAO, University of California, Irvine, USA

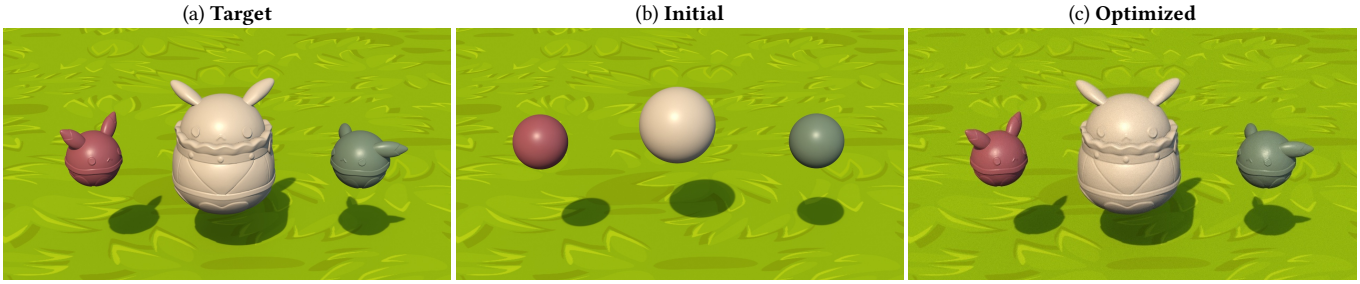


Fig. 1. In physics-based differentiable rendering, *boundary* integrals are crucial for handling scene parameters that control object geometry such as mesh vertex positions. In this paper, we introduce a new Monte Carlo method to efficiently estimate *boundary* integrals under the differential path integral formulation [Zhang et al. 2020]. This example consists of three objects under environmental lighting, and our technique allows efficient reconstruction of their shapes via a single inverse-rendering optimization. (Please use Adobe Acrobat and click column (c) to see the optimization process animated.)

Boundary integrals are unique to physics-based differentiable rendering and crucial for differentiating with respect to object geometry. Under the differential path integral framework—which has enabled the development of sophisticated differentiable rendering algorithms—the boundary components are themselves path integrals. Previously, although the mathematical formulation of boundary path integrals have been established, efficient estimation of these integrals remains challenging.

In this paper, we introduce a new technique to efficiently estimate boundary path integrals. A key component of our technique is a primary-sample-space guiding step for importance sampling of boundary segments. Additionally, we show multiple importance sampling can be used to combine multiple guided samplings. Lastly, we introduce an optional edge sorting step to further improve the runtime performance. We evaluate the effectiveness of our method using several differentiable-rendering and inverse-rendering examples and provide comparisons with existing methods for reconstruction as well as gradient quality.

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

ACM Reference Format:
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, Article 123 (July 2022), 13 pages. https://doi.org/10.1145/3528223.3530080

Authors' addresses: Kai Yan, kyan8@uci.edu, University of California, Irvine, USA and Meta Reality Labs, USA; Christoph Lassner, classner@fb.com; Brian Budge, bbudge@fb.com; Zhao Dong, zhaodong@fb.com, Meta Reality Labs, USA; Shuang Zhao, shz@ics.uci.edu, University of California, Irvine, USA.

0730-0301/2022/7-ART123
https://doi.org/10.1145/3528223.3530080

1 INTRODUCTION

Provided a virtual scene with fully specified object geometries, material optical properties, and detector configurations, *forward rendering* focuses on numerical estimations of a detector's radiometric response. *Differentiable rendering*, in contrast, computes gradients of detector responses with respect to differential changes of the scene and has applications in a wide range of areas such as computational imaging, remote sensing, and computational fabrication.

Recently, great progress has been made in physics-based differentiable rendering theory, algorithms, and systems [Li et al. 2018; Zhang et al. 2019; Loubet et al. 2019; Nimier-David et al. 2019; Zhang et al. 2020; Bangaru et al. 2020; Zhang et al. 2021b,a; Zeltner et al. 2021]. These advances have made it possible to differentiate a scene representation 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. To estimate these terms, previous differentiable rendering techniques have mostly relied on existing stochastic sampling strategies developed for forward rendering. Recently, several differentiable-rendering-specific Monte Carlo methods have been developed [Zeltner et al. 2021; Zhang et al. 2021a] with advantages in terms of speed and quality.

In contrast, the *boundary* integrals are unique to differentiable rendering and defined on discontinuity boundaries of the ordinary rendering integrals. To handle them, several new techniques—such as Monte Carlo edge sampling [Li et al. 2018], reparameterization of the ordinary rendering integral [Loubet et al. 2019; Bangaru et al. 2020], and differential path integrals [Zhang et al. 2020, 2021b]—have been introduced. Among these techniques, differential path

integrals introduced by Zhang et al. [2020; 2021b] have enabled the development of sophisticated differentiable rendering algorithms beyond unidirectional path tracing—similar to ordinary path integrals for forward rendering. Unfortunately, Monte Carlo estimation of *boundary* integrals under Zhang et al.'s path-space formulation remains largely under-explored, making it difficult to solve complex inverse-rendering problems efficiently.

In this paper, we introduce a new technique[1] for efficient estimation of these *boundary* path integrals. Concretely, our contributions include:

- We develop a primary-sample-space guiding technique for efficient importance sampling of boundary light paths (§4.1). Additionally, we show how multiple importance sampling (MIS) can be used to combine multiple guided sampling processes (§4.2).
- We introduce an optional edge sorting step that can further improve the performance of our technique (§4.3).

To show the effectiveness of our method, we compare gradient images estimated with our method and previous techniques in Figures 6 and 7. Additionally, we conduct detailed ablation studies in Figures 8 and 9 to evaluate individual components of our technique. Further, with our improved estimation of *boundary* integrals, inverse-rendering problems under challenging conditions, such as environmental and indirect illumination, can be solved more efficiently. We demonstrate this via several synthetic inverse-rendering examples in Figures 10 and 11.

2 RELATED WORK

Sampling for forward rendering. Monte Carlo sampling of light transport paths has been an essential component for most, if not all, physics-based forward rendering techniques. To this end, a commonly used approach is *local sampling* that grows light paths vertex by vertex. This boils down path sampling to drawing an incident direction given the exitant one (or the other way around) at each vertex—which is typically achieved by importance sampling local BSDFs. Previously, a large variety of BSDF models have been proposed (e.g., [Phong 1975; Cook and Torrance 1982; Ashikhmin and Shirley 2000; Oren and Nayar 1994]). Among these models, microfacet BSDFs (e.g., [Cook and Torrance 1982; Ward 1992; Schlick 1994; van Ginneken et al. 1998; Kelemen and Szirmay-Kalos 2001; Pont and Koenderink 2002; Walter et al. 2007; Heitz et al. 2016; Lee et al. 2018; Xie and Hanrahan 2018]) have been widely adopted. Importance sampling of these models, therefore, has been studied thoroughly and can be performed by sampling the underlying normal distributions [Walter et al. 2007; Heitz and d'Eon 2014] or via localized Monte Carlo processes [Heitz et al. 2016].

All these sampling methods are developed for forward rendering and can be repurposed to estimate *interior* integrals for differentiable rendering. On the other hand, they are largely complementary to the estimation of *boundary* integrals—which is the main focus of this paper.

Path guiding. Recently, several methods (e.g., [Jensen 1995; Lafortune and Willems 1995; Vorba et al. 2014; Müller et al. 2017; Guo et al. 2018; Zheng and Zwicker 2019; Müller et al. 2019]) have been developed to accelerate unidirectional path tracing for forward rendering. Although these techniques are conceptually related to our technique, especially the primary-sample-space guiding step, our technique focuses on a very different use case of estimating boundary integrals that are unique to differentiable rendering.

Physics-based differentiable rendering. Physics-based differentiable rendering is the process of numerically computing derivatives of forward-rendering results with respect to arbitrary scene parameters such as object geometries and optical material properties. Although simple (e.g., one-bounce) light transport can be simulated and differentiated approximately using soft rasterization (e.g., [Laine et al. 2020]) or analytical methods (e.g., [Zhou et al. 2021]), physics-based differentiable rendering generally requires estimating (i) *interior* integrals given by differentiating the integrands of corresponding forward-rendering integrals; and (ii) *boundary* ones defined over discontinuities of those integrands.

Previously, the *interior* integrals have been mostly estimated using path sampling methods developed for forward rendering. Recently, Zeltner et al. [2021] have studied how various parameterizations (e.g., "attached" and "detached") affect the performance of Monte Carlo estimation of the *interior* integral. Additionally, Zhang et al. [2021a] have introduced antithetic sampling (at both BSDF- and path-level) for efficiently estimating the *interior* integral. Both of these methods are orthogonal to our technique.

The *boundary* integrals are unique to differentiable rendering. Recent works have shown that the *boundary* integrals can be estimated by Monte Carlo edge sampling [Li et al. 2018; Zhang et al. 2019] or avoided altogether by reparameterizing rendering integrals [Loubet et al. 2019; Bangaru et al. 2020].

Further, Zhang et al. [2020; 2021b] have introduced the differential path integral framework that formulates both the *interior* and the *boundary* components as full path integrals, making it possible for the development of sophisticated Monte Carlo estimators for both components (beyond unidirectional path tracing). On the other hand, efficiently estimating *boundary* path integrals (by importance sampling full *boundary* light paths) remains challenging. In this paper, we introduce a new solution to this problem.

3 PRELIMINARIES

We now briefly revisit some mathematical and algorithmic preliminaries on the differential path integral formulation (§3.1) and Monte Carlo estimation of *boundary* integrals (§3.2). Table 1 presents a list symbols that are commonly used in this paper and their definitions.

3.1 Differential Path Integral

In forward rendering, the path integral formulation introduced by Veach [1997] has allowed the development of many advanced techniques such as bidirectional path tracing (BDPT). Under this formulation, the intensity of a pixel is given by

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

where $\Omega := \cup_{N \geq 1} \mathcal{M}^{N+1}$ is the **path space** comprised of all **light transport paths** $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ with finite lengths (with

[1] Our implementation is available at https://shuangz.com/projects/psdr-aq-sg22/.

Table 1. List of symbols commonly used in this paper.

Symbol	Definition
θ	abstract scene parameter
$\mathcal{M}(\theta)$	object surfaces
f	measurement contribution
$\Omega(\theta)$	path space
X	motion
$\mathcal{B}$	reference configuration
$\hat{f}$	material measurement contribution
$\hat{\Omega}$	material path space
$\partial\hat{\Omega}(\theta)$	material boundary path space

$\mathcal{M}$ being the union of all object surfaces), f is the **measurement contribution** function, and μ is the area-product measure.

Material-form reparameterization. Differentiable rendering is about differentiating I given by Eq. (1) with respect to some (arbitrary) scene parameter $\theta \in \mathbb{R}$. However, when the scene geometry $\mathcal{M}$ depends on the parameter θ, so will the path space Ω, making the differentiation more difficult. To address this problem, Zhang et al. [2020] propose to reparameterize the path integral (1) using some **reference configuration** $\mathcal{B}$ independent of θ coupled with a mapping (or a **motion**) X such that, for any θ, $\mathsf{X}(\cdot, \theta)$ is a differentiable one-to-one mapping from the reference $\mathcal{B}$ to the scene geometry $\mathcal{M}(\theta)$. This reparameterization induces a change of variable from a ordinary light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ to a **material path** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ via the relation $\boldsymbol{x}_n = \mathsf{X}(\boldsymbol{p}_n, \theta)$ for all $0 \le n \le N$. Applying this change of variable to the path integral (1) yields:

$$I = \int_{\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{2}$$

where $\hat{\Omega} = \cup_{N \ge 1} \mathcal{B}^{N+1}$ is the **material path space**, and $\hat{f}$ is the **material measurement contribution** function that captures the ordinary measurement contribution and the Jacobian resulting from the reparameterization.

In theory, the reference configuration $\mathcal{B}$ and the corresponding mapping X can be arbitrary as long as smoothness conditions hold. In practice, when estimating the gradient $\mathrm{d}I/\mathrm{d}\theta$ at some $\theta = \theta_0$, the reference is typically set to $\mathcal{B} = \mathcal{M}(\theta_0)$.

Differential path integral. Zhang et al. [2020] have recently shown that differentiating Eq. (2) with respect to a scene parameter θ yields the material-form **differential path integral** of the form

$$\frac{\mathrm{d}I}{\mathrm{d}\theta} = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \hat{f}^{\mathrm{B}} \hat{f}^{\mathrm{S}} \hat{f}^{\mathrm{D}} \, \mathrm{d}\dot{\mu}}_{\text{boundary}}, \tag{3}$$

where the *interior* term is obtained by simply differentiating the integrand $\hat{f}$; the *boundary* term, on the other hand, is unique to differentiable rendering.

In Eq. (3), the domain of the *boundary* integral is the **material boundary path space** $\partial\hat{\Omega}$. Unlike the material path space that is independent of the scene parameter θ, $\partial\hat{\Omega}$ generally does depend on θ. If we let superscripts 'S' and 'D' indicate, respectively, vertices

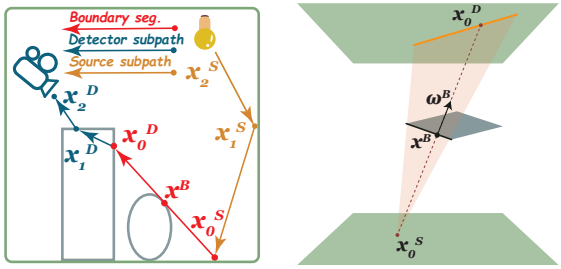


Fig. 2. **Boundary segments:** (a) A boundary light path is comprised of a boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ (shown in red) as well as a source and a detector subpath (illustrated in orange and blue, respectively). The two endpoints of the boundary segment are located on the visibility boundary of each other. (b) A boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ can be uniquely determined with a surface point $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{M}$ interior to the segment, along with a direction $\boldsymbol{\omega}^{\mathrm{B}}$.

belonging to the source and detector subpaths, then each **material boundary light path** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_s^{\mathrm{S}}, \ldots, \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}, \ldots, \boldsymbol{p}_t^{\mathrm{D}}) \in \partial\hat{\Omega}$ contains exactly one **material boundary segment** $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$ such that their corresponding ordinary path vertices $\boldsymbol{x}_0^{\mathrm{S}} = \mathsf{X}(\boldsymbol{p}_0^{\mathrm{S}}, \theta)$ and $\boldsymbol{x}_0^{\mathrm{D}} = \mathsf{X}(\boldsymbol{p}_0^{\mathrm{D}}, \theta)$ lie on the visibility boundary of each other (see Figure 2-a).

In the integrand of the *boundary* integral, given a boundary path, the terms $\hat{f}^{\mathrm{B}}$, $\hat{f}^{\mathrm{S}}$, and $\hat{f}^{\mathrm{D}}$ capture, respectively, the contributions of the boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$, the source subpath $(\boldsymbol{p}_s^{\mathrm{S}}, \ldots, \boldsymbol{p}_0^{\mathrm{S}})$, and the detector subpath $(\boldsymbol{p}_0^{\mathrm{D}}, \ldots, \boldsymbol{p}_t^{\mathrm{D}})$. Please refer to the work by Zhang et al. [2020] for exact definitions of these terms as well as the measure $\dot{\mu}$.

3.2 Multi-Directional Sampling of Boundary Paths

Numerically estimating the *boundary* integral in Eq. (3) is known to be challenging. Monte Carlo edge sampling [Li et al. 2018], for instance, requires detecting object silhouettes at each vertex of a light path and can be prohibitively expensive for scenes with complex geometry.

To avoid explicit silhouette detection, Zhang et al. [2020] propose to sample boundary paths in a multi-directional fashion by (i) sampling the material boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$; and (ii) using standard methods such as unidirectional and bidirectional path sampling to construct the source and detector subpaths.

Using this approach, the first boundary-segment sampling step is unique to path-space differentiable rendering and crucial to the effectiveness of the resulting Monte Carlo estimator. To this end, instead of directly sampling the two endpoints $\boldsymbol{p}_0^{\mathrm{S}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$ (or, equivalently, the corresponding ordinary vertices $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$), Zhang et al. have proposed to sample a point $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{M}$ and a direction $\boldsymbol{\omega}^{\mathrm{B}} \in \mathbb{S}^2$ such that $\boldsymbol{x}^{\mathrm{B}}$ lies in the interior of the segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ and $\boldsymbol{\omega}^{\mathrm{B}}$ determines the directions of the segment. In other words, $\boldsymbol{x}_0^{\mathrm{S}} = \mathrm{rayIntersect}(\boldsymbol{x}^{\mathrm{B}}, -\boldsymbol{\omega}^{\mathrm{B}})$ and $\boldsymbol{x}_0^{\mathrm{D}} = \mathrm{rayIntersect}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$, as illustrated in Figure 2-b.

Sampling the boundary segment this way enjoys the benefit of not requiring explicit silhouette detection (since $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{B}}$ are

guaranteed to located on the visibility boundary of each other). On the other hand, efficiently sampling $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}^{\mathrm{B}}$ remains highly nontrivial. In this paper, we introduce a new technique to tackle this problem.

4 EFFICIENT SAMPLING OF BOUNDARY SEGMENTS

We now introduce our technique that allows efficient sampling of boundary segments.

4.1 Primary-Sample-Space Guiding

With the scene geometry $\mathcal{M}(\theta)$ expressed using polygonal meshes (which is the case for many, if not most, applications in computer graphics and vision), it has been demonstrated that the point $\boldsymbol{x}^{\mathrm{B}}$ must lie on an edge of some polygonal face—a 1D manifold. Further, the direction $\boldsymbol{\omega}^{\mathrm{B}}$ is drawn from a 2D manifold. In other words, the primary sample space for $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ is three-dimensional. This allows us to rewrite the *boundary* integral in Eq. (3) as a primary-sample-space integral:

$$\int_{[0,1)^3} F(u_1, \boldsymbol{u}_2)\, \mathrm{d}\boldsymbol{u}_2\, \mathrm{d}u_1, \tag{4}$$

where $u_1 \in \mathbb{R}$, $\boldsymbol{u}_2 \in \mathbb{R}^2$, and

$$F(u_1, \boldsymbol{u}_2) \coloneqq \underbrace{\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, J^{\mathrm{B}}(u_1, \boldsymbol{u}_2)}_{\text{boundary}} \underbrace{h^{\mathrm{S}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})}_{\text{source}} \underbrace{h^{\mathrm{D}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})}_{\text{detector}}. \tag{5}$$

In Eq. (5), J^{B} is the Jacobian determinant capturing the change of variable from $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}^{\mathrm{B}})$ to $(u_1, \boldsymbol{u}_2)$, and

$$h^{\mathrm{S}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}) \coloneqq \int \hat{f}^{\mathrm{S}}(\bar{\boldsymbol{p}}^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}}^{\mathrm{S}}), \tag{6}$$

$$h^{\mathrm{D}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}) \coloneqq \int \hat{f}^{\mathrm{D}}(\bar{\boldsymbol{p}}^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \mathrm{d}\mu(\bar{\boldsymbol{p}}^{\mathrm{D}}), \tag{7}$$

capture, respectively, the contributions of the source subpath $\bar{\boldsymbol{p}}^{\mathrm{S}} \coloneqq (\boldsymbol{p}_1^{\mathrm{S}}, \boldsymbol{p}_2^{\mathrm{S}}, \ldots)$ and the detector subpath $\bar{\boldsymbol{p}}^{\mathrm{D}} \coloneqq (\boldsymbol{p}_1^{\mathrm{D}}, \boldsymbol{p}_2^{\mathrm{D}}, \ldots)$.

We note that, under this formulation, $\boldsymbol{p}_0^{\mathrm{S}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$ are essentially functions of the primary samples u_1 and $\boldsymbol{u}_2$:

$$\boldsymbol{p}_0^{\mathrm{S}} = \mathsf{X}^{-1}\left(\mathrm{rayIntersect}\left(\boldsymbol{x}^{\mathrm{B}}(u_1),\ -\boldsymbol{\omega}^{\mathrm{B}}(\boldsymbol{u}_2)\right), \theta\right), \tag{8}$$

$$\boldsymbol{p}_0^{\mathrm{D}} = \mathsf{X}^{-1}\left(\mathrm{rayIntersect}\left(\boldsymbol{x}^{\mathrm{B}}(u_1),\ \boldsymbol{\omega}^{\mathrm{B}}(\boldsymbol{u}_2)\right), \theta\right), \tag{9}$$

where $\mathsf{X}^{-1}(\cdot, \theta)$ denotes the inverse of the (predetermined) mapping from the reference configuration $\mathcal{B}$ to the scene geometry $\mathcal{M}(\theta)$.

By randomly drawing the primary samples $(u_1, \boldsymbol{u}_2)$, the source subpath $\bar{\boldsymbol{p}}^{\mathrm{S}}$, and the detector subpath $\bar{\boldsymbol{p}}^{\mathrm{D}}$, we obtain an unbiased single-sample Monte Carlo estimator for Eq. (4):

$$\left\langle \frac{\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, J^{\mathrm{B}}(u_1, \boldsymbol{u}_2)\, \hat{f}^{\mathrm{S}}(\bar{\boldsymbol{p}}^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \hat{f}^{\mathrm{D}}(\bar{\boldsymbol{p}}^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})}{p(u_1, \boldsymbol{u}_2)\, p(\bar{\boldsymbol{p}}^{\mathrm{S}})\, p(\bar{\boldsymbol{p}}^{\mathrm{D}})} \right\rangle, \tag{10}$$

where $p(u_1, \boldsymbol{u}_2)$, $p(\bar{\boldsymbol{p}}^{\mathrm{S}})$, $p(\bar{\boldsymbol{p}}^{\mathrm{D}})$ denote the corresponding probability densities. In practice, given the boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}}$, the source and detector subpaths $\bar{\boldsymbol{p}}^{\mathrm{S}}$ and $\bar{\boldsymbol{p}}^{\mathrm{D}}$ can be constructed using standard unidirectional or bidirectional methods. In what follows, we focus on sampling the boundary segment itself.

Ideally, we would like to have $p(u_1, \boldsymbol{u}_2)$ being proportional to $|F(u_1, \boldsymbol{u}_2)|$. This, however, is difficult because (i) the h^{S} and h^{D} components are given by path integrals; and (ii) the boundary contribution $\hat{f}^{\mathrm{B}}\, J^{\mathrm{B}}$ can be highly complex and potentially discontinuous.

To address these challenges, Zhang et al. [2020] propose to approximate (the absolute value of) Eq. (5) with

$$\tilde{F}(u_1, \boldsymbol{u}_2) = \left|\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\right| J^{\mathrm{B}}(u_1, \boldsymbol{u}_2)\, \tilde{h}^{\mathrm{S}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \tilde{h}^{\mathrm{D}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}), \tag{11}$$

where $\tilde{h}^{\mathrm{S}}$ and $\tilde{h}^{\mathrm{D}}$ are, respectively, approximations of h^{S} and h^{D} obtained using photon mapping. We note that, with the photon (and importon) maps pre-generated, the evaluations of $\tilde{h}^{\mathrm{S}}$ and $\tilde{h}^{\mathrm{D}}$—which query those maps—become deterministic.

Further, Zhang et al. [2020] discretize $\tilde{F}$ as a piecewise-constant function using a regular three-dimensional grid with the value of each cell computed by averaging evaluations of $\tilde{F}$ at several random locations within that cell. With the regular grid computed, it can then be used to importance sample $(u_1, \boldsymbol{u}_2)$. We note that, despite the use of approximations (i.e., photon mapping and discretization), the resulting Monte Carlo method generally remains unbiased since the approximations are used only to construct the PDF $p(u_1, \boldsymbol{u}_2)$.

Although this method works adequately for simple scenes, more complex ones (with, for example, detailed geometries and environmental illumination) can lead to rapidly varying $\tilde{F}$ that requires grids with unpractically high resolutions. To address this problem, we introduce a new technique that utilizes Kd-trees that automatically adapt to the structure of the function $\tilde{F}$ given by Eq. (11).

Kd-tree construction. We use Kd-trees to partition the primary sample space $[0, 1)^3$ for $(u_1, \boldsymbol{u}_2)$. Specifically, we pre-partition the first dimension into multiple intervals each of which corresponds to a face edge. This allows that, for each interval $[a, b) \subset [0, 1)$, the mapping from the primary sample $u_1 \in [a, b)$ to the corresponding point $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{M}(\theta)$ is continuous.

Then, for each interval $[a, b)$, we build a Kd-tree that further subdivides the primary sample space $[a, b) \times [0, 1)^2$ based on the function $\tilde{F}$ of Eq. (11). As described in Algorithm 1, at each tree node r with axis-aligned bounding box *aabb* corresponding to the primary sample space covered by this node, we compute a (tri)linear fit of the function $\tilde{F}$ using samples drawn from *aabb*. If the fitting error is low (i.e., below some predetermined threshold), the fit is likely already a close approximation of $\tilde{F}$, and the node no longer needs to be subdivided. Otherwise, we split the node into two equal-sized children. For each possible dimension $\mathrm{ax} \in \{x, y, z\}$ of the split, we compute fits of $\tilde{F}$ in the two children (Lines 5–8 of Algorithm 1). Then, we select the dimension ax_* with maximal total fitting error, split the node accordingly, and recursively process the two child nodes (Lines 9–13 of Algorithm 1).

When building the Kd-trees, we make the number of samples used for computing (tri)linear fits of $\tilde{F}$ a user-specifiable parameter. Generally, this sampling rate needs to be sufficiently high to capture the detailed structures of $\tilde{F}$. In practice, since we pre-subdivide the first dimension of the primary space, we find 32 to work well for all of our examples.

Sampling. When building the Kd-trees, we also construct a global list containing references of all the leaf nodes. This allows us to

ALGORITHM 1: Build Kd-trees to approximate the function $\tilde{F}$ in Eq. (11) in a piecewise linear fashion

```
BuildTree(r, aabb)
  Input: Current tree node r, the corresponding axis-aligned
         bounding box aabb, and error threshold ε ∈ ℝ_{>0}
begin
    Compute first-order fit of F̃ within aabb with err being the
     fitting error
    if err > ε then
        for ax ∈ {x, y, z} do
            Split the box aabb in half along the axis ax
            Compute first-order fits of F̃ within the two split
             regions and let err_ax be the total fitting error
        end
        ax_* ← arg max_ax err_ax
        Split aabb along ax_* into aabb_1 and aabb_2
        Make r an internal node with children r_1 and r_2
        BuildTree (r_1, aabb_1)
        BuildTree (r_2, aabb_2)
    else
        Make r a leaf node and add it to a (global) list
    end
end
```

create a discrete distribution of all nodes such that the probability mass of each node is proportional to the (trilinear) fit of $\tilde{F}$ integrated over the node's axis-aligned bounding box.

At render time, to sample a boundary segment, we first select a tree node using the discrete distribution. Then, we draw the primary samples $(u_1, \boldsymbol{u}_2) \in \mathbb{R}^3$ using the inversion method based on the fit of the $\tilde{F}$ function associated with that node.

4.2 Multiple Importance Sampling

When estimating the *boundary* integral in Eq. (3), Zhang et al. [2020] utilize next-event estimation by further decomposing this integral into a *direct* and an *indirect* term where the former accounts for boundary paths with the vertex $\boldsymbol{x}_0^{\mathrm{S}} = \mathrm{X}(\boldsymbol{p}_0^{\mathrm{S}}, \theta)$ located on a light source.

To estimate the direct component, after selecting $\boldsymbol{x}^{\mathrm{B}}$, Zhang et al. draw the direction $\boldsymbol{\omega}^{\mathrm{B}} \in \mathbb{S}^2$ of the boundary segment by sampling the light source. For area lights, this process becomes drawing a point $\boldsymbol{x}_0^{\mathrm{S}}$ on the surface of an area light and letting $\boldsymbol{\omega}^{\mathrm{B}} = \mathrm{normalize}(\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}})$. Although this works well for simple (e.g., small area) lighting, it can become less effective for complex illumination conditions (e.g., image-based environmental lighting) since the sampled direction $\boldsymbol{\omega}^{\mathrm{B}}$ can usually be invalid due to occlusion (see Figure 3). In these cases, it is usually more efficient to directly sample the direction $\boldsymbol{\omega}^{\mathrm{B}}$.

To address this problem, we combine multiple importance sampling (MIS) with our primary-sample-space guiding. To this end, one possibility is to guide a full MIS sampler which uses a primary sample $u_1 \in [0, 1)$ to obtain the location $\boldsymbol{x}^{\mathrm{B}}$ as well as $\boldsymbol{u}_2^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{dir}} \in [0, 1)^2$ to draw, respectively, two direction samples $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{light}}$ and $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{dir}}$.

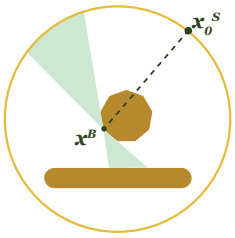


Fig. 3. When sampling the direction $\boldsymbol{\omega}^{\mathrm{B}}$ of the boundary segment by first drawing $\boldsymbol{x}_0^{\mathrm{S}}$ on a light source and setting $\boldsymbol{\omega}^{\mathrm{B}} = \mathrm{normalized}(\boldsymbol{x}_0^{\mathrm{S}} - \boldsymbol{x}^{\mathrm{B}})$, the resulting $\boldsymbol{\omega}^{\mathrm{B}}$ may be invalid: Taking this direction, the boundary segment would penetrate the object.

Unfortunately, using this MIS sampler leads to a target function $F(u_1, \boldsymbol{u}_2^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{dir}})$ over $[0, 1)^5$, which can be expensive to guide.

Instead, we apply our primary-sample-space guiding (§4.1) to two processes that draw the direction $\boldsymbol{\omega}^{\mathrm{B}}$ using light and direction sampling, respectively. These sampling processes give two mappings $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{light}}$ and $\boldsymbol{\omega}^{\mathrm{B}}_{\mathrm{dir}}$ from a primary sample $\boldsymbol{u}_2 \in [0, 1)^2$ to the direction $\boldsymbol{\omega}^{\mathrm{B}}$ of the boundary segment, which further yield two versions of the function $\tilde{F}$ from Eq. (11):

$$\tilde{F}_{\mathrm{light}}(u_1, \boldsymbol{u}_2) = \left|\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\right| J^{\mathrm{B}}_{\mathrm{light}}(u_1, \boldsymbol{u}_2)\, \tilde{h}^{\mathrm{S}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \tilde{h}^{\mathrm{D}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}), \tag{12}$$

$$\tilde{F}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2) = \left|\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\right| J^{\mathrm{B}}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2)\, \tilde{h}^{\mathrm{S}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, \tilde{h}^{\mathrm{D}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}). \tag{13}$$

We note that, since $\boldsymbol{p}_0^{\mathrm{S}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$ are determined by $\boldsymbol{\omega}^{\mathrm{B}}$ via Eqs. (8) and (9), they take different values in Eqs. (12) and (13) for the same primary sample $(u_1, \boldsymbol{u}_2)$.

During preprocessing, we apply our primary-sample-space guiding to both $\tilde{F}_{\mathrm{light}}$ and $\tilde{F}_{\mathrm{dir}}$, obtaining two probability densities p_{light} and p_{dir} that are approximately proportional to $\tilde{F}_{\mathrm{light}}$ and $\tilde{F}_{\mathrm{dir}}$, respectively. At render time, we draw two sets of primary samples $(u_1^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{light}})$ and $(u_1^{\mathrm{dir}}, \boldsymbol{u}_2^{\mathrm{dir}})$ with the probability densities p_{light} and p_{dir}, respectively. This gives us two unbiased estimators of the *boundary* integral:

$$\left\langle \frac{\tilde{F}_{\mathrm{light}}(u_1^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{light}})}{p_{\mathrm{light}}(u_1^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{light}})} \right\rangle, \qquad \left\langle \frac{\tilde{F}_{\mathrm{dir}}(u_1^{\mathrm{dir}}, \boldsymbol{u}_2^{\mathrm{dir}})}{p_{\mathrm{dir}}(u_1^{\mathrm{dir}}, \boldsymbol{u}_2^{\mathrm{dir}})} \right\rangle. \tag{14}$$

To obtain proper weights for these estimators, we rewrite the probability densities p_{light} and p_{dir} with respect to $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}^{\mathrm{B}}$ (as opposed to u_1 and $\boldsymbol{u}_2$) by letting

$$\hat{p}_{\mathrm{light}}(u_1, \boldsymbol{u}_2) := \frac{p_{\mathrm{light}}(u_1, \boldsymbol{u}_2)}{J^{\mathrm{B}}_{\mathrm{light}}(u_1, \boldsymbol{u}_2)}, \quad \hat{p}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2) := \frac{p_{\mathrm{dir}}(u_1, \boldsymbol{u}_2)}{J^{\mathrm{B}}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2)}. \tag{15}$$

Using the balance heuristic, this allows us to weight contributions of the two estimators in Eq. (14) with

$$\frac{\hat{p}_{\mathrm{light}}(\boldsymbol{u}^{\mathrm{light}})}{\hat{p}_{\mathrm{light}}(\boldsymbol{u}^{\mathrm{light}}) + \hat{p}_{\mathrm{dir}}(\boldsymbol{u}^{\mathrm{light}})}, \qquad \frac{\hat{p}_{\mathrm{dir}}(\boldsymbol{u}^{\mathrm{dir}})}{\hat{p}_{\mathrm{light}}(\boldsymbol{u}^{\mathrm{dir}}) + \hat{p}_{\mathrm{dir}}(\boldsymbol{u}^{\mathrm{dir}})}, \tag{16}$$

respectively, where $\boldsymbol{u}^{\mathrm{light}} := (u_1^{\mathrm{light}}, \boldsymbol{u}_2^{\mathrm{light}})$ and $\boldsymbol{u}^{\mathrm{dir}} := (u_1^{\mathrm{dir}}, \boldsymbol{u}_2^{\mathrm{dir}})$.

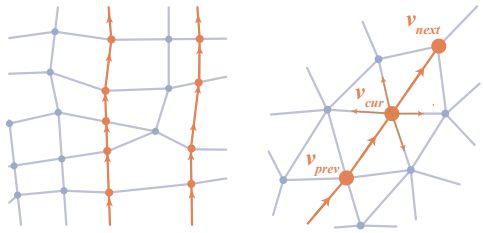


Fig. 4. **Edge sorting:** (a) Sorting the edges of a mesh allows us to create long "chains" (marked in red), improving the continuity of $\boldsymbol{x}^{\mathrm{B}}(u_1)$. (b) When extending a chain ending at the vertices $\boldsymbol{v}_{\mathrm{prev}}$ and $\boldsymbol{v}_{\mathrm{cur}}$, we pick the neighbor $\boldsymbol{v}_{\mathrm{next}}$ such that the direction of $(\boldsymbol{v}_{\mathrm{cur}}, \boldsymbol{v}_{\mathrm{next}})$ is as close as possible to that of $(\boldsymbol{v}_{\mathrm{prev}}, \boldsymbol{v}_{\mathrm{cur}})$.

In practice, evaluating the probability densities p_{light} and p_{dir} in Eq. (15) requires traversing the corresponding Kd-trees. To accelerate the computation, we exploit the fact that $\hat{p}_{\mathrm{light}}$ and $\hat{p}_{\mathrm{dir}}$ are approximately proportional to $\tilde{F}_{\mathrm{light}}$ and $\tilde{F}_{\mathrm{dir}}$, respectively. This allow us to write

$$\hat{p}_{\mathrm{light}}(u_1, \boldsymbol{u}_2) \approx \frac{\tilde{F}_{\mathrm{light}}(u_1, \boldsymbol{u}_2)}{c_{\mathrm{light}}\, J^{\mathrm{B}}_{\mathrm{light}}(u_1, \boldsymbol{u}_2)}, \quad \hat{p}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2) \approx \frac{\tilde{F}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2)}{c_{\mathrm{dir}}\, J^{\mathrm{B}}_{\mathrm{dir}}(u_1, \boldsymbol{u}_2)}, \tag{17}$$

where c_{light} and c_{dir} are normalization constants obtained when building the Kd-trees (by summing the probability masses stored in all leaf nodes). We note that, in Eq. (17), $\tilde{F}_{\mathrm{light}}$ and $\tilde{F}_{\mathrm{dir}}$ can be calculated, respectively, via Eqs. (12) and (13) in $O(1)$ time.

4.3 Edge Sorting

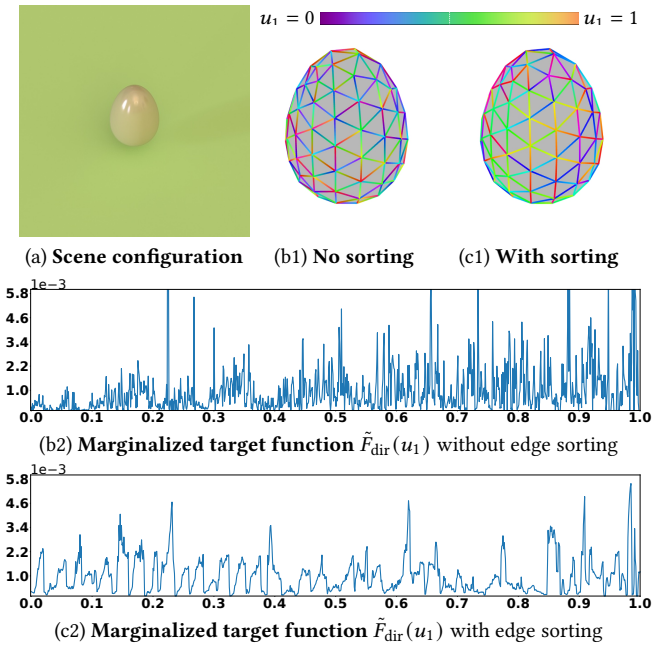
When the target functions $\tilde{F}$ of Eq. (11), $\tilde{F}_{\mathrm{light}}$ of Eqs. (12), and $\tilde{F}_{\mathrm{dir}}$ of Eqs. (13) are highly discontinuous, the performance of our technique can degrade as the Kd-trees need to have very large depths to properly approximate the targets. In practice, this is mostly caused by discontinuities of $\boldsymbol{x}^{\mathrm{B}}$ as a function of the primary sample u_1.

To address this problem, we introduce an optional step that can greatly reduce the number of jump discontinuity points of $\boldsymbol{x}^{\mathrm{B}}(u_1)$. Specifically, we sort the edges to form long "chains" along which the mapping from u_1 to $\boldsymbol{x}^{\mathrm{B}}$ is continuous (see Figure 4-a). Since finding the optimal edge ordering that results in a minimal number of chains is NP-hard, we instead traverse the mesh edges in a greedy fashion. As described in Algorithm 2, starting from an arbitrary mesh edge, we construct a chain of edges by iteratively moving to a neighboring vertex $\boldsymbol{v}_{\mathrm{next}}$ from the current one $\boldsymbol{v}_{\mathrm{cur}}$. When there are multiple possible neighbors, we pick the one such that the direction of the edge $(\boldsymbol{v}_{\mathrm{cur}}, \boldsymbol{v}_{\mathrm{next}})$ is as close to that of the previous one $(\boldsymbol{v}_{\mathrm{prev}}, \boldsymbol{v}_{\mathrm{cur}})$ as possible (see Figure 4-b). We demonstrate the usefulness of our edge sorting process using a simple scene in Figure 5 and will show more ablations in §5.

For applications where remeshing can be applied (which is typically the case when solving inverse-rendering problems), we pre-apply remeshing [Jakob et al. 2015] to make the input mesh field-aligned. This allows our edge sorting process to output fewer longer chains, further improving the overall effectiveness of our technique.

ALGORITHM 2: Construct a chain of mesh edges

```
BuildChain(M, v_0)
  Input: A polygonal mesh M and a mesh edge (v_0, v_1)
begin
    chain ← [(v_0, v_1)] ;                    // The resulting chain
    v_prev ← v_0; v_cur ← v_1;
    while v_cur is associated with some unused edge do
        Select an unused edge (v_cur, v_next) with a direction as close
          to that of (v_prev, v_cur) as possible;
        Append the edge (v_cur, v_next) to the end of chain;
        v_prev ← v_cur; v_cur ← v_next;
    end
    return chain;
end
```



(a) **Scene configuration** (b1) **No sorting** (c1) **With sorting**

(b2) **Marginalized target function** $\tilde{F}_{\mathrm{dir}}(u_1)$ without edge sorting

(c2) **Marginalized target function** $\tilde{F}_{\mathrm{dir}}(u_1)$ with edge sorting

Fig. 5. We demonstrate the **effectiveness of our edge sorting** using a simple scene containing a glossy egg under environmental illumination (a). Without edge sorting, the mapping from the primary sample u_1 to points on mesh edges can be highly discontinuous (b1), causing the target functions to have many jump discontinuities (b2). With edge sorting, on the other hand, the mapping becomes significantly more continuous (c1), causing the target functions to contain fewer discontinuities (c2).

5 RESULTS

In what follows, we validate our technique and evaluate its effectiveness by comparing gradient images (§5.1). Then, we compare the performance of our technique and baseline methods using several inverse-rendering examples (§5.2) and show additional results (§5.3).

Please refer to the supplemental material for more examples and animated inverse-rendering results.

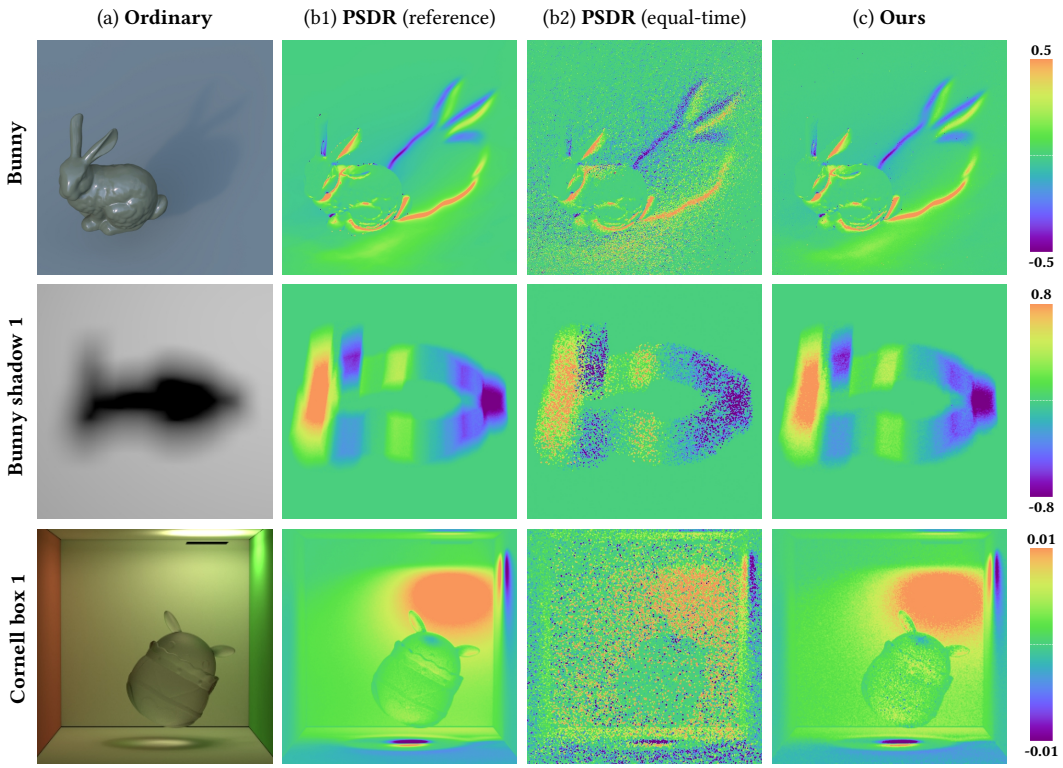


Fig. 6. **Differentiable-rendering comparisons:** We compare *boundary*-component-only gradient images generated with our technique (c) with Zhang et al.'s method [2020] indicated as "PSDR" (b1, b2). At equal-time, our technique produces significantly cleaner gradient estimates (b2, c).

5.1 Validation and Evaluation

Validation and comparison with Zhang et al. We evaluate our technique by comparing gradient images generated with our method and with Zhang et al.'s approach [2020] which performs simple guiding using regular grids. Since our technique focuses on estimating the *boundary* component of Eq. (3), we estimate this term only (i.e., by neglecting the *interior* integral) for both methods.

We show three examples in Figure 6. The **bunny** scene contains a glossy bunny under environmental illumination, and the gradient images are computed with respect to the rotation angle of the bunny (around the vertical axis). The **bunny shadow 1** scene involves a bunny lit by a large area light, casting a soft shadow on the ground. The gradients are computed with respect to the translation of the bunny. The **Cornell box 1** scene has a rough-glass object and an area light pointing toward the ceiling, causing most of the scene to be indirectly lit. In all examples, our results closely match the references (obtained using Zhang et al.'s method [2020] with very high sample counts). At equal-time—and approximately equal sample and storage—our results enjoy much lower variances.

Comparison with warped-area sampling. Additionally, in Figure 7 and the supplement, we compare gradients estimated using our method and warped-area sampling [Bangaru et al. 2020]. This technique is largely orthogonal to ours as it handles visibility discontinuities very differently (by turning the *boundary* integral into an *interior* one using the divergence theorem). Nonetheless, we compare with this technique for completeness. Since warped-area sampling uses a different formulation than ours, we show full gradient estimates—as opposed to *boundary*-component-only estimates—where our results have the *interior* components computed using Zhang et al.'s method [2020].

In Figure 7, the **bunny shadow 1** scene is identical to the one used in Figure 6, and the **Cornell box 2** scene is a simplified version of **Cornell box 1** with a diffuse object[2] and the area light pointing downward. When using equal numbers of samples, our technique is significantly faster while producing gradient estimates with lower noise. At equal time, our method offers even greater advantage in result quality.

Ablation: differentiable rendering. Lastly, we conduct an ablation study using the **bunny** scene to evaluate the effectiveness of our edge sorting and remeshing (§4.3) as well as multiple important sampling (§4.2). This scene uses outdoor environmental lighting with a bright sun, creating a clear shadow on the ground (as shown in the top row of Figure 6-a).

When performing light sampling, the area containing the sun will be selected with a very high probability. This strategy works well for boundary segments with $\boldsymbol{x}^{\mathrm{B}}$ located on regions that are

[2]We use a diffuse object in the **Cornell box 2** scene because the implementation of warped-area sampling [Bangaru et al. 2020] released by the authors does not support the rough dielectric BSDF.

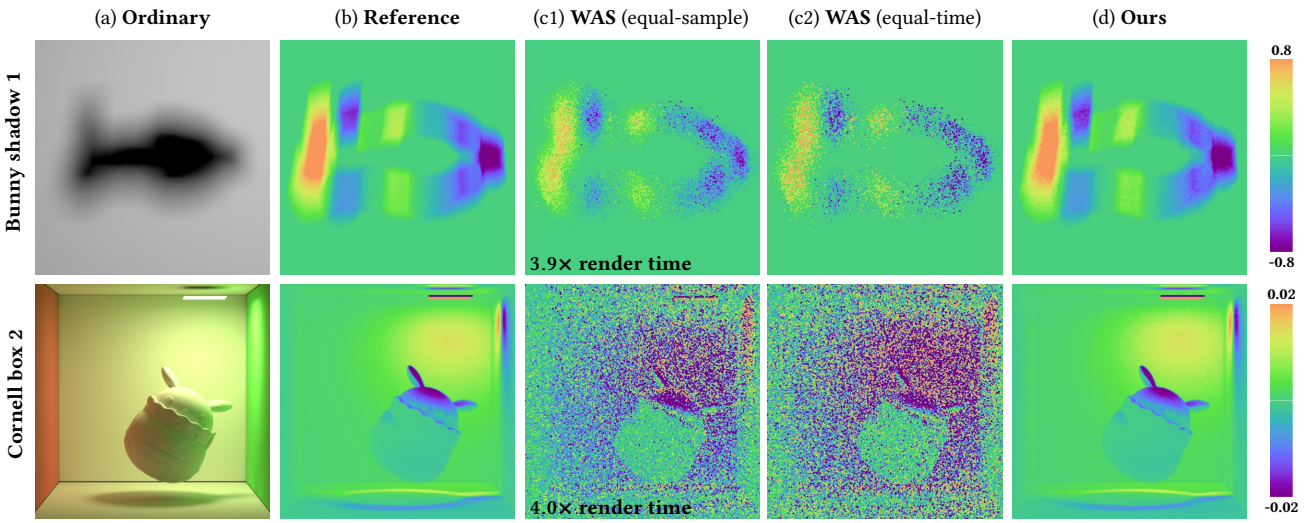


Fig. 7. **Differentiable-rendering comparisons:** We compare gradient images generated with our technique and the warped-area sampling (WAS) method [Bangaru et al. 2020]. With equal sample counts, our technique produces cleaner results while being significantly faster. At equal-time, the quality differences become even greater.

visible to the sun and produces clean gradient estimates around the shadow boundaries. On the other hand, for boundary segments with $\boldsymbol{x}^{\mathrm{B}}$ occluded from the sun, light sampling is a poor strategy—as illustrated in Figure 8—and causes high variance around the bunny. Even with Zhang et al.'s (Figure 8-b) and our primary-sample-space guiding (Figure 8-c1), the results remain noisy as the support of the target function $\tilde{F}_{\text{light}}$ of Eq. (12) is very narrow, making it difficult to detect the high-value regions of $\tilde{F}_{\text{light}}$.

When using direction sampling with our guiding, the result is much cleaner overall since the sampled directions of boundary segments are no longer affected by the position of the sun (Figure 8-c2). On the other hand, estimated gradients around the shadow boundaries become slightly worse.

Our multiple importance sampling technique (§4.2) combines light- and direction-sampling strategies and, thus, enjoys the advantages of both (Figure 8-d).

With edge sorting (§4.3), Zhang et al.'s method—which does not adapt to the structure of the target function—has hardly any improvement in result quality. In contrast, our method benefits from improved continuity of the target functions and produces cleaner results, as shown in the bottom row of Figure 8.

5.2 Inverse-Rendering Comparisons

We now demonstrate the effectiveness of our technique using several inverse-rendering examples using synthetic input images. Since our technique focuses on estimating the *boundary* integral, we use Zhang et al.'s method [2020] to estimate the *interior* term of Eq. (3). For all inverse-rendering optimizations, we use Nicolet et al.'s method [2021] to update mesh vertex positions (provided the estimated gradients). Additionally, we utilize mini-batch gradient descent by only rendering a subset of input images per iteration. Please refer to Table 2 for optimization configurations and performance statistics.

Ablation: inverse rendering. As a continuation of the ablation study (Figure 8), we show two inverse-rendering comparisons in Figure 9. Both the **jumpy dumpty** (top) and the **Klee** (bottom) scenes are comprised of glossy objects under environmental illuminations with cast shadows on the ground. Using multiple target images under identical illumination but varying viewing directions (with one shown in the figure), we solve for the object shapes by minimizing image L_1 losses. Initialized with a sphere (Figure 9-a) and using identical settings (e.g., learning rates and numbers of iterations), we configure three optimizations with gradients estimated using our primary-sample-space guiding with light sampling, direction sampling, and multiple importance sampling (MIS), respectively (Figure 9-cd). Further, we ran these optimizations with and without our edge sorting method, respectively. The results demonstrate that, both our MIS and edge sampling processes yield reduced variance in gradient estimates, allowing the optimization to converge faster and produce more accurate reconstructions.

Further, we compare the performance of inverse-rendering optimization using gradients estimated with our (full) technique and with Zhang et al.'s method [2020]. Figure 10 shows three examples. The **kirby** scene contains a rough-glass object under area illumination; The **bunny in glass** scene consists of a diffuse bunny enclosed by a cube made of rough glass. Taking as input 35 images for the first scene and 50 for the second, we solve for the shapes of these objects to minimize image L_1 losses. Since our technique has identical *interior* estimates and offers lower variance *boundary* integral estimates, gradients obtained with our method result in faster convergence.

Additionally, the **bunny shadow 2** scene contains a bunny (outside the camera's field of view) under environmental illumination, casting a shadow on the ground. Taking as input 70 shadow images

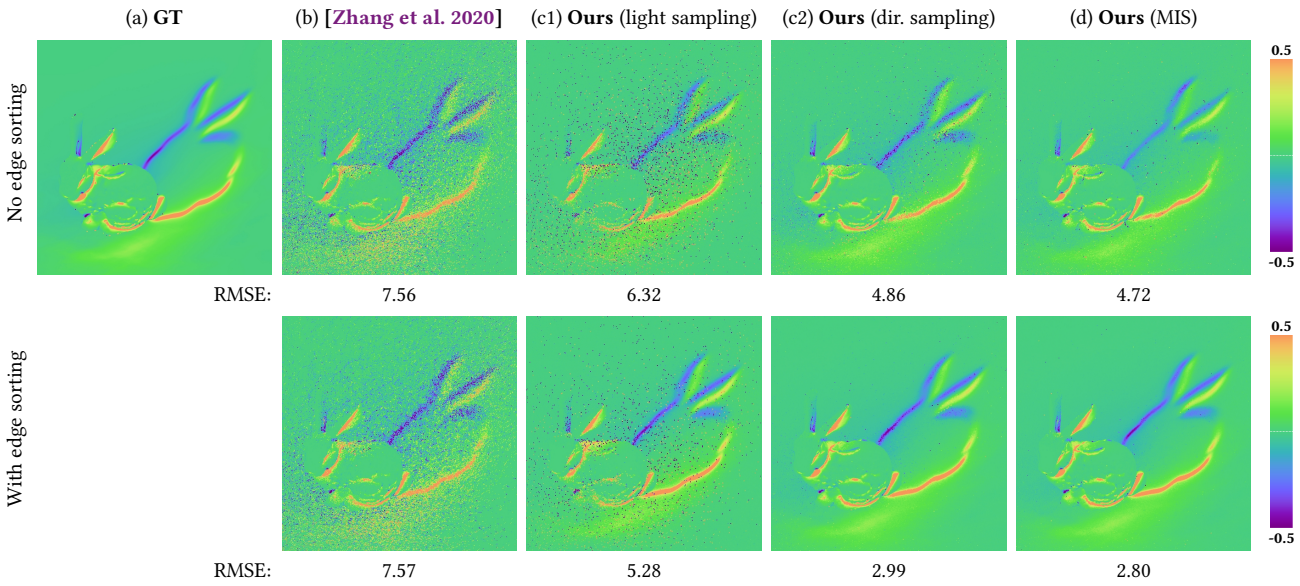


Fig. 8. **Ablation study (differentiable rendering):** We evaluate the effectiveness of our edge-sorting (§4.3) and multiple-importance-sampling (§4.2) steps. All results in columns (b–d) are generated in equal time.

Table 2. Optimization configuration and performance statistics for our inverse-rendering results. The "guiding time" numbers indicate per-iteration computation time for building the Kd-trees; the "render time" numbers account for the differentiable rendering time; and "postproc. time" captures the cost for updating mesh vertices (using Nicolet et al.'s method [2021]). All experiments are conducted on a workstation with an AMD Ryzen Threadripper Pro 3975WX CPU (with 32 cores) and an Nvidia RTX Titan graphics card.

Scene		# Target images	# Param.	Batch size	# Iter.	Guiding memory	Guiding time	Render time	Postproc. time
Dodoco	(Fig. 1)	70	450,000	2	3000	4 MB	1.34 s	2.59 s	0.310 s
Jumpy Dumpty	(Fig. 9)	140	60,000	2	520	2 MB	0.61 s	1.11 s	0.016 s
Klee	(Fig. 9)	40	60,000	1	200	2 MB	0.33 s	0.82 s	0.016 s
Kirby	(Fig. 10)	35	15,000	1	400	80 KB	0.91 s	3.67 s	0.017 s
Bunny in glass	(Fig. 10)	50	30,000	2	2000	128 KB	1.79 s	17.55 s	0.022 s
Bunny shadow 2	(Fig. 10)	70	30,000	2	600	2 MB	0.47 s	0.72 s	0.012 s
Duck	(Fig. 11)	35	3,205,728	1	400	2 MB	0.34 s	0.94 s	0.092 s
Mora	(Fig. 11)	70	150,001	1	1600	4 MB	0.43 s	1.01 s	0.109 s
Glass Dodoco	(Fig. 11)	140	60,000	1	700	400 KB	0.94 s	5.76 s	0.093 s

with the object having varying known orientations, we solve for the shape of the object. Since the gradients around shadow boundaries are captured solely by the *boundary* integral, the quality of its estimates is crucial to the performance. Compared with Zhang et al.'s method, our gradient estimates have significantly lower noise, yielding much better reconstruction of object geometry. We note that the object's geometric details are difficult to recover due to the non-line-of-sight configuration.

5.3 Additional Inverse-Rendering Results

We show additional inverse-rendering results in Figure 11. The **Duck** scene [Dong et al. 2014] contains a textured duck model under environmental illumination. Using 35 target images of the object with different viewing directions, we jointly optimize the shape of the object and its spatially varying albedo. The **Mora** scene involves a glossy coin under environmental lighting. Using 70 target images of the object under multiple views, we jointly optimize the shape of the object and the global rotation angle of the environment map (around the vertical axis). The **glass Dodoco** scene consists of a rough-glass object under area lighting. Taking as input 140 multi-view images, we optimize the shape of the object.

The *boundary* component of Eq. (3) plays a significant role in all these examples. Our technique successfully provides low-variance gradient estimates, allowing all the inverse-rendering optimizations to converge smoothly. Please refer to the supplement for animated versions of these results.

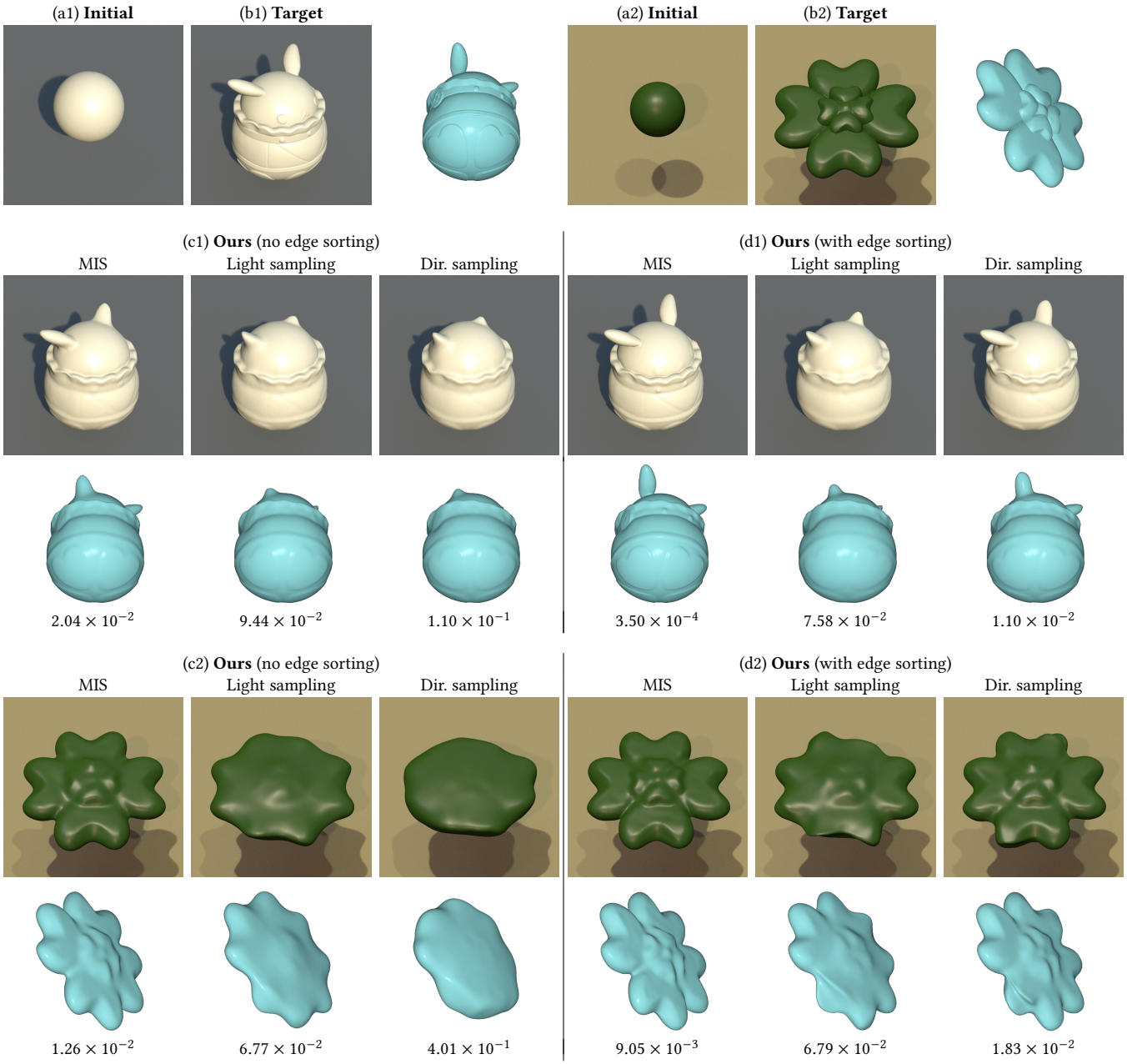


Fig. 9. **Ablation study (inverse rendering):** We evaluate the effectiveness of our edge sorting and multiple-importance-sampling (MIS) processes by comparing inverse-rendering results. On the bottom of (a3), (a4), (b3) and (b4), we visualize the reconstructed models and show the corresponding Chamfer distances [Barrow et al. 1977] to the groundtruth geometries (normalized so that the GT has a unit bounding box). Using identical target images (one of which is shown for each example), losses, initializations, and optimization configurations (including learning rates and numbers of iterations), our edge sorting and multiple importance sampling can both lead to more accurate reconstructions.

6 DISCUSSION AND CONCLUSION

Limitations and future work. Our technique focuses on the surface-only light transport. For scenes with participating media, as demonstrated by Zhang et al. [2021b], the dimensionality of the primary-sample space for constructing boundary segments increases from three to five, making the guiding much more challenging. Generalizing our technique to support volumetric light transport is an important future topic.

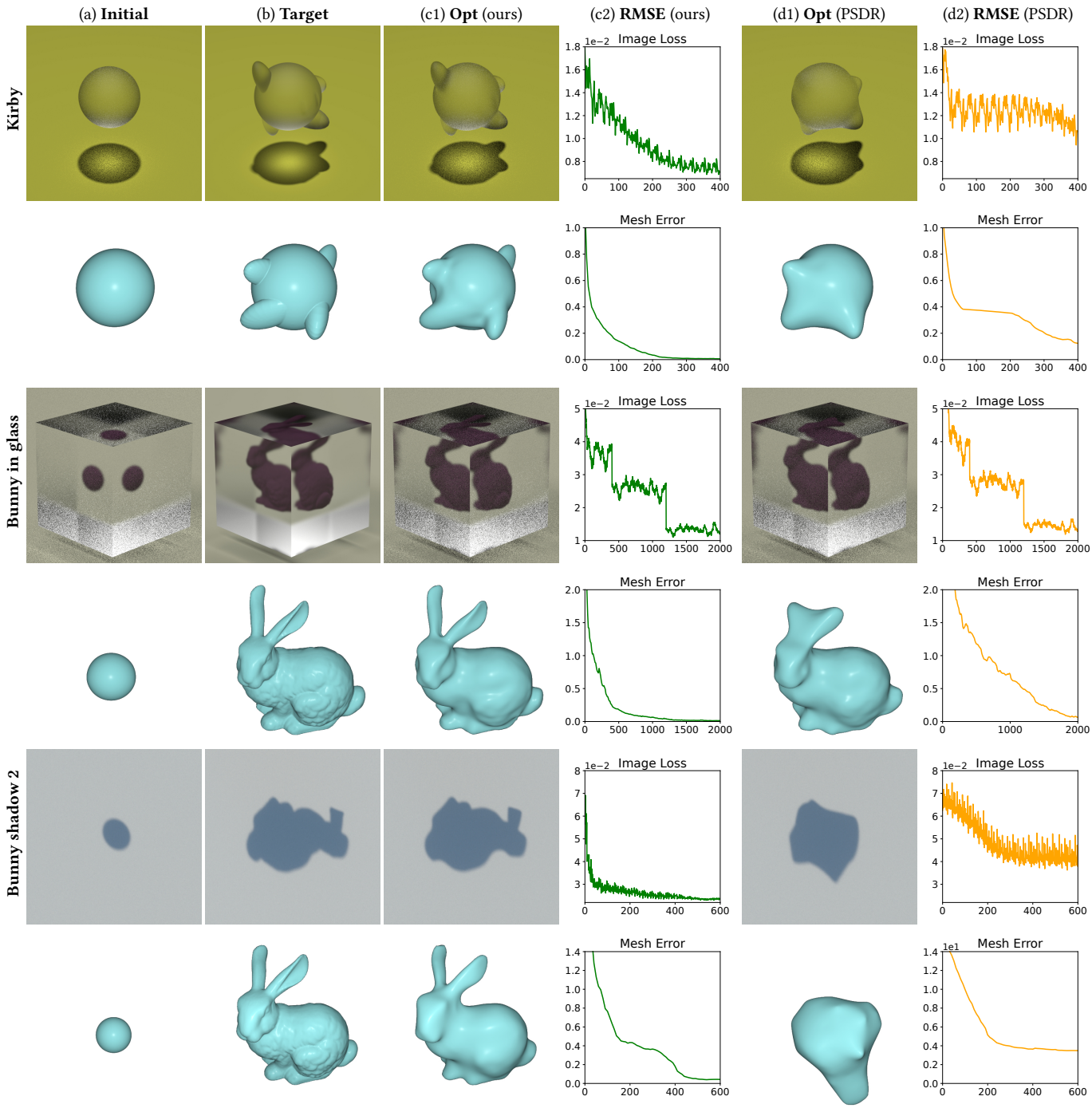


Fig. 10. **Inverse-rendering comparison:** We solve inverse-rendering problems using gradients estimated using our technique and PSDR [Zhang et al. 2020]. For each example, we use identical target images (with one shown), losses, and optimization settings (e.g., initial states, optimizers, and learning rates). At equal-time, our technique has allowed significantly faster convergence for all examples. The plotted mesh error captures the Chamfer distance [Barrow et al. 1977] between the reconstructed and groundtruth geometries (normalized so that the GT has a unit bounding box). We use this information only for evaluation, and not for optimization.

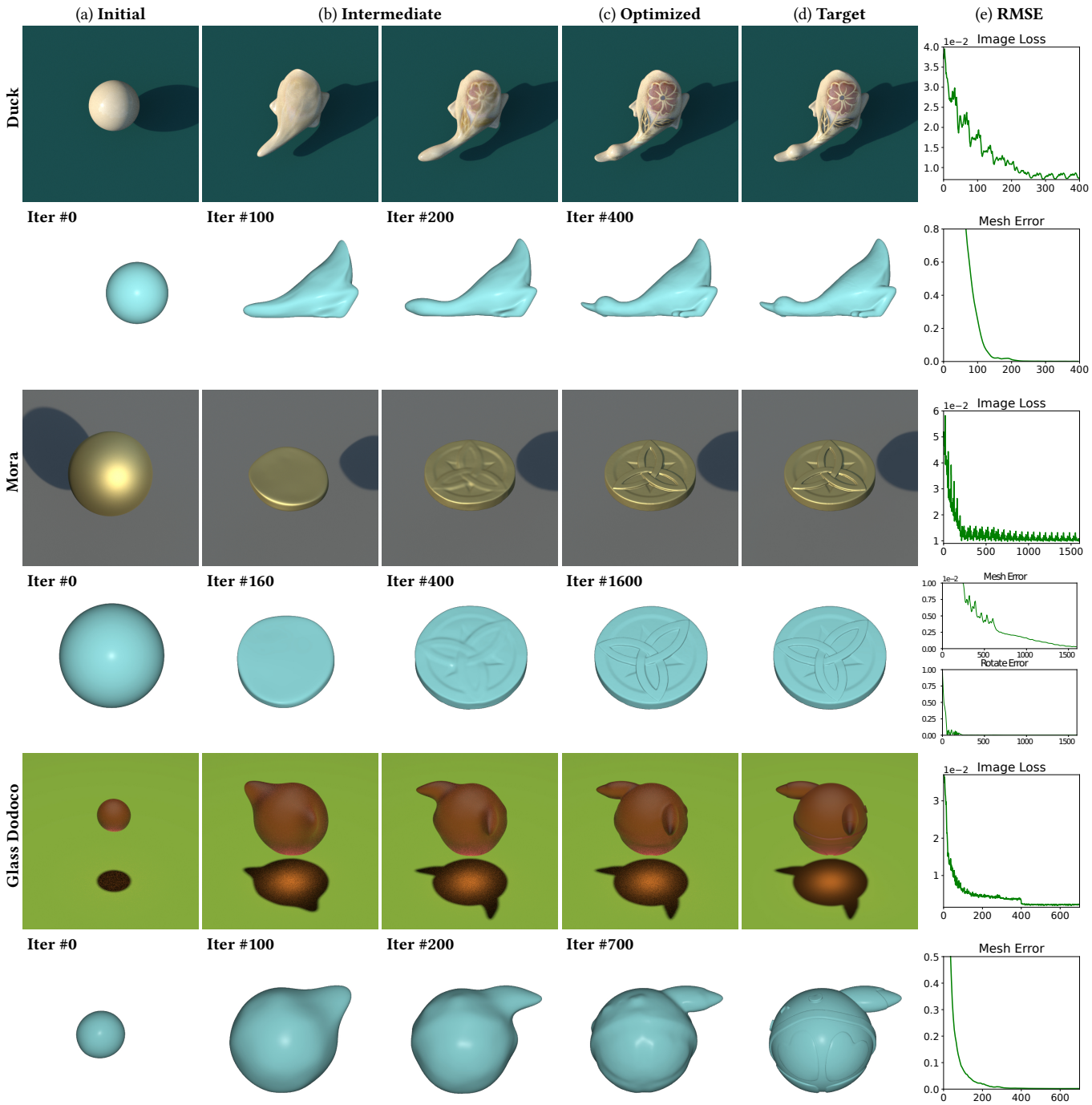


Fig. 11. Additional **inverse-rendering results** generated with gradient estimates obtained using our method. The mesh error [Jensen et al. 2014] is used for evaluation only, and not for optimization.

Additionally, our technique focuses on constructing boundary segments, so improving the sampling of source and detector subpaths may further benefit the performance of our technique.

Lastly, combining our method with Markov-Chain Monte Carlo (MCMC) sampling may be an interesting topic for future exploration.

Conclusion. We introduced a new technique to effectively estimate *boundary* path integrals. A core component of our technique is a primary-sample-space guiding algorithm that adapts to the structures of the integrand of *boundary* path integrals and allows efficient importance sampling of boundary segments. We also showed how multiple importance sampling can be applied to combine two guiding strategies. Additionally, we introduced an optional edge sorting step to further improve the performance of our technique.

We evaluated the effectiveness of our technique by comparing with existing methods using several differentiable-rendering and inverse-rendering experiments.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments and suggestions. Thanks to the Stanford University Computer Graphics Laboratory for the Bunny model. This work started when Kai Yan was an intern at Meta Reality Labs Research. Kai Yan's contributions while at the University of California, Irvine were partially supported by NSF grant 1900927.

REFERENCES

Michael Ashikhmin and Peter Shirley. 2000. An anisotropic phong BRDF model. *Journal of graphics tools* 5, 2 (2000), 25–32.

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.

Harry G Barrow, Jay M Tenenbaum, Robert C Bolles, and Helen C Wolf. 1977. *Parametric correspondence and chamfer matching: Two new techniques for image matching.* Technical Report. SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER.

Robert L Cook and Kenneth E. Torrance. 1982. A reflectance model for computer graphics. *ACM Transactions on Graphics (ToG)* 1, 1 (1982), 7–24.

Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. 2014. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Trans. Graph.* 33, 6 (2014), 1–12.

Jerry Guo, Pablo Bauszat, Jacco Bikker, and Elmar Eisemann. 2018. Primary sample space path guiding. In *Eurographics Symposium on Rendering*, Vol. 2018. 73–82.

Eric Heitz and Eugene d'Eon. 2014. Importance sampling microfacet-based BSDFs using the distribution of visible normals. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 103–112.

Eric Heitz, Johannes Hanika, Eugene d'Eon, and Carsten Dachsbacher. 2016. Multiple-scattering microfacet BSDFs with the Smith model. *ACM Trans. Graph.* 35, 4 (2016), 58:1–58:14.

Wenzel Jakob, Marco Tarini, Daniele Panozzo, and Olga Sorkine-Hornung. 2015. Instant Field-Aligned Meshes. *ACM Trans. Graph.* 34, 6 (2015), 189:1–189:15.

Henrik Wann Jensen. 1995. Importance driven path tracing using the photon map. In *Eurographics Workshop on Rendering Techniques*. Springer, 326–335.

Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. 2014. Large scale multi-view stereopsis evaluation. In *CVPR*. 406–413.

Csaba Kelemen and Laszlo Szirmay-Kalos. 2001. A microfacet based coupled specular-matte BRDF model with importance sampling. In *Eurographics short presentations*, Vol. 2. 4.

Eric P Lafortune and Yves D Willems. 1995. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Eurographics Workshop on Rendering Techniques*. Springer, 11–20.

Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. 2020. Modular Primitives for High-Performance Differentiable Rendering. *ACM Trans. Graph.* 39, 6 (2020), 194:1–194:14.

Joo Ho Lee, Adrian Jarabo, Daniel S. Jeon, Diego Gutierrez, and Min H. Kim. 2018. Practical multiple scattering for rough surfaces. *ACM Trans. Graph.* 37, 6 (2018), 275:1–275:12.

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.

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

Thomas Müller, Markus Gross, and Jan Novák. 2017. Practical path guiding for efficient light-transport simulation. In *Computer Graphics Forum*, Vol. 36. 91–100.

Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. 2019. Neural Importance Sampling. *ACM Trans. Graph.* 38, 5 (2019), 145:1–145:19.

Baptiste Nicolet, Alec Jacobson, and Wenzel Jakob. 2021. Large Steps in Inverse Rendering of Geometry. *ACM Trans. Graph.* 40, 6 (2021), 248:1–248:13.

Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. 2019. Mitsuba 2: a retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)* 38, 6 (2019), 203.

Michael Oren and Shree K Nayar. 1994. Generalization of Lambert's reflectance model. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*. 239–246.

Bui Tuong Phong. 1975. Illumination for computer generated pictures. *Commun. ACM* 18, 6 (1975), 311–317.

Sylvia C Pont and Jan J Koenderink. 2002. Bidirectional reflectance distribution function of specular surfaces with hemispherical pits. *JOSA A* 19, 12 (2002), 2456–2466.

Christophe Schlick. 1994. An inexpensive BRDF model for physically-based rendering. In *Computer graphics forum*, Vol. 13. Wiley Online Library, 233–246.

Bram van Ginneken, Marigo Stavridi, and Jan J Koenderink. 1998. Diffuse and specular reflectance from rough surfaces. *Applied optics* 37, 1 (1998), 130–139.

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

Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. 2014. On-Line Learning of Parametric Mixture Models for Light Transport Simulation. *ACM Trans. Graph.* 33, 4 (2014), 101:1–101:11.

Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. 2007. Microfacet models for refraction through rough surfaces. *Rendering techniques* 2007 (2007), 18th.

Gregory J Ward. 1992. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*. 265–272.

Feng Xie and Pat Hanrahan. 2018. Multiple scattering from distributions of specular V-grooves. *ACM Trans. Graph.* 37, 6 (2018), 276:1–2767:14.

Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. 2021. Monte Carlo Estimators for Differential Light Transport. *ACM Trans. Graph.* 40, 4 (2021), 78:1–78:16.

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

C. Zhang, B. Miller, K. Yan, I. Gkioulekas, and S. Zhao. 2020. Path-Space Differentiable Rendering. *ACM Transactions on Graphics (SIGGRAPH 2020)* 39, 4 (2020), 143:1–143:19.

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

Cheng Zhang, Zihan Yu, and Shuang Zhao. 2021b. Path-Space Differentiable Rendering of Participating Media. *ACM Trans. Graph.* 40, 4 (2021), 76:1–76:15.

Quan Zheng and Matthias Zwicker. 2019. Learning to importance sample in primary sample space. In *Computer Graphics Forum*, Vol. 38. 169–179.

Yang Zhou, Lifan Wu, Ravi Ramamoorthi, and Ling-Qi Yan. 2021. Vectorization for Fast, Analytic, and Differentiable Visibility. *ACM Trans. Graph.* 40, 3 (2021), 27:1–27:21.