Efficient Computational Differentiation for Path-Space Differentiable Rendering

ZIHAN YU, CHENG ZHANG, DEREK NOWROUZEZAHRAI, ZHAO DONG, and SHUANG ZHAO

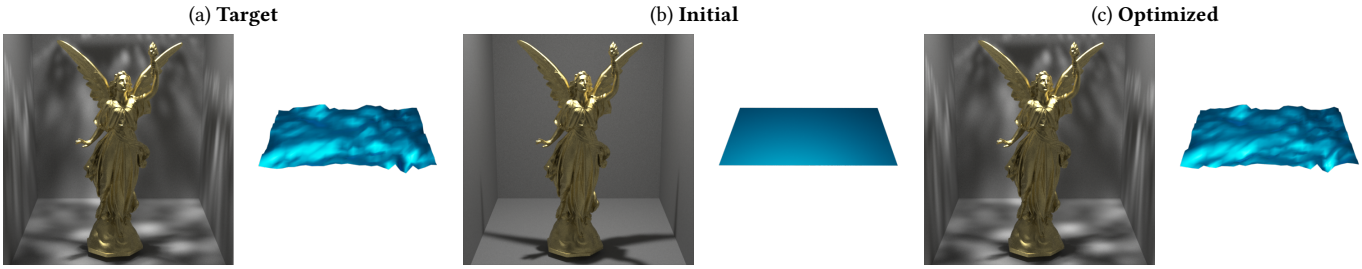

Fig. 1. We introduce a new, efficient and scalable computational differentiation technique for path-space differentiable rendering. We formulate image-loss gradients – an essential ingredient for solving inverse-rendering optimizations – as differential path integrals, before introducing new algorithms to efficiently estimate these path integrals. This example involves a underwater scene with a glossy Lucy model. We optimize the shape of the air-water interface (described using 651 mesh parameters) to match the caustics patterns. Our system improves on the generality and performance for solving inverse-rendering problems that involve both complex light transport effects and large numbers of parameters. (Using Adobe Acrobat and clicking on column (c) will play an animation of our optimization.)

Efficient differentiation is crucial for the practicality of differentiable rendering systems. Unfortunately, due to the complex and irregular computations involved in physics-based rendering, the differentiation process can lead to very large computation graphs—even with widely adopted and highly optimized automatic differentiation (AD) libraries.

In this paper, we introduce a new formulation that expresses image-loss gradients as differential path integrals and fully decouples path sampling and computational differentiation. Based on this formulation, we develop new algorithms that offer the flexibility of using general light sampling techniques (such as bidirectional path tracing) and handling computational differentiation without relying on adjoint simulations. Additionally, we develop a new differentiable renderer capable of solving challenging inverse-rendering problems with large numbers of parameters, geometric changes, and complex light transport effects like caustics. We demonstrate the effectiveness of our technique using several synthetic examples.

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

1 Introduction

Physics-based *forward rendering* methods simulate the flow of light to numerically estimate the responses of radiometric detectors given fully described virtual scenes. In contrast, physics-based *differentiable rendering* treat the problem of estimating *gradients* of detector responses with respect to scene parameters. Differentiable rendering enables (i) gradient-based *inverse-rendering* optimization and (ii) the integration of forward rendering into machine learning and probabilistic inference pipelines, and so has a wide array of applications in graphics, vision, computational imaging and computational fabrication.

Many recent works address physics-based differentiable rendering. Li et al. [2018] introduced the first general-purpose demonstration of the differentiability (in principle) of forward rendering. Their method proposes Monte Carlo edge sampling to handle geometric discontinuities which are essential for differentiation with respect to scene geometries. This was later generalized by Zhang et al. [2019; 2020; 2021b] to handle volumetric light transport, before proposing a unified formulation of the differential path integral. Here, similar to the path integral formulation for forward rendering [Veach 1997; Pauly et al. 2000], the differential path integral formulation has enabled the development of sophisticated differentiable rendering algorithms.

Computational differentiation is another key technology in differentiable rendering. Previously, differentiable renderers typically relied on general-purpose automatic differentiation (AD) frameworks such as PyTorch [Paszke et al. 2019] and Enoki [Jakob 2019] to differentiate light path contributions. Unfortunately, since the computations involved in differentiable rendering are usually complex and irregular, the computational differentiation process can lead to very large computation graphs that are prohibitively expensive to evaluate and store—even with widely-adopted, highly optimized reverse-mode AD libraries.

To address this problem, several recent methods formulate image-loss gradients as solutions to an adjoint transport problem [Nimier-David et al. 2020; Vicini et al. 2021] . This avoids the need to store entire computation graphs and permit the "local" application of AD (e.g., when differentiating BSDF evaluations), significantly improving the overall system performance. The design of these techniques is, however, a bi-product of a differentiable unidirectional path tracing formulations. As a result, they are not readily generalizable to other transport paradigms, such as bidirectional methods.

We introduce a new formulation of gradients of image losses as differential path integrals, and an efficient algorithm for computing these gradients. Our formulation completely decouples path sampling and computational differentiation, allowing the former to be handled using general light sampling techniques (i.e., beyond

Authors' Contact Information: Zihan Yu; Cheng Zhang; Derek Nowrouzezahrai; Zhao Dong; Shuang Zhao.

unidirectional path tracing) and the latter to be treated without relying on any adjoint simulation.

From a theoretical perspective, previous techniques [Nimier-David et al. 2020; Vicini et al. 2021] can be treated as special-case instance of our formulation and method—akin to how unidirectional path tracing can be considered as a specific sampling strategy for the path integral formulation of light transport. Concretely, our contributions are:

- a new mathematical framework of image-loss gradients as differential path integrals involving *interior* and a *boundary* components (§4);
- highly-efficient computational differentiation algorithms to compute the *interior* component, exploiting the layered structure of the global computation graph (§5.1);
- a reverse-mode treatment of the *boundary* component computation under our unified framework (§5.2); and,
- a new CPU-based system with efficiency rivaling state-of-the-art GPU-based solvers (§6.1).

We validate our technique and system using several differentiable-rendering and inverse-rendering experiments.

2 Related Work

Path-space rendering. Veach [1997] introduced the formulation of *path integrals* by recursively expanding the rendering equation [Kajiya 1986]. This formulation was later extended to handle volumetric light transport by Pauly et al. [2000].

The path-integral formulation expresses radiometric measurements as high-dimensional integrals, enabling the development of a wide array of new Monte Carlo estimators (e.g., [Veach and Guibas 1995, 1997; Jakob and Marschner 2012]) that are capable of efficiently simulating challenging effects such as indirect illumination and near-specular transport.

Physics-based differentiable rendering. Physics-based differentiable rendering is concerned with numerically computing derivatives of forward-rendering results with respect to arbitrary scene parameters (such as object geometries and optical material properties). In general, physics-based differentiable rendering requires estimating *interior* integrals given by differentiating forward-rendering integrands as well as *boundary* integrals defined over discontinuity boundaries of those integrands.

Previously, the *interior* integrals have been mostly estimated by adopting Monte Carlo methods developed for forward rendering. Recently, new sampling techniques specifically designed for differentiable rendering have been introduced [Zeltner et al. 2021; Zhang et al. 2021a]. Since our formulation completely decouples path sampling and computational differentiation, our technique is largely orthogonal to these methods.

The *boundary* integrals are unique to differentiable rendering. It has been shown that the integrals in this special case can be handled 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].

Recently, 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). Our technique is built upon this formulation but with a focus on efficient computational differentiation.

Computational differentiation. Automatic differentiation (AD) allows the derivative of a function specified by a computer program to be evaluated numerically. These techniques have been widely used in machine learning and statistical inference [Griewank and Walther 2008; Wengert 1964; McClelland et al. 1986] to obtain gradients of complex functions (e.g., neural networks). Recently, several general-purpose AD frameworks such as TensorFlow [Abadi et al. 2016], PyTorch [Paszke et al. 2019], and Enoki [Jakob 2019] have been developed. Further, systematic handling of discontinuities—which has been neglected by most AD frameworks—has been explored [Bangaru et al. 2021].

Most general-purpose differentiable rendering techniques, including ours, utilize automated differentiation. Theoretically, our main theory and algorithms (§4, §5) are orthogonal to the choice of automated differentiation method. In practice, we utilize the Enzyme AD framework [Moses and Churavy 2020] to develop our new differentiable rendering system (§6.1).

3 Preliminaries

We now briefly review mathematical preliminaries of the path integral (§3.1) and differential path integral (§3.2) formulations.

3.1 Path integral

A key mathematical formulation that has enabled the development of many advanced rendering algorithms (such as bidirectional path tracing) is the **path integral** [Veach 1997; Pauly et al. 2000]. Under this formulation, the response $I \in \mathbb{R}$ of a radiometric detector is expressed as an integral of the form:

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

where $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denotes a **light transport path** with vertices $\boldsymbol{x}_n \in \mathcal{M}$ for each $0 \le n \le N$ (such that $\boldsymbol{x}_0$ lies on a light source and $\boldsymbol{x}_N$ on the detector); Ω is the **path space**; f is the **measurement contribution function**; and μ is the Lebesgue measure on Ω. In case of surface-only light transport, for instance, $\Omega = \cup_{N=1}^{\infty} \mathcal{M}^{N+1}$ where $\mathcal{M}$ is the union of all object surfaces, and μ is the area-product measure (i.e., $\mathrm{d}\mu(\bar{\boldsymbol{x}}) = \prod_{n=0}^{N} \mathrm{d}A(\boldsymbol{x}_n)$ with surface-area measure A).

Physics-based rendering typically involves estimating multiple, say m_I, response values (e.g., one per pixel). To this end, we make the measurement contribution function to be vector-valued, denoted as $\boldsymbol{f}$. Then, Eq. (1) can be rewritten in a vector-valued form as

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

where $\boldsymbol{I}$, $\boldsymbol{f}(\bar{\boldsymbol{x}})$ are m_I-dimensional (column, i.e., $\mathbb{R}^{m_I \times 1}$) vectors.

3.2 Differential path integral

Recently, to facilitate the development of advanced Monte Carlo differentiable rendering techniques, Zhang et al. [2020; 2021b] have introduced the formulation of *differential path integral* that gives the derivatives of path integrals of Eq. (1) with respect to arbitrary scene parameters $\theta \in \mathbb{R}$.

Material-form reparameterization. When the scene geometry $\mathcal{M}$ depends on the parameter θ, so will the path space Ω. This makes the differentiation of Eq. (1) more complicated. To address this issue, Zhang et al. have introduced a **material-form** reparameterization to path integrals, which works by applying a change of variable to Eq. (1). Specifically, let $\mathcal{B}$ be some **reference configuration** that is independent of the scene parameter θ and $\mathsf{X}(\cdot, \theta)$ be a differentiable one-to-one mapping from $\mathcal{B}$ to $\mathcal{M}(\theta)$ for any θ. Then, $\mathsf{X}(\cdot, \theta)$ induces another one-to-one mapping $\bar{\mathsf{X}}(\cdot, \theta)$ from **material light paths** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ to ordinary light paths $\bar{\mathsf{X}}(\bar{\boldsymbol{p}}, \theta) = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N) \in \Omega(\boldsymbol{\theta})$ where $\boldsymbol{x}_n = \mathsf{X}(\boldsymbol{p}_n, \theta)$ for all $0 \le n \le N$. Applying a change of variable based on the relation $\bar{\boldsymbol{x}} = \bar{\mathsf{X}}(\bar{\boldsymbol{p}}, \theta)$ to Eq. (1) leads to the **material-form path integral**:

$$I = \int_{\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{3}$$

where $\hat{\Omega}$ is the **material path space** independent of θ, and $\hat{f}$ is the **material measurement contribution** given by the ordinary measurement contribution and the Jacobian determinant capturing the change of variable from $\bar{\boldsymbol{x}}$ to $\bar{\boldsymbol{p}}$.

If practice, when estimating derivatives at some $\theta = \theta_0$, the reference configuration is typically selected as $\mathcal{B} = \mathcal{M}(\theta_0)$. This makes $\mathsf{X}(\cdot, \theta_0)$ the identity map.

Differential path integral. Differentiating Eq. (3) w.r.t. a scene parameter θ yields the **material-form differential path integral**:

$$\frac{\mathrm{d}I}{\mathrm{d}\theta} = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}\hat{f}}{\mathrm{d}\theta}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \Delta\hat{f}_K(\bar{\boldsymbol{p}}) \, v(\boldsymbol{p}_K) \, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}}, \tag{4}$$

where $\partial\hat{\Omega}$ is the (material) **boundary path space** comprised of (material) **boundary paths** that are essentially material paths with exactly one vertex $\boldsymbol{p}_K$ such that $\overline{\boldsymbol{x}_{K-1}\,\boldsymbol{x}_K}$ is a **boundary segment**—that is, $\boldsymbol{x}_{K-1}$ and $\boldsymbol{x}_K$ are located on the visibility boundary[1] of each other. We note that, although the material path space $\hat{\Omega}$ is independent of the scene parameter θ, the boundary path space $\partial\hat{\Omega}$ generally does depend on θ (as the visibility boundaries can move when the scene geometry changes). Additionally, $v(\boldsymbol{p}_K)$ is a scalar capturing the change rate of $\boldsymbol{p}_K$ (when $\boldsymbol{p}_{K-1}$ is fixed) with respect to θ along the normal of the visibility boundary.

Vector-valued forms. Rewriting the material path integral (3) in vector-valued forms for $m_{\boldsymbol{I}}$ radiometric measurements (i.e., $\boldsymbol{I} \in \mathbb{R}^{m_{\boldsymbol{I}} \times 1}$) and $m_{\boldsymbol{\theta}}$ scene parameters (i.e., $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times 1}$) gives:

$$\boldsymbol{I} = \int_{\hat{\Omega}} \hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{5}$$

[1] When $\boldsymbol{x}_K$ is a surface vertex, the visibility boundary (with respect to $\boldsymbol{x}_{K-1}$) consists of a set of curves. When $\boldsymbol{x}_K$ is a volume vertex, the boundary consist of a set of surfaces. Please see the work by Zhang et al. [2021b] for more details.

where $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$ is an $m_{\boldsymbol{I}}$-dimensional (column) vector. We note that, in general, the scene geometry $\mathcal{M}$ as well as the mappings X (from the reference configuration $\mathcal{B}$ to the scene geometry) and $\bar{\mathsf{X}}$ (from material paths to ordinary ones) can depend on all scene parameters $\boldsymbol{\theta}$.

Given Eq. (5), it is easy to verify that the vector-valued form of the differential path integral (4) is:

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}} = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}\hat{\boldsymbol{f}}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}) \, \boldsymbol{v}(\boldsymbol{p}_K)^\top \, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}}, \tag{6}$$

where $\mathrm{d}\boldsymbol{I}/\mathrm{d}\boldsymbol{\theta}$, $(\mathrm{d}\hat{\boldsymbol{f}}/\mathrm{d}\boldsymbol{\theta})(\bar{\boldsymbol{p}}) \in \mathbb{R}^{m_{\boldsymbol{I}} \times m_{\boldsymbol{\theta}}}$; $\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}) \in \mathbb{R}^{m_{\boldsymbol{I}} \times 1}$; and $\boldsymbol{v}(\boldsymbol{p}_K) \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times 1}$. We note that, $\boldsymbol{v}(\boldsymbol{p}_K)$ is a vector now since the scalar change rates (along the normal of visibility boundaries) vary among different scene parameters. Further, although some terms in Eq. (6) such as $\mathrm{d}\boldsymbol{I}/\mathrm{d}\boldsymbol{\theta}$ and $(\mathrm{d}\hat{\boldsymbol{f}}/\mathrm{d}\boldsymbol{\theta})(\bar{\boldsymbol{p}})$ can be extremely large (i.e., consist of trillions of elements), they do not need to be stored explicitly when calculating gradients of loss functions, which we discuss in §4.

As a special case, when the scene parameters $\boldsymbol{\theta}$ do not control geometry (in other words, no visibility boundary depends on $\boldsymbol{\theta}$), the mapping $\mathsf{X}(\cdot, \boldsymbol{\theta})$ reduces to the identity map for all $\boldsymbol{\theta}$—which causes the material path space $\hat{\Omega}$ to be identical to the ordinary path space Ω—and the *boundary* integrals in Eqs. (4) and (6) vanish. Then, Eq. (6) reduces to the well-known result

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}} = \int_{\Omega} \frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}}) \, \mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{7}$$

obtained by exchanging the order of differentiation and integration.

4 Gradients as Path Integrals

Many inverse-rendering problems are formulated as finding scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ minimizing some (scalar-valued) loss $\mathcal{L}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{I}(\boldsymbol{\theta})) \, . \tag{8}$$

This framing is referred to as *analysis by synthesis* in computer vision. In practice, the loss $\mathcal{L}$ can also directly depend on $\boldsymbol{\theta}$ when, for instance, regularizing scene parameters. This is orthogonal to our work and we omit this dependency in the remainder of our exposition.

Efficiently solving optimization problems as in Eq. (8) requires computing gradients of the loss $\mathcal{L}$ with respect to the scene parameters $\boldsymbol{\theta}$. Based on the chain rule, it holds that

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = (\partial_{\boldsymbol{I}}\mathcal{L}) \frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}, \tag{9}$$

where $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta} \in \mathbb{R}^{1 \times m_{\boldsymbol{\theta}}}$ and $\partial_{\boldsymbol{I}}\mathcal{L} := \partial\mathcal{L}/\partial\boldsymbol{I} \in \mathbb{R}^{1 \times m_{\boldsymbol{I}}}$ are row vectors, and $\mathrm{d}\boldsymbol{I}/\mathrm{d}\boldsymbol{\theta}$ is an $(m_{\boldsymbol{I}} \times m_{\boldsymbol{\theta}})$-matrix. Lastly, substituting the differential path integral of Eq. (6) into Eq. (9) yields:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \underbrace{\int_{\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L}) \frac{\mathrm{d}\hat{\boldsymbol{f}}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L}) \, \Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}) \, \boldsymbol{v}(\boldsymbol{p}_K)^\top \, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}} . \tag{10}$$

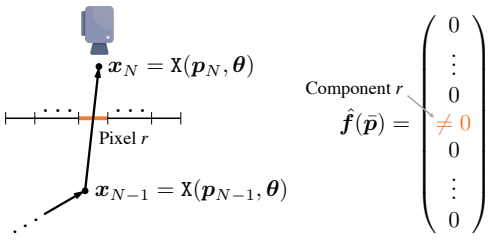

Fig. 2. Given a material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{p}_N)$, its material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$ is typically very **sparse** where only components corresponding to pixels that "intersect" the segment $\overline{\boldsymbol{x}_{N-1}\,\boldsymbol{x}_N}$—i.e., pixels whose reconstruction filters have supports covering the projection of $\boldsymbol{x}_{N-1}$ on the image plane—are nonzero.

Given Eq. (10), we make the following key observation: Taking as input $\partial_{\boldsymbol{I}}\mathcal{L}$ (with the same dimension as $\boldsymbol{I}$), *the gradient* $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ *can be computed directly by estimating (*interior *and* boundary*) path integrals*—that is, without the need to compute or store the large matrix $\mathrm{d}\boldsymbol{I}/\mathrm{d}\boldsymbol{\theta}$ (or individual elements of $\boldsymbol{I}$ in a differentiable fashion).

In what follows, we examine both the *interior* and the *boundary* components of Eq. (10). Additionally, we will discuss numerical estimations of these terms in §5.

Interior component. Without loss of generality, the vector-valued form of material measurement contribution function $\hat{\boldsymbol{f}}$ can be expressed as the product of a vector-valued $\hat{\boldsymbol{f}}_0$ and a scalar-valued $\hat{f}_1$. That is, for any material path $\bar{\boldsymbol{p}}$, we have

$$\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}) = \hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}})\,\hat{f}_1(\bar{\boldsymbol{p}}), \tag{11}$$

where $\hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}}) \in \mathbb{R}^{m_{\boldsymbol{I}}\times 1}$ and $\hat{f}_1(\bar{\boldsymbol{p}}) \in \mathbb{R}$.

When the radiometric measurements $\boldsymbol{I}$ are the pixels intensities of a perspective pinhole camera – which is the case we focus on – $\hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}})$ encodes the per-pixel reconstruction filters, while $\hat{f}_1(\bar{\boldsymbol{p}})$ captures the product of all the other components—such as BRDFs, geometric terms, and Jacobian determinants resulting from the material-form reparameterization—that are invariant across pixels. In this case, for any given material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{p}_N)$, $\hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}}) \in \mathbb{R}^{m_{\boldsymbol{I}}\times 1}$ is generally *sparse* since only pixels whose reconstruction filters "cover" the segment $\overline{\boldsymbol{x}_{N-1}\,\boldsymbol{x}_N}$ will have nonzero values (see Figure 2). Let $\mathrm{nz}(\bar{\boldsymbol{p}}) \subseteq \{1, 2, \ldots, m_{\boldsymbol{I}}\}$ denote the indices of nonzero elements of $\hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}})$. Then, it holds that the integrand of the *interior* term in Eq. (10) equals

$$(\partial_{\boldsymbol{I}}\mathcal{L})\,\frac{\mathrm{d}\hat{\boldsymbol{f}}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{p}}) = \sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}}\mathcal{L})[r]\,\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]\right), \tag{12}$$

where $(\partial_{\boldsymbol{I}}\mathcal{L})[r]$ and $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$—both of which are scalars—denote the r-th components of $\partial_{\boldsymbol{I}}\mathcal{L}$ and $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$, respectively.

Boundary component. The *boundary* integral in Eq. (10) is unique to differentiable rendering. Similar to $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$ in the *interior* term, $\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}})$ in the *boundary* integral is also *sparse* in general, when the radiometric measurement $\boldsymbol{I}$ are pixel intensities. It follows that the integrand of the *boundary* integral equals

$$(\partial_{\boldsymbol{I}}\mathcal{L})\,\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}})\,\boldsymbol{v}(\boldsymbol{p}_K)^\top = \sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}}\mathcal{L})[r]\,(\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}))[r]\,\boldsymbol{v}(\boldsymbol{p}_K)^\top, \tag{13}$$

where $(\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}))[r] \in \mathbb{R}$ denotes the r-th component of $\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}})$. Further, the $\mathrm{nz}(\cdot)$ function in Eq. (13) is the same as the one in Eq. (12), since the set of pixels to which a light transport path contribute is regardless of whether the path is ordinary or boundary.

Simple case. In the special case where the parameters $\boldsymbol{\theta}$ do not affect scene geometry (or visibility boundaries), as discussed in §3, the material path space $\hat{\Omega}$ and the ordinary one Ω coincide, and the *boundary* integrals vanish. Eq. (10), therefore, simplifies to

$$\begin{aligned}\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} &= \int_\Omega (\partial_{\boldsymbol{I}}\mathcal{L})\,\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}}) \\ &= \int_\Omega \sum_{r\in\mathrm{nz}(\bar{\boldsymbol{x}})} (\partial_{\boldsymbol{I}}\mathcal{L})[r]\,\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\boldsymbol{f}(\bar{\boldsymbol{x}})[r]\right)\mathrm{d}\mu(\bar{\boldsymbol{x}}).\end{aligned} \tag{14}$$

5 Efficient Computation of Gradients

We now introduce a new Monte Carlo framework for estimating the gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ given by Eq. (10).

5.1 Estimating the Interior Path Integral

Given Eq. (12), we can obtain the following unbiased (single-sample) Monte Carlo estimator of the *interior* component:

$$\left\langle \frac{\sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}}\mathcal{L})[r]\,\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]\right)}{\mathrm{pdf}_{\mathrm{i}}(\bar{\boldsymbol{p}})} \right\rangle, \tag{15}$$

where $\mathrm{pdf}_{\mathrm{i}}(\bar{\boldsymbol{p}})$ denotes the probability density for sampling the material light path $\bar{\boldsymbol{p}} \in \hat{\Omega}$.

A key benefit offered by our formulation of Eq. (15) is the complete *decoupling* of *path sampling* (that is, the construction of material light path $\bar{\boldsymbol{p}}$) and the *differentiation* of measurement contribution (that is, the evaluation of the gradient $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]}{\mathrm{d}\boldsymbol{\theta}}$). Not having to differentiate path sampling provides the benefit that expensive operations like ray-mesh (or ray-triangle) intersection need not be differentiated.

We present a general-purpose method (that is not specific to any path sampling method) for efficient evaluation of Eq. (15) in §5.1.1, followed by a discussing of how this method can be further optimized in unidirectional (§5.1.2) and bidirectional path tracing (§5.1.3) settings.

5.1.1 Differentiating measurement contributions. Once a material light path $\bar{\boldsymbol{p}}$ is drawn, the estimation of the *interior* integral boils down to (i) identifying all pixels affected by this path (i.e., $\mathrm{nz}(\bar{\boldsymbol{p}})$); and (ii) computing the numerator of Eq. (15) for each affected pixel r. Since the first step can typically be done easily (by examining the segment corresponding to the camera ray), we focus on the second step in the following.

Given any material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$, for all $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}\times 1}$, let $\bar{\boldsymbol{x}} = \bar{\mathtt{X}}(\bar{\boldsymbol{p}}, \boldsymbol{\theta}) = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ be the corresponding ordinary path (where $\boldsymbol{x}_i = \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})$ for $0 \le i \le N$). It holds that

$$\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r] = \left[\prod_{n=0}^{N} \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1}\to\boldsymbol{x}_n\to\boldsymbol{x}_{n+1})\right]\left[\prod_{n=0}^{N-1} G(\boldsymbol{x}_n\leftrightarrow\boldsymbol{x}_{n+1})\right], \tag{16}$$

where G is the (generalized) geometric term. Additionally,

$$\hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1}\to\boldsymbol{x}_n\to\boldsymbol{x}_{n+1}) := f_{\mathrm{v}}(\boldsymbol{x}_{n-1}\to\boldsymbol{x}_n\to\boldsymbol{x}_{n+1})\,J(\boldsymbol{p}_n), \tag{17}$$

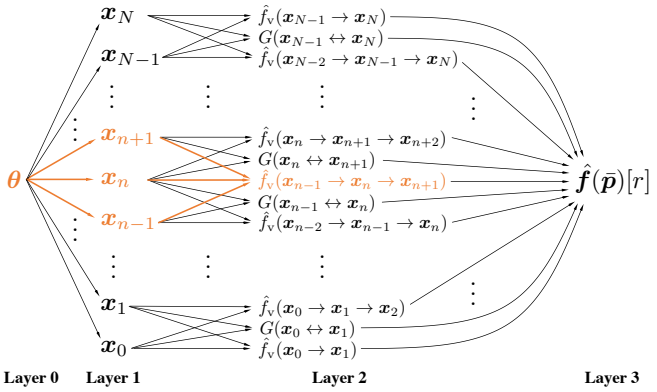

Fig. 3. A layered **computation graph** for evaluating the material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$. All terms on which $\hat{f}_\mathrm{v}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ depends are highlighted in orange. We omitted the vertices $\boldsymbol{p}_n$ of the material path $\bar{\boldsymbol{p}}$ as they are independent of the scene parameters $\boldsymbol{\theta}$.

where J is Jacobian determinant resulting from the material-form reparameterization (i.e., the change of variable from $\boldsymbol{x}_n$ to $\boldsymbol{p}_n$), and f_v captures the measurement contribution of each path vertex:

- When $0 < n < N$, $f_\mathrm{v}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ is given by the surface BSDF or the scaled single-scattering phase function at $\boldsymbol{x}_n$;
- When $n = 0$, $f_\mathrm{v}(\boldsymbol{x}_{-1} \to \boldsymbol{x}_0 \to \boldsymbol{x}_1) := L_\mathrm{e}(\boldsymbol{x}_0 \to \boldsymbol{x}_1)$ captures the emission at $\boldsymbol{x}_0$ (with $\boldsymbol{x}_{-1}$ being a dummy variable);
- When $n = N$, $f_\mathrm{v}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N \to \boldsymbol{x}_{N+1}) := W_\mathrm{e}^{(r)}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N)$ represents the response of pixel r and encodes the pixel reconstruction filter (with $\boldsymbol{x}_{N+1}$ being a dummy variable).

Please refer to the work of Zhang et al. [2021b] for more details.

Evaluating Eq. (15) requires computing the gradient of Eq. (16) with respect to the scene parameters $\boldsymbol{\theta}$. Since Eq. (16) is a scalar-valued expression (per color channel or wavelength), the computation can be implemented using standard reverse-mode automatic differentiation; however, when the light path $\bar{\boldsymbol{p}}$ contains many vertices, evaluating Eq. (16) will involve a great number of computations that require a large computation graph to represent. As observed in prior work [Nimier-David et al. 2020], this can be problematic for storage (e.g., precluding GPU implementation) and performance.

Exploiting independencies. To address this problem, we make an important observation that the individual $\hat{f}_\mathrm{v}$ and G terms on the right-hand side of Eq. (16) can be evaluated in a largely *independently* fashion—even if the parameters $\boldsymbol{\theta}$ control scene geometry. This is thanks to the material-form reparameterization: gradients $\mathrm{d}\boldsymbol{x}_n/\mathrm{d}\boldsymbol{\theta}$ of path vertices can be computed independently by differentiating the mapping $\mathsf{X}(\cdot, \boldsymbol{\theta})$ for all $0 \le n \le N$:

$$\frac{\mathrm{d}\boldsymbol{x}_n}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathsf{X}(\boldsymbol{p}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}. \tag{18}$$

Figure 3 illustrates the computation graph for evaluating $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$. This graph consists of several layers where all nodes in each layer can be evaluated independent of each other. Exploiting this structure, we evaluate the gradient $\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]/\mathrm{d}\boldsymbol{\theta}$ by traversing the computation graph in a layer-by-layer fashion.

ALGORITHM 1: Efficient differentiation of material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$ in Eq. (27) with respect to scene parameters $\boldsymbol{\theta}$

```
1  ComputerMeasurementContribution(θ, p̄, r)
   Input: Scene parameters θ, a material path p̄ = (p_0, ..., p_N), and a
          pixel index r
   Output: f̂(p̄)[r] and its gradient df̂(p̄)[r]/dθ
2  begin
       /* Forward pass (layer 1)                                   */
3      x_n = X(p_n, θ) for each 0 ≤ n ≤ N;
       /* Forward pass (layer 2)                                   */
4      g_n = f̂_v(x_{n-1} → x_n → x_{n+1}) for each 0 ≤ n ≤ N;
5      g_n = G(x_{n-N-1} ↔ x_{n-N}) for each N < n ≤ 2N;
       /* Forward pass (layer 3)                                   */
6      f̂ = ∏_{n=0}^{2N} g_n;
       /* Backward pass (layer 2): compute dg_n := df̂/dg_n         */
7      dg_n = f̂/g_n for each 0 ≤ n ≤ 2N ;   // df̂/dg_n = ∏_{n'≠n} g_{n'} = f̂/g_n
       /* Backward pass (layer 1): compute dx_n := df̂/dx_n         */
8      foreach 0 ≤ n ≤ N do
9          (dx_{n-1}, dx_n, dx_{n+1}) += dg_n ∂f̂_v(x_{n-1}→x_n→x_{n+1})/∂(x_{n-1}, x_n, x_{n+1});
10     end
11     foreach 0 ≤ n < N do
12         (dx_n, dx_{n+1}) += dg_{N+1+n} ∂G(x_n↔x_{n+1})/∂(x_n, x_{n+1});
13     end
       /* Backward pass (layer 0) compute dθ := df̂/dθ             */
14     dθ = Σ_{n=0}^{N} dx_n ∂X(p_n, θ)/∂θ;
15     return (f̂, dθ);
16 end
```

As shown in Algorithm 1, our technique first performs a *forward* pass that evaluates $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$ followed with a *backward* pass that computes the gradient $\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}[r])/\mathrm{d}\boldsymbol{\theta}$. During the latter pass, the gradients $\frac{\partial \hat{f}_\mathrm{v}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$, $\frac{\partial G(\boldsymbol{x}_n \leftrightarrow \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$, and $\frac{\partial \mathsf{X}(\boldsymbol{p}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ from Lines 9, 12 and 14, respectively, can be computed efficiently using standard reverse-mode automatic differentiation.

In what follows, we discuss how Algorithm 1 can be further optimized for unidirectional and bidirectional path tracing – two widely adopted path sampling methods.

5.1.2 Path-tracing-specific optimizations. Unidirectional path tracing with next-event estimation (NEE) constructs a single detector subpath $\bar{\boldsymbol{p}}^\mathrm{D} = (\boldsymbol{p}_0^\mathrm{D}, \ldots, \boldsymbol{p}_N^\mathrm{D})$ coupled with a set of vertices $\boldsymbol{p}_1^\mathrm{S}, \ldots, \boldsymbol{p}_N^\mathrm{S}$ obtained by sampling emitter surfaces. For every $0 < n \le N$, connecting $\boldsymbol{p}_n^\mathrm{D}$ and $\boldsymbol{p}_n^\mathrm{S}$ produces a full light transport path

$$\bar{\boldsymbol{p}}_n = (\boldsymbol{p}_n^\mathrm{S}, \boldsymbol{p}_n^\mathrm{D}, \boldsymbol{p}_{n-1}^\mathrm{D}, \ldots, \boldsymbol{p}_0^\mathrm{D}), \tag{19}$$

as illustrated in Figure 4.[2] Then, the *interior* component of Eq. (10) can be estimated by summing Eq. (15) over all $\bar{\boldsymbol{p}}_n$:

$$\left\langle \sum_{n=1}^{N} \frac{\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}}_n)} (\partial_I \mathcal{L})[r] \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_n)[r] \right)}{\mathrm{pdf}_\mathrm{NEE}(\bar{\boldsymbol{p}}_n)} \right\rangle. \tag{20}$$

[2] Strictly speaking, we need to also consider two-vertex paths of the form $(\boldsymbol{p}_0^\mathrm{S}, \boldsymbol{p}_0^\mathrm{D})$. We neglect this corner case to simplify our derivations.

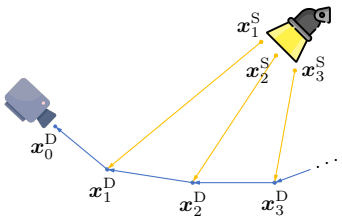

Fig. 4. With next-event estimation (NEE), a unidirectional path tracer effectively constructs a set of light paths that share one detector subpath $(\boldsymbol{x}_0^{\mathrm{D}}, \boldsymbol{x}_1^{\mathrm{D}}, \ldots)$ shown in blue. The arrows in this figure illustrates the flow of light. We present a technique to efficiently compute and differentiate the measurement contribution of all paths by factoring out common terms.

Although this expression can be evaluated by applying Algorithm 1 to each path $\bar{\boldsymbol{p}}_n$ individually, doing so would lead to suboptimal performance since many terms such as $G(\boldsymbol{x}_0^{\mathrm{D}} \leftrightarrow \boldsymbol{x}_1^{\mathrm{D}})$ will be computed (and differentiated) multiple times.

To address this problem, noting that $\mathrm{nz}(\bar{\boldsymbol{p}}_n) = \mathrm{nz}(\bar{\boldsymbol{p}}^{\mathrm{D}})$ for all $0 < n \le N$, we rearrange the terms of Eq. (20) and obtain:

$$\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}}^{\mathrm{D}})} (\partial_I \mathcal{L})[r] \frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}\boldsymbol{\theta}} \text{ where } h_{\mathrm{NEE}} := \sum_{n=1}^{N} \frac{\hat{f}(\bar{\boldsymbol{p}}_n)[r]}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)}, \tag{21}$$

with $\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)$ treated as "detached" (i.e., independent of $\boldsymbol{\theta}$). To efficiently compute Eq. (21), similar to how unidirectional path tracing is implemented for forward rendering, we factor out the common terms in the inner summation of Eq. (21). Let

$$h_n^{\mathrm{D}} := \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n+1}^{\mathrm{D}} \to \boldsymbol{x}_n^{\mathrm{D}} \to \boldsymbol{x}_{n-1}^{\mathrm{D}})\, G(\boldsymbol{x}_{n+1}^{\mathrm{D}} \leftrightarrow \boldsymbol{x}_n^{\mathrm{D}}), \tag{22}$$

$$h_n^{\mathrm{S}} := \frac{\hat{f}_{\mathrm{v}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_n^{\mathrm{D}} \to \boldsymbol{x}_{n-1}^{\mathrm{D}})\, G(\boldsymbol{x}_0^{\mathrm{S}} \leftrightarrow \boldsymbol{x}_n^{\mathrm{D}})\, \hat{L}_{\mathrm{e}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)}, \tag{23}$$

where $\hat{L}_{\mathrm{e}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_0^{\mathrm{D}}) := L_{\mathrm{e}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_0^{\mathrm{D}})\, J(\boldsymbol{p}_0^{\mathrm{S}})$ captures the emission at $\boldsymbol{x}_0^{\mathrm{S}}$. Then, it is easy to verify that, for all $0 < n \le N$,

$$\frac{\hat{f}(\bar{\boldsymbol{p}}_n)[r]}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)} = \left(\prod_{n'=0}^{n} h_{n'}^{\mathrm{D}}\right) h_n^{\mathrm{S}}. \tag{24}$$

It follows that

$$h_{\mathrm{NEE}} = h_0^{\mathrm{D}} \left(h_1^{\mathrm{S}} + h_1^{\mathrm{D}} \left(h_2^{\mathrm{S}} + h_2^{\mathrm{D}} \left(h_3^{\mathrm{S}} + h_3^{\mathrm{D}}(\ldots)\right)\right)\right), \tag{25}$$

which can be differentiated in a layered fashion using a process similar to Algorithm 1. Specifically, in the forward pass, we first obtain path vertices $\boldsymbol{x}_n^{\mathrm{D}} = \mathsf{X}(\boldsymbol{p}_n^{\mathrm{D}}, \boldsymbol{\theta})$ and $\boldsymbol{x}_n^{\mathrm{S}} = \mathsf{X}(\boldsymbol{p}_n^{\mathrm{S}}, \boldsymbol{\theta})$ for all n (layer 1), followed with computing individual h_n^{D} and h_n^{S} terms (layer 2). Then, we evaluate h_{NEE} (layer 3). In the backward pass, we start with obtaining derivatives $\mathrm{d}h_n^{\mathrm{D}} := \mathrm{d}h_{\mathrm{NEE}}/\mathrm{d}h_n^{\mathrm{D}}$ and $\mathrm{d}h_n^{\mathrm{S}} := \mathrm{d}h_{\mathrm{NEE}}/\mathrm{d}h_n^{\mathrm{S}}$ (layer 2) by differentiating Eq. (25). Then, we evaluate $\mathrm{d}h_{\mathrm{NEE}}/\mathrm{d}\boldsymbol{x}_n^{\mathrm{D}}$ and $\mathrm{d}h_{\mathrm{NEE}}/\mathrm{d}\boldsymbol{x}_n^{\mathrm{S}}$ (layer 1) followed with the gradient $\mathrm{d}h_{\mathrm{NEE}}/\mathrm{d}\boldsymbol{\theta}$ (layer 0) that can be used to estimate the *interior* integral via Eq. (21).

5.1.3 BDPT-specific optimizations. A bidirectional path tracer typically constructs a source subpath $\bar{\boldsymbol{p}}^{\mathrm{S}} = (\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_1^{\mathrm{S}}, \ldots)$ and a detector subpath $\bar{\boldsymbol{p}}^{\mathrm{D}} = (\boldsymbol{p}_0^{\mathrm{D}}, \boldsymbol{p}_1^{\mathrm{D}}, \ldots)$. Let $\bar{\boldsymbol{p}}_{s,t}$ be the material light path obtained by connecting the s-th vertex in the source subpath and the t-th vertex in the detector subpath. That is, $\bar{\boldsymbol{p}}_{s,t} = (\boldsymbol{p}_0^{\mathrm{S}}, \ldots, \boldsymbol{p}_s^{\mathrm{S}}, \boldsymbol{p}_t^{\mathrm{D}}, \ldots, \boldsymbol{p}_0^{\mathrm{D}})$. Then, it holds that the *interior* integral in Eq. (10) can be estimated using

$$\left\langle \sum_{s,t} \sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}}_{s,t})} w_{s,t}(\bar{\boldsymbol{p}}_{s,t}) \frac{(\partial_I \mathcal{L})[r] \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left(\hat{f}(\bar{\boldsymbol{p}}_{s,t})[r]\right)}{\mathrm{pdf}_{s,t}(\bar{\boldsymbol{p}}_{s,t})} \right\rangle, \tag{26}$$

where $\mathrm{pdf}_{s,t}$ is the probability density (for sampling a path with s vertices from the source and t from the detector), and $w_{s,t}$ is the corresponding multiple-importance-sampling (MIS) weight.

To evaluate Eq. (26) numerically, we start with constructing the source and detector subpaths $\bar{\boldsymbol{p}}^{\mathrm{S}}$ and $\bar{\boldsymbol{p}}^{\mathrm{D}}$ followed with computing the PDFs $\mathrm{pdf}_{s,t}(\bar{\boldsymbol{p}}_{s,t})$ and the MIS weights $w_{s,t}(\bar{\boldsymbol{p}}_{s,t})$ for all s and t. Since none of these terms need to be differentiated, they can be computed in a similar way as conventional BDPT does (for forward rendering). Then, we evaluate (in a differentiable fashion) the BSDF and geometric terms along both the source subpath $\bar{\boldsymbol{p}}^{\mathrm{S}}$ and the detector subpath $\bar{\boldsymbol{p}}^{\mathrm{D}}$ in a layered fashion similar to Algorithm 1. Lastly, we reuse these terms to evaluate the gradient $\mathrm{d}\hat{f}(\bar{\boldsymbol{p}}_{s,t})[r]/\mathrm{d}\boldsymbol{\theta}$ for all s and t while avoiding duplicate computation/differentiation.

5.1.4 Relation with prior works. Our technique presented above is closely related to some recent works [Nimier-David et al. 2020; Vicini et al. 2021]. These techniques formulate image-loss gradients as solutions of an adjoint transport problem. From a theoretical perspective, they can be considered specific realization of our technique—akin to how adjoint particle tracing relates to the path integral formulation in forward rendering. In Appendix A, we provide further discussions on the theoretical connection between our technique and those works.

5.2 Estimating the Boundary Path Integral

Eq. (13) induces an unbiased (single-sample) Monte Carlo estimator of the *boundary* component of Eq. (10) as

$$\left\langle \frac{\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_I \mathcal{L})[r]\, (\Delta \hat{f}_K(\bar{\boldsymbol{p}}))[r]\, \boldsymbol{v}(\boldsymbol{p}_K)^{\top}}{\mathrm{pdf}_{\mathrm{b}}(\bar{\boldsymbol{p}})} \right\rangle, \tag{27}$$

where $\mathrm{pdf}_{\mathrm{b}}(\bar{\boldsymbol{p}})$ denotes the probability density for sampling the boundary path $\bar{\boldsymbol{p}} \in \partial\hat{\Omega}$. In practice, we follow Zhang et al. [2020; 2021b] and sample $\bar{\boldsymbol{p}}$ in a multi-directional fashion (starting with the boundary segment).

With the boundary path $\bar{\boldsymbol{p}}$ sampled, evaluating the numerator of Eq. (27) is, in fact, relatively inexpensive. This is because, with $\partial_I \mathcal{L}$ provided, the only term in the numerator that requires differentiation is the scalar change rate $\boldsymbol{v}(\boldsymbol{p}_K)$—as oppose to the evaluation of the *interior* integral (§5.1) that requires differentiating the full material measurement contribution. Specifically, it holds that

$$\boldsymbol{v}(\boldsymbol{p}_K) = \left(\mathrm{d}\boldsymbol{p}_K / \mathrm{d}\boldsymbol{\theta}\right)^{\top} \boldsymbol{n}(\boldsymbol{p}_K), \tag{28}$$

where $\mathrm{d}\boldsymbol{p}_K/\mathrm{d}\boldsymbol{\theta} \in \mathbb{R}^{3 \times m_{\boldsymbol{\theta}}}$ and $\boldsymbol{n}(\boldsymbol{p}_K)$ is a three-dimensional (column) vector representing the normal of the visibility boundary at $\boldsymbol{p}_K$. Let

$$V(\boldsymbol{p}_K) := \boldsymbol{p}_K^{\top} \boldsymbol{n}(\boldsymbol{p}_K). \tag{29}$$

Then, it is easy to verify that, with the normal $\boldsymbol{n}(\boldsymbol{p}_K)$ fixed (i.e., set independent of $\boldsymbol{\theta}$), V is an "anti-gradient" of $\boldsymbol{v}$ satisfying

$$\boldsymbol{v}(\boldsymbol{p}_K) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} V(\boldsymbol{p}_K). \tag{30}$$

Fig. 5. Our system utilizes the Enzyme [Moses and Churavy 2020] automatic differentiation framework that operates at the LLVM level by generating differentiable versions of LLVM Intermediate Representations (IR).

It follows that the scalar-valued expression

$$\boldsymbol{p}_K^\top \left(\boldsymbol{n}(\boldsymbol{p}_K) \frac{\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_I \mathcal{L})[r] \, (\Delta \hat{f}_K(\bar{\boldsymbol{p}}))[r]}{\mathrm{pdf}_\mathrm{b}(\bar{\boldsymbol{p}})} \right), \tag{31}$$

with all the terms expect the first (i.e., $\boldsymbol{p}_K^\top$) fixed, is an "anti-gradient" of Eq. (27). Thus, by applying reverse-mode automatic differentiation (i.e., backward) to the result of this expression, we can accumulate the contribution given by Eq. (27) to the final gradient $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$.

6 Results

We describe our differentiable rendering system that implements techniques discussed in §4 and §5 in §6.1, before the effectiveness of our techniques and system in §6.2 and §6.3.

6.1 Our System

We develop a CPU-based differentiable renderer that utilizes Enzyme [Moses and Churavy 2020] for reverse-mode automatic differentiation (AD).[3] As opposed to most conventional AD libraries that work at the source code (e.g., C++) level, Enzyme operates at the LLVM level by taking as input arbitrary code in LLVM Intermediate Representation and computing gradients of that function.

Specifically, as illustrated in Figure 5, we use LLVM to compile our C++ source code (that contains Enzyme-specific intrinsics) into LLVM's Intermediate Representation (IR). Then, we use Enzyme to automatically differentiate the LLVM IR and, in turn, compile the output into executables using LLVM.

Utilizing the Enzyme framework, we compute gradients such as $\frac{\partial \hat{f}_\mathrm{v}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$, $\frac{\partial G(\boldsymbol{x}_n \leftrightarrow \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$, and $\frac{\partial \mathsf{X}(\boldsymbol{p}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$ for Algorithm 1 as well as differentiating Eq. (31) without writing much additional code nor using special AD types (that are typically required by conventional AD frameworks). Since the differentiation occurs at the LLVM level and is integrated with LLVM's powerful optimization pipeline, the resulting system offers high performance and scalability – which we demonstrate concretely in §6.2 and §6.3.

Moreover, we develop an alternative pipeline for generating gradient images with respect to one scene parameter. We use this pipeline **exclusively** for debugging and validation purposes, i.e., not for solving inverse-rendering problems. As with the main pipeline, we employ Algorithm 1 to compute $(\mathrm{d}/\mathrm{d}\boldsymbol{\theta})(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r])$ in the *interior* term and Enzyme to compute $(\mathrm{d}/\mathrm{d}\boldsymbol{\theta})(\boldsymbol{p}_K^\top \boldsymbol{n}(\boldsymbol{p}_K))$ for the *boundary* term. For each computed gradient vector, we select one (predetermined) component and accumulate its contribution to the resulting gradient image.

[3]We will open-source our implementation upon publication.

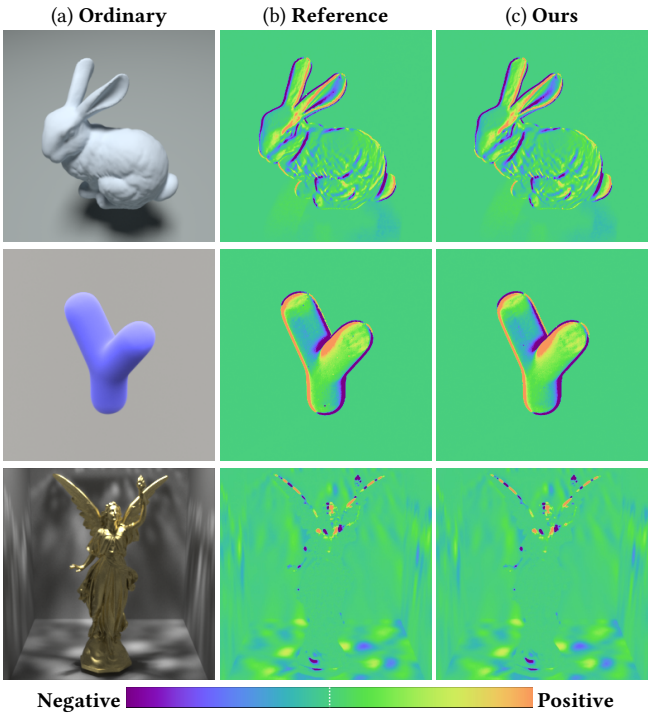

Fig. 6. We **validate** our system by comparing our generated gradient images with references.

6.2 Validations and Evaluations

Validation. We validate our technique and system by comparing gradient images against our strong alternative baseline pipeline (Figure 6). The first two rows illustrate BUNNY scene with a diffuse bunny and a LETTER Y scene with a Y-shaped object containing a homogeneous participating medium. Both scenes are lit by area lights and differentiated with respect to horizontal translations of the objects. The last row uses a CAUSTICS scene with a glossy Lucy underwater lit by a small area light above water. The gradient images for this scene are computed with respect to the translation of the light source.

We use differentiable unidirectional path tracing for the first two scenes and bidirectional for the last one. In all cases, our gradient estimates well match the references.

Performance comparisons. To evaluate the efficiency of our system, we first compare with the GPU-based system psdr-cuda [Zhao 2021] that is based on the Enoki automatic differentiation framework [Jakob 2019]. We choose this system as it shares the same path-space formulation [Zhang et al. 2020, 2021b] as our technique.

To compare performance, we measure the time for both systems to estimate image-loss gradients using the BUNNY scene from Figure 6 but with respect to individual vertex positions (expressed with 30,000 parameters). Since psdr-cuda supports direct illumination only, we configure our system to only simulate one-bounce light transport as well. As shown in Figure 7, our system offers a similar level of performance as psdr-cuda. The latter scales better to higher

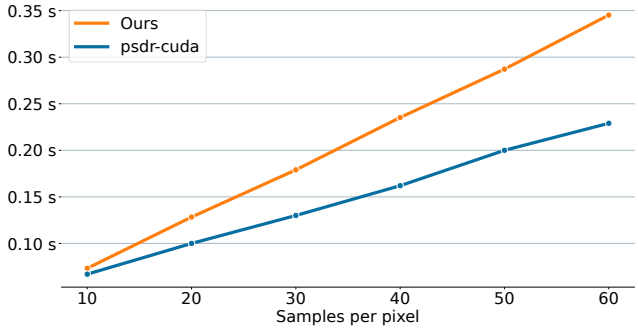

Fig. 7. **Performance comparison:** When estimating image-loss gradients under direct illumination, our CPU-based system offers a similar (albeit improved) level of performance as psdr-cuda [Zhao 2021]. We measure performance statistics on a single workstation with an AMD Ryzen 5950X CPU and an Nvidia RTX Titan GPU.

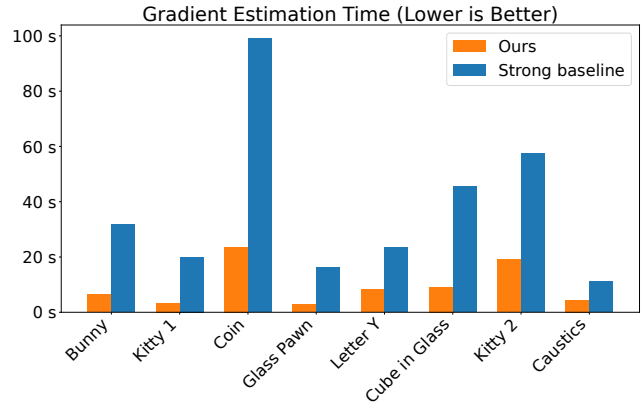

Fig. 8. **Performance comparison:** We compare running times for estimating image-loss gradients $\mathrm{d}\mathcal{L}/\mathrm{d}\boldsymbol{\theta}$ when solving the inverse-rendering problems in Figures 9, 10, and 11. The strong baseline, which already outperforms existing CPU-based systems [Zhang et al. 2020, 2021b], uses Enzyme to differentiate full material measurement contributions $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$ in Eq. (15).

sample counts due to its wavefront design and higher parallelism offered by the GPU.

Additionally, we demonstrate the effectiveness of Algorithm 1 (as well as its further optimizations discussed in §5.1.2 and §5.1.3) by comparing to baselines where full material measurement contributions $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$ are differentiated directly using Enzyme. As shown in Figure 8, exploiting the layered structures of the computation graphs makes the estimation of image-loss gradients up to 6× faster when solving inverse-rendering problems.

6.3 Inverse-Rendering Results

We now show inverse-rendering results using gradients estimated with our differentiable *unidirectional* path tracer in Figures 9 and 10 as well as *bidirectional* path tracer in Figure 11. When optimizing object shapes (expressed as polygonal meshes), we use Nicolet et al.'s method [2021] to update mesh vertex positions in a robust fashion (provided the estimated gradients). Additionally, we utilize mini-batch gradient descent when using multiple target images.

Table 1. Optimization configuration and performance statistics for our inverse-rendering results. The "render time" numbers indicate per-iteration computation time for estimating image-loss gradients; 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 5950X 16-core CPU.

Scene	# Target images	# Param.	Batch size	# Iter.	Render time	Postproc. time
Bunny	40	30,000	2	1,000	8.40s	0.38s
Kitty 1	50	30,000	2	1,000	4.67s	0.35s
Coin	20	1,500,000	3	100	30.57s	11.45s
Glass pawn	40	60,000	2	1,000	4.06s	0.37s
Letter Y	30	15,000	4	200	9.06s	0.97s
Cube in glass	50	30,004	2	500	12.75s	0.23s
Kitty 2	50	150,000	4	600	34.52s	1.75s
Caustics	1	651	1	300	6.46s	0.03s

Table 1 summarizes optimization configurations and performance statistics; we include animations of these results in our supplement.

In Figure 9, the bunny scene is the same as the top row of Figure 6 (with a diffuse bunny inside a box). The kitty 1 scene contains a glossy kitty inside a Cornell box, exhibiting strong interreflections. The coin scene has a very detailed coin geometry. All there scenes are lit by area illumination. For each of these three scenes, we take multiple input image views (only one is shown in the figure) and solve for object shapes by minimizing L_1 image loss. Our system's performance and scalability allows us to optimize high-resolution meshes in these inverse-rendering problems, resulting in high-quality geometric reconstruction.

Additionally, Figure 10 demonstrates our ability to treat scenes with non-opaque (i.e., transparent or translucent) objects: These settings preclude the use of simpler differentiable rasterization-based or direct illumination-only methods. The glass pawn scene contains a pawn made of blue rough glass. The letter Y scene is identical to the second row of Figure 6 and has a Y-shaped object containing a volumetric homogeneous participating medium; our system is sufficiently fast to explicitly treat volumetric scattering effects. The cube in glass scene is comprised of a diffuse cube inside a rough glass enclosure; the cube is only visible after refraction, complicating the inverse-rendering problem. For each scene, similar to the examples shown in Figure 9, we take multiple input images of the object (only showing one in the figure) and optimize the object's shape starting from a spherical initialization. In the cube in glass scene, we jointly optimize the albedo of the diffuse object, the roughness of the glass, and the cube geometry.

Lastly, Figure 11 illustrates inverse-rendering results leveraging our differentiable *bidirectional* path tracer. The Kitty 2 scene contains a Cornell box with a glossy kitty, comprising more complex secondary transport and a *flipped area emitter* that faces and illuminates the ceiling (instead of the central objects), resulting in a mostly indirectly-lit scene. We optimized the shape of the glossy object

(initialized as a sphere) to minimize image loss. The CAUSTICS scene is an underwater setting with a glossy Lucy object (as in Figures 1 and 6). This time, with a single target image, we optimize the shape of the air-water interface *without every directly viewing the surface*.

7 Discussion and Conclusion

Limitations and future work. Our formulation is specific to the material-form variant of steady-state differential path integral. Generalizing it to other parameterizations (such as [Bangaru et al. 2020]) and/or time-resolved settings (e.g., [Wu et al. 2021]) is an interesting topic for future work.

We also suspect that developing efficient differentiation techniques based on Algorithm 1 for path-level antithetic sampling [Zhang et al. 2021a] can lead to compelling results.

Finally, since our system is CPU-based, developing GPU-based extension that exploits the layered structures of the computation graph can further the benefits of our method, opening up opportunities for broader applications where large-scale differentiable rendering is necessary.

Conclusion. Efficient computational differentiation is essential for the development of general-purpose differentiable renderers. We devised a new mathematical formulation that represents image-loss gradients as differential path integrals, allowing us to fully decouple the sampling of light paths and the differentiation of path measurement contributions. Based on our formulation, we introduced new algorithms that efficiently compute the *interior* integrals (by exploiting the layered structure of computation graphs) and demonstrated how the *boundary* integrals can be handled under a unified framework. Furthermore, we developed a new CPU-based system that leverages the Enzyme AD framework.

Thanks to its generality and scalability, our system allows us to solve challenging inverse-rendering problems with millions of parameters, global geometry changes, and complex light transport effects (such as caustics) – which we demonstrated on several challenging scenarios.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*. 265–283.

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.

Sai Praveen Bangaru, Jesse Michel, Kevin Mu, Gilbert Bernstein, Tzu-Mao Li, and Jonathan Ragan-Kelley. 2021. Systematically Differentiating Parametric Discontinuities. *ACM Trans. Graph.* 40, 4 (2021), 107:1–107:18.

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.

Andreas Griewank and Andrea Walther. 2008. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. Vol. 105. Siam.

Wenzel Jakob. 2019. Enoki: structured vectorization and differentiation on modern processor architectures. https://github.com/mitsuba-renderer/enoki.

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 pages.

James T. Kajiya. 1986. The Rendering Equation. *SIGGRAPH Comput. Graph.* 20, 4 (1986), 143–150.

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.

James L McClelland, David E Rumelhart, PDP Research Group, et al. 1986. Parallel distributed processing. *Explorations in the microstructure of cognition* 2 (1986), 216–271.

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.

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, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. 2020. Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering. *ACM Trans. Graph.* 39, 4 (2020), 146:1–146:15 pages.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.

M. Pauly, T. Kollig, and A. Keller. 2000. Metropolis light transport for participating media. In *Rendering Techniques 2000*. Springer, 11–22.

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

Eric Veach and Leonidas Guibas. 1995. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*. Springer, 145–167.

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

Robert Edwin Wengert. 1964. A simple automatic derivative evaluation program. *Commun. ACM* 7, 8 (1964), 463–464.

Lifan Wu, Guangyan Cai, Ravi Ramamoorthi, and Shuang Zhao. 2021. Differentiable Time-Gated Rendering. *ACM Trans. Graph.* 40, 6 (2021), 287:1–287:16.

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

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

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.

Shuang Zhao. 2021. Path-Space Differentiable Renderer. https://github.com/uci-rendering/psdr-cuda/.

A Relation to Prior Works

We will show that key results in some recent works [Nimier-David et al. 2020; Vicini et al. 2021] can be, at a high level, considered as specific realizations of Eq. (10). To do so, we first assume the following:

- The scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times 1}$ do not control object geometry (i.e., do not affect visibility discontinuities);
- The vector-valued measurement contribution satisfies $\boldsymbol{f}(\bar{\boldsymbol{x}}) = \boldsymbol{W}_\mathrm{e}(\bar{\boldsymbol{x}})\, f_1(\bar{\boldsymbol{x}})$ where $\boldsymbol{W}_\mathrm{e}(\bar{\boldsymbol{x}}) \in \mathbb{R}^{m_I \times 1}$ indicates the detector responses, and $f_1(\bar{\boldsymbol{x}}) \in \mathbb{R}$ captures the remaining measurement-contribution terms;
- $\boldsymbol{W}_\mathrm{e}$ is independent of the scene parameters $\boldsymbol{\theta}$.

Given these assumptions, Eq. (14) simplifies to

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \int_\Omega \underbrace{(\partial_I \mathcal{L})\, \boldsymbol{W}_\mathrm{e}(\bar{\boldsymbol{x}})}_{=:\boldsymbol{A}_\mathrm{e}(\bar{\boldsymbol{x}})} \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}})\, \mathrm{d}\mu(\bar{\boldsymbol{x}}). \tag{32}$$

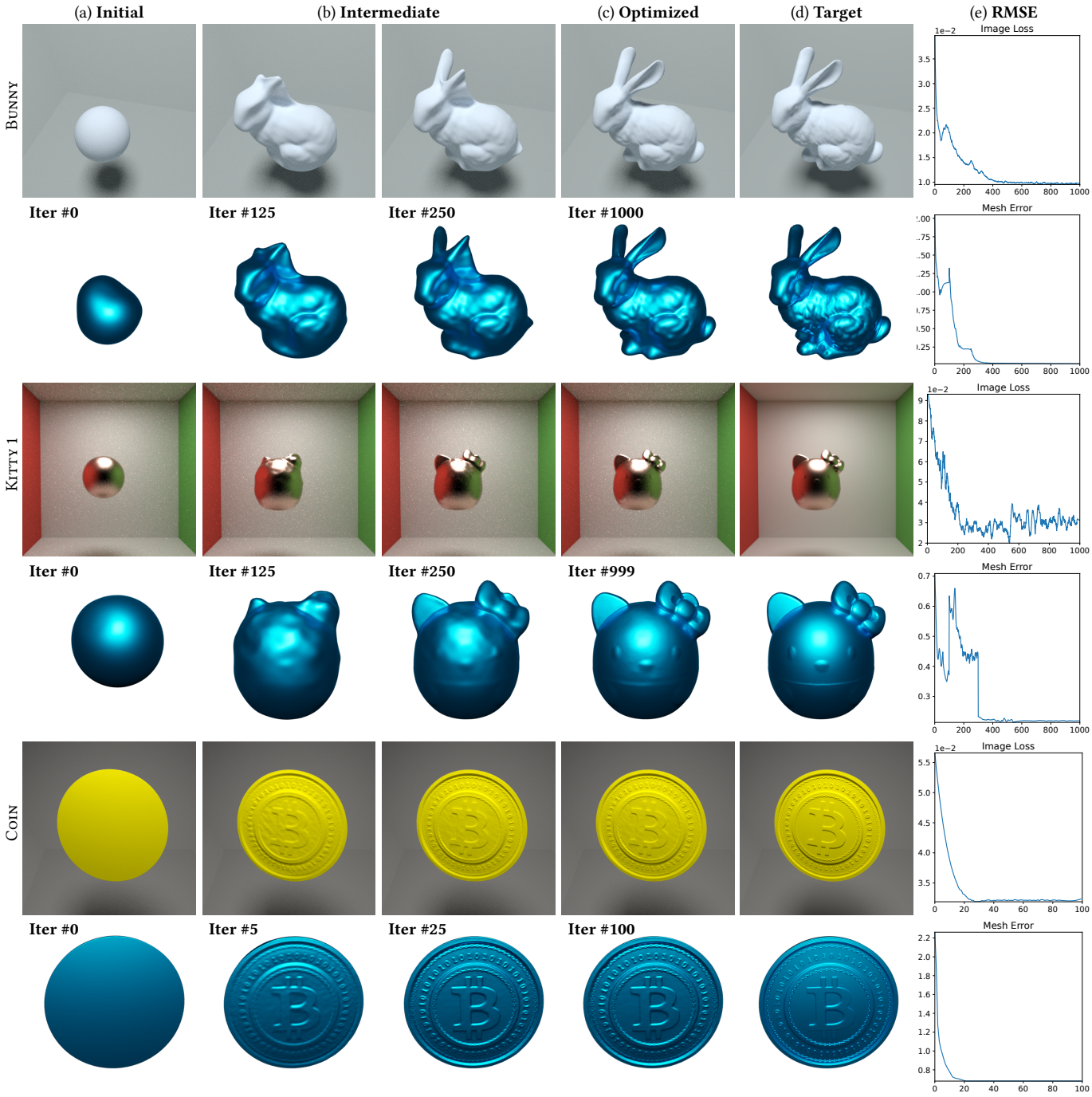

Fig. 9. **Inverse-rendering results** obtained with gradients estimated using our differentiable *unidirectional* path tracer. The mesh error plotted in column (e) captures the Chamfer distance [Barrow et al. 1977] between the reconstructed and groundtruth geometries (normalized so that the GT has a unit bounding box). We use this information only for evaluation (and not for optimization).

In practice, the detector responses $\boldsymbol{W}_\mathrm{e}$ usually depend only on the last segment $\overline{\boldsymbol{x}_{N-1}\boldsymbol{x}_N}$ of a light path $\bar{\boldsymbol{x}}$ – that is, $\boldsymbol{W}_\mathrm{e}(\bar{\boldsymbol{x}}) = \boldsymbol{W}_\mathrm{e}(\boldsymbol{x}_N \to \boldsymbol{x}_{N-1})$. This allows further simplification of Eq. (32) as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \int_{\mathcal{M}^2} \boldsymbol{A}_\mathrm{e}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N) \left[\int_\Omega \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}}_-)\right] \mathrm{d}A(\boldsymbol{x}_{N-1})\,\mathrm{d}A(\boldsymbol{x}_N), \quad (33)$$

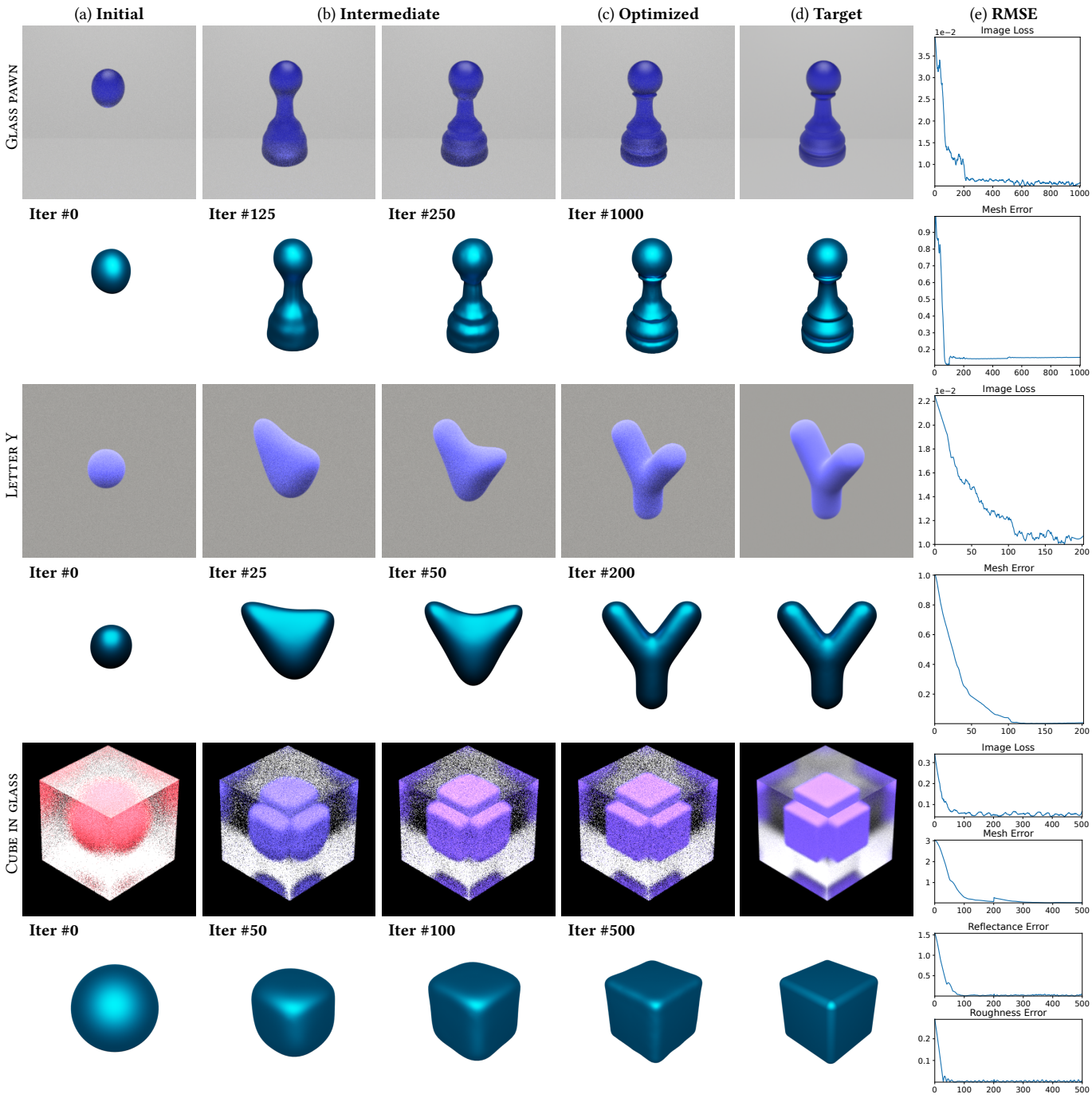

Fig. 10. **Inverse-rendering results** obtained with gradients estimated using our differentiable *unidirectional* path tracer. All examples in this figure involve non-opaque (i.e., transparent or translucent) objects that cannot be easily handled by simple rasterization-based or direct-illumination-only differentiable renderers. The mesh, reflectance, and roughness errors are used for evaluation only (and not for optimization).

where $\bar{\boldsymbol{x}}_-$ is a light path given by $\bar{\boldsymbol{x}}$ with its last two vertices $\boldsymbol{x}_{N-1}$ and $\boldsymbol{x}_N$ removed. Eq. (33) is equivalent to a key result in §3.3 of the radiative backpropagation work [Nimier-David et al. 2020].

Moreover, a key idea in path replay backpropagation [Vicini et al. 2021] is to reuse light transport paths when recursively expanding

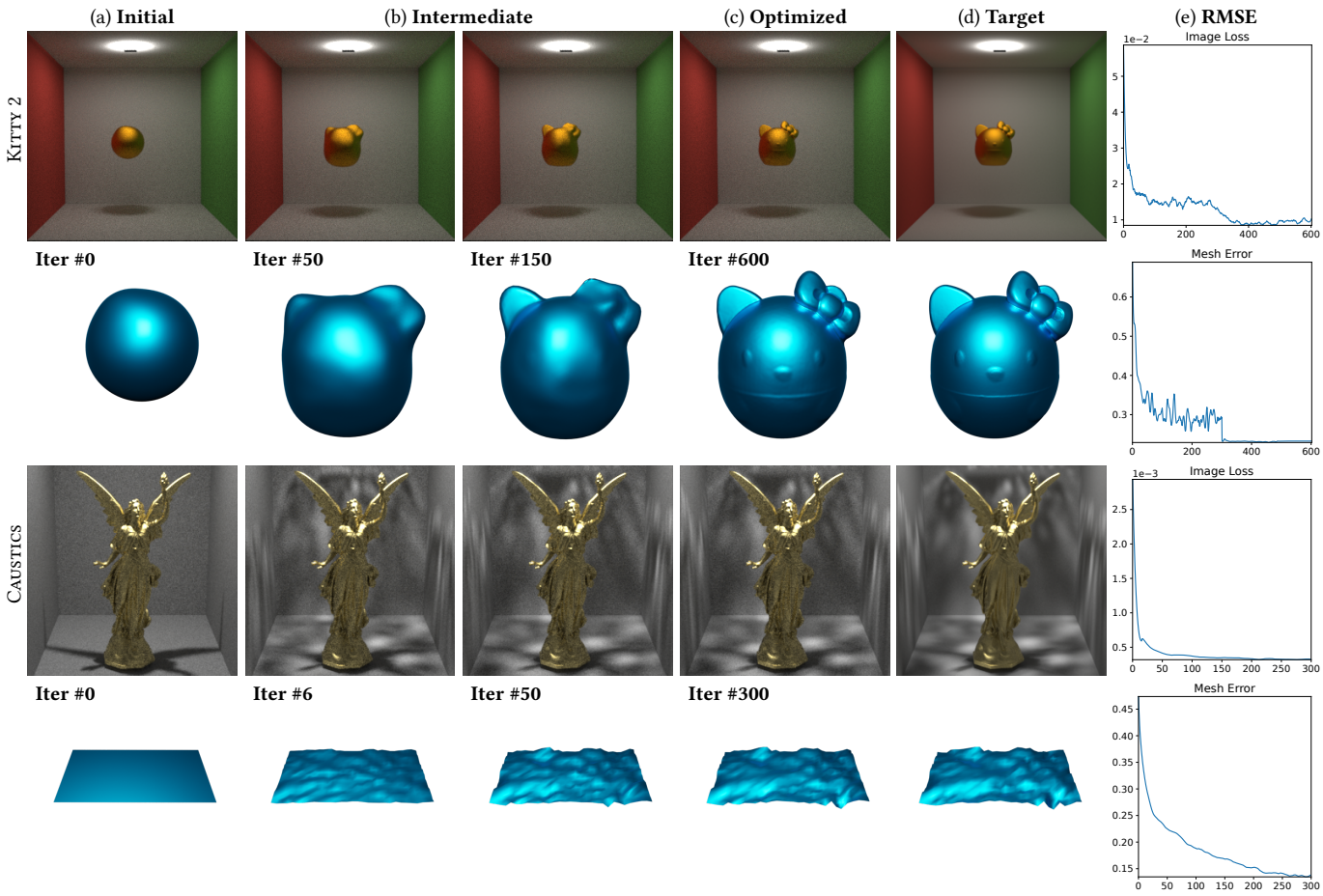

Fig. 11. **Inverse-rendering results** obtained with gradients estimated using our differentiable *bidirectional* path tracer. The mesh error is used for evaluation only (and not for optimization).

the differential rendering equation. Under the differential path integral formulation, this is equivalent to applying the product rule when differentiating the measurement contributions. Specifically, given a material light path $\bar{\boldsymbol{p}}$, assume $\hat{f}(\bar{\boldsymbol{p}}) = \prod_n g_n(\bar{\boldsymbol{p}})$ with $g_n(\bar{\boldsymbol{p}})$ capturing individual components (e.g., BSDFs and geometric terms) of $\hat{f}(\bar{\boldsymbol{p}})$. Then,

$$\int_{\hat{\Omega}} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \,\mathrm{d}\mu(\bar{\boldsymbol{p}}) = \int_{\hat{\Omega}} \left(\sum_n \frac{\mathrm{d}g_n(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \prod_{n' \neq n} g_{n'}(\bar{\boldsymbol{p}}) \right) \mathrm{d}\mu(\bar{\boldsymbol{p}}). \tag{34}$$

The path replay idea essentially states that the right-hand side of this equation can be computed by reusing one path sample $\bar{\boldsymbol{p}}$, which leads to a natural practice under the differential path integral formulation (as is indeed the case for Algorithm 1).