UNIVERSITY OF CALIFORNIA,
IRVINE

Path-Space Differentiable Rendering

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Cheng Zhang

Dissertation Committee:
Professor Shuang Zhao, Chair
Professor Gopi Meenakshisundaram
Professor Aditi Majumder

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude to my advisor Shuang Zhao for his help with my research from the first day I joined his group back in 2017. As his student, I am privileged to receive guidance from him on all aspects of research: not only coding, writing, and presentation skills, but, more importantly, how to think as a research scientist. The completion of this dissertation would not have been possible without his support.

Many thanks also to my committee members, professor Gopi Meenakshisundaram and Aditi Majumder, for their helpful suggestions on the dissertation writing. Furthermore, I would like to extend my gratitude to my colleagues from iGravi Lab, who have created a stimulating environment for graphics research: Zahra Montazeri, Yu Guo, Marco (Zhanhang) Liang, Kai Yan, Guangyan Cai, Zihan Yu, Yuchen Wang, Muhammad Twaha Ibrahim, Andy Thai, Jessica Souza, Isabela Figueira.

I am very grateful to Zhao Dong for appreciating my research and continuously sharing his insights on graphics research from an industrial perspective. I would also like to thank my mentors and friends during my internships: Edward (Shiqiu) Liu, Michael Doggett, Lifan Wu, Zexiang Xu, Yue Liu, Qi Guo, Jiahao Lin, Merlin Nimier-David, Delio Vicini, and Jamorn Sriwasansak.

I am fortunate to have had the opportunity to work with many outstanding collaborators on research projects related to and beyond this dissertation: Bailey Miller, Changxi Zheng, Derek Nowrouzezahrai, Ioannis Gkioulekas, Kai Yan, Lifan Wu, Michael Doggett, Ravi Ramamoorthi, Zhao Dong, Zihan Yu. Thanks to their passion and commitment, the stressful days near SIGGRAPH deadlines are made easier.

Finally, I thank my family for their understanding and unconditional love throughout these years. I thank my girlfriend Chris (Yuhan) Wang for her accompany during this difficult period of the pandemic over the past few years.

My research has been supported by generous grants from the National Science Foundation (grant 1900927), a Meta (formally known as Facebook) Ph.D. Fellowship, and a gift from the AWS Cloud Credits for Research program.

VITA

Cheng Zhang

EDUCATION

Doctor of Philosophy in Computer Science **2017-2022**
University of California, Irvine *Irvine, CA*

Master of Science in Computer Science **2015-2017**
Columbia University in the City of New York *New York, NY*

Bachelor of Engineering in Electronics Information Engineering **2011-2015**
Beijing Univeristy of Technology *Beijing, China*

REFEREED PUBLICATIONS

[1] Zhang, Cheng, Zhao Dong, Michael Doggett, and Shuang Zhao. "Antithetic Sampling for Monte Carlo Differentiable Rendering." ACM Transactions on Graphics 40, no. 4 (2021): 1–12.

[2] Zhang, Cheng, Zihan Yu, and Shuang Zhao. "Path-Space Differentiable Rendering of Participating Media." ACM Transactions on Graphics 40, no. 4 (2021): 1–15.

[3] Zhang, Cheng, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. "Path-Space Differentiable Rendering." ACM Transactions on Graphics 39, no. 4 (2020).

[4] Zhang, Cheng, and Shuang Zhao. "Multi-Scale Appearance Modeling of Granular Materials with Continuously Varying Grain Properties." In EGSR (DL), pp. 25-37. 2020.

[5] Zhang, Cheng, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. "A differential theory of radiative transfer." ACM Transactions on Graphics (TOG) 38, no. 6 (2019): 1-16.

[6] Xiao, Chang, Cheng Zhang, and Changxi Zheng. "Fontcode: Embedding information in text documents using glyph perturbation." ACM Transactions on Graphics (TOG) 37, no. 2 (2018): 1-16.

ABSTRACT OF THE DISSERTATION

Path-Space Differentiable Rendering

By

Cheng Zhang

Doctor of Philosophy in Computer Science

University of California, Irvine, 2022

Professor Shuang Zhao, Chair

Physics-based differentiable rendering—which is concerned with estimating derivatives of photorealistic rendering with respect to arbitrary scene parameters—has a diverse array of applications from solving inverse-rendering (aka. analysis-by-synthesis) problems to incorporating forward rendering into probabilistic-inference and machine-learning pipelines. Recently, great progress has been made in physics-based differentiable rendering. Unfortunately, most existing techniques lack the generality to support volumetric light transport and the efficiency to handle complex geometries and light transport effects.

To address these problems, we introduce in this dissertation a fundamentally new *path-space differentiable rendering* framework. Specifically, by differentiating the forward-rendering path integrals with respect to arbitrary scene parameters, we establish the mathematical formulation of *differential path integrals* that capture both interfacial and volumetric light transport.

Based on this formulation, we develop new unbiased and consistent differentiable rendering algorithms capable of efficiently handling challenging geometric discontinuities and light-transport phenomena such as soft shadows, interreflection, and caustics.

To further improve the robustness of our techniques, we leverage antithetic sampling to

efficiently differentiate glossy BSDFs and pixel reconstruction filters.

Lastly, we present the formulation of *differential image-loss path integrals* that expresses gradients of image losses as another form of differential path integrals. Based on this formulation, we develop a new approach that allows reverse-mode automatic differentiation to be integrated efficiently into our path-space differentiable rendering algorithms.

Chapter 1

Introduction

The goal of this dissertation is to develop general-purpose and efficient algorithms for *physics-based differentiable rendering*. To be specific, we expect the algorithms to be capable of numerically estimating derivatives of rendered images that exhibit complex light-transport effects (e.g., soft shadows, interreflection, and volumetric scatterings) with respect to arbitrary scene parameters (e.g., the shape or reflectance of an object, the position of a light source, and the pose of a camera). To this end, we concentrate on devising *differential* variants of mathematical formulations previously developed for *forward rendering*. These new differential formulations will then serve as the theoretical foundation for the development of new Monte Carlo differentiable-rendering methods.

In the past few years, significant progresses have been made in physics-based differentiable rendering. Starting with the realization that forward rendering is generally differentiable—even with the presence of discontinuities emerging from occlusions—general-purpose algorithms have been developed for interfacial [45, 49, 4] and volumetric [95] light transport. Unfortunately, these methods suffer from several fundamental limitations. First, some of them [45, 95] require detecting object silhouettes and, thus, scale poorly to scenes with com-

plex geometries. Second, all these methods rely on unidirectional path tracing—which is known to lack the robustness for handling many complex light-transport effects.

To overcome these limitations, in this dissertation, we establish the mathematical formulation of *differential path integrals*—the differential counterpart of (forward-rendering) path integrals [82, 65]. Based on this formulation, we introduce a family of path-space Monte Carlo algorithms which, compared with state-of-the-art methods, offer a new level of robustness for handling not only detailed geometries but also complex light-transport effects such as soft shadows, glossy reflection, indirect-dominated illumination, and caustics.

In the following, we begin with a detailed description of what *physics-based differentiable rendering* is and why it is important in §1.1. Then, we summarize the contributions of this dissertation in §1.2 and outline its organization in §1.3. Lastly, we provide an overview of related research works in §1.4.

1.1 Definition and Motivations

As a core research topic since the inception of computer graphics, *physics-based forward rendering* aims to synthesize photorealistic images from fully specified virtual scenes by simulating light transport in the scenes. On the contrary, *physics-based differentiable rendering* is concerned with computing derivatives of rendered images with respect to differential changes of virtual scenes. Mathematically, if we treat the forward-rendering process as a function that maps *scene parameters* into (photorealistic) images, the specific concern of differentiable rendering is estimating the derivatives of this function. In general, a scene parameter can be used to describe any aspect of a virtual 3D scene, including, but not limited to, the geometric representation (e.g., mesh vertices) and associated material properties (e.g., BSDF parameters) for all objects within the scene (see Figure 1.1)

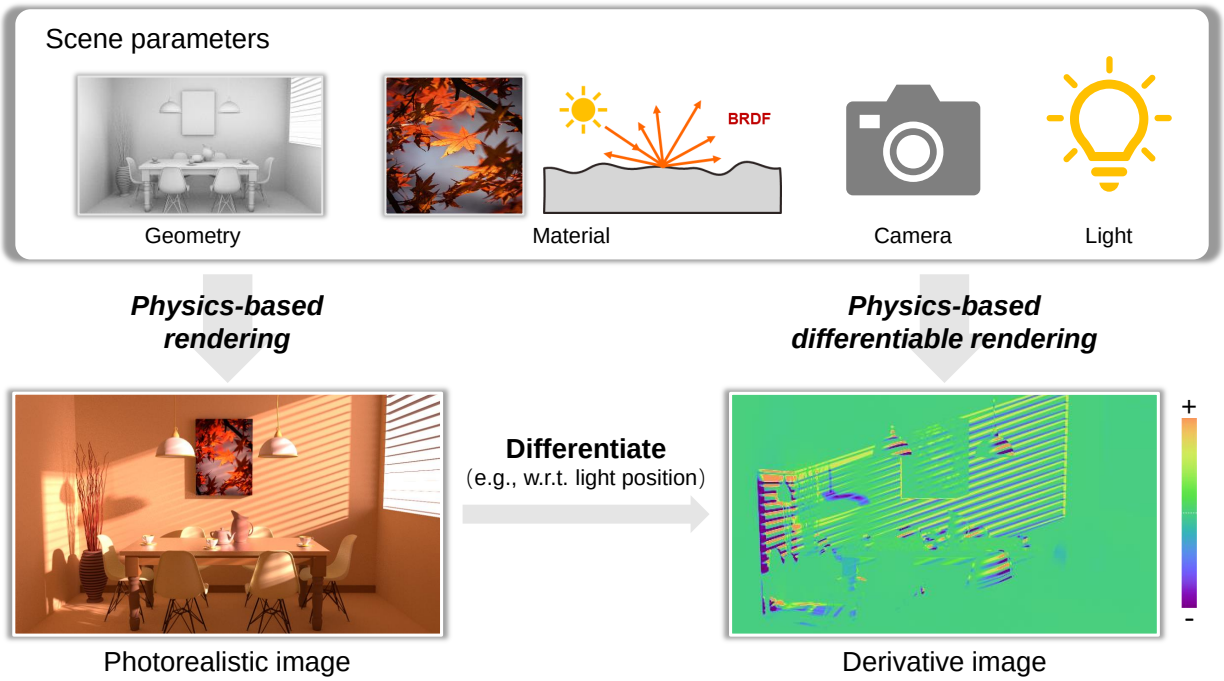


Figure 1.1: *Physics-based forward rendering* is about synthesizing photorealistic images based on fully described virtual scenes. *Physics-based differentiable rendering*, in contrast, is concerned with differentiating the forward-rendering process with respect to arbitrary scene parameters (e.g., the vertical position of the light source outside the window for this example).

One major application of physics-based differentiable rendering is solving *inverse-rendering* (aka. *analysis-by-synthesis*) problems: That is, the inference of scene parameters from photorealistic images. Efficient and robust solutions can benefit a wide array of applications such as computational fabrication, remote sensing, robotics, autonomous navigation, and architectural design.

In general, practical physics-based forward-rendering functions—which capture complicated interactions among shape, lighting, and material optical properties—cannot be inverted analytically. As a consequence, inverse rendering is typically formulated as optimization problems that seek scene parameters minimizing some predefined loss function that measures the difference between rendered and input images.

Differentiable rendering—which allows forward-rendering functions and, in turn, losses to be

differentiated with respect to scene parameters—is a key ingredient for applying gradient-based methods (e.g., stochastic gradient descent and its preconditioned variants) to inverse-rendering optimizations.

Differentiable rendering also opens the door for integrating forward rendering into *probabilistic-inference* and *machine-learning* pipelines. For instance, many data-driven models are trained using image losses computed by comparing rendered and target images. Physics-based forward and differentiable rendering allows the rendered images to capture complex optical phenomena that can, then, be learned during the model training process. It has been demonstrated by several recent works (e.g., [11]) that making a machine-learning model *physics-aware* can greatly improve its generalizability to novel conditions.

1.2 Our Contributions

In this dissertation, we introduce our *path-space differentiable rendering* (PSDR) framework that offers a new level of generality by supporting, for example, volumetric light transport, advanced (e.g., bidirectional) path sampling techniques, and differentiation with respect to arbitrary scene parameters (e.g., object geometries). Concretely, the technical contributions of this dissertation include:

- Establishing a new mathematical formulation by differentiating forward-rendering path integrals [81, 65] under the **material-form parameterization** with respect to arbitrary scene parameters. The resulting **differential path integrals** consist of completely separated *interior* and *boundary* components that can be estimated independently.

- Developing **path-space differentiable rendering algorithms** that provide unbiased and consistent estimates of the *interior* and *boundary* components of *differential path*

integrals. When estimating the *boundary* integrals, our technique avoids expensive silhouette detection [45, 95] without sacrificing unbiasedness.

- Introducing **antithetic sampling** in the context of *path-space differentiable rendering* for efficiently handling (i) glossy and near-specular BSDFs; and (ii) pixel reconstruction filters.

- Devising the formulation of **differential image-loss path integrals** that expresses gradients of image losses as *differential path integrals*. Based on this formulation, we develop **computational differentiation algorithms** to efficiently compute *interior* integrals by exploiting the layered structures of the corresponding computational graphs.

1.3 Organization of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we briefly revisit the mathematical preliminaries upon which our formulations and algorithms are built. In Chapter 3, we introduce the *material-form parameterization* which facilitates the derivation of our formulation of *differential path integrals* in Chapter 4. Based on this new formulation, we present in Chapter 5 new unbiased and consistent Monte Carlo estimators that efficiently estimate the *interior* and *boundary* components of *differential path integrals.*

To further improve the effectiveness of our techniques, we introduce *antithetic sampling* in Chapter 6 for efficient differentiation of glossy BSDFs and pixel reconstruction filters, and *differential image-loss path integrals* in Chapter 7 for scaling out to large numbers of scene parameters. Lastly, we discuss in Chapter 8 physics-based inverse rendering—a major application for our techniques—and show synthetic results.

Publications. This dissertation covers results from, but not limited to, our following publications:

- Zhang, Cheng, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. "Path-Space Differentiable Rendering." ACM Transactions on Graphics 39, no. 4 (2020).
- Zhang, Cheng, Zihan Yu, and Shuang Zhao. "Path-Space Differentiable Rendering of Participating Media." ACM Transactions on Graphics 40, no. 4 (2021): 1–15.
- Zhang, Cheng, Zhao Dong, Michael Doggett, and Shuang Zhao. "Antithetic Sampling for Monte Carlo Differentiable Rendering." ACM Transactions on Graphics 40, no. 4 (2021): 1–12.

1.4 Overview of Related Works

Derivatives for rendering. Analytical derivatives have been used in forward rendering to compute pixel footprints [29], handle specular light paths [13, 32], use Hamiltonian Monte Carlo to sample paths [46], and enable interactive editing of single-scattering albedo [27]. Arvo [2] presented an analytical method for calculating the gradients of irradiance in diffuse scenes. Ramamoorthi et al. [68] introduced a first-order analysis of light transport, focusing on effects such as soft shadows. All these works leverage certain type of derivatives in the image formation process to facilitate the efficiency and quality of forward rendering rather than the inverse problem.

Differentiable rendering in vision. Having derivatives of rendered images allows rendering to be efficiently integrated into probabilistic programming [43] and deep learning pipelines (e.g., as the decoder of an auto-encoder architecture [11]). Many recent works utilize various forms of rendering losses to regularize the training and improve generalization of

neural network models [88, 53, 73, 47, 11]. The differentiable renderers in the computer vision community are mostly based on the rasterization rendering model [48, 38, 44, 57, 26, 66] with restrictive simplification such as single-bounce illumination.

Physics-based differentiable rendering. Differentiable rendering has been used to solve *analysis-by-synthesis* problems in a wide range of applications including volumetric scattering [21, 20], cloth rendering [41], prefiltering of high-resolution volumes [97], appearance modeling of human teeth [83], fabrication of translucent materials [78], reflectance and lighting estimation [3, 59], and 3D reconstruction [79]. In order to address the specific requirements of the downstream tasks, all these methods compute radiance derivatives using algorithms specialized to specific light transport effects, with no regard to flexibility or generality.

A main challenge towards developing general-purpose differentiable rendering engines has been the differentiation with respect to scene geometry, which generally requires calculating additional boundary integrals. To address this problem, Li et al. [45] introduced a Monte Carlo edge-sampling method that provides unbiased estimates of these boundary integrals. This technique was then generalized in our DTRT framework [95] to handle volumetric light transport. All of these edge-sampling techniques do not scale well to complex geometry due to the necessity for run-time silhouette detection. To avoid sampling on discontinuity edges, one alternative for handling boundary integrals is to convert them into area integrals by reparameterizing the integrand [49, 4]. However, these reparamterization-based methods may lead to biased derivative estimates [49] and all require tracing auxiliary rays to detect discontinuity locations. Despite their ability to differentiate with respect to arbitrary scene parameterizations, all these methods are obtained by differentiating the rendering equation [36] (and the radiative transfer equation [10]), and rely on unidirectional path tracing for derivative estimations, which can be inefficient when handling complex light transport effects like concentrated indirect lighting and caustics.

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 [22, 86, 51] to obtain gradients of complex functions (e.g., neural networks). Recently, several general-purpose AD frameworks such as TensorFlow [1], PyTorch [64], Enoki [31], and Enzyme [55] have been developed. Further, systematic handling of discontinuities—which has been neglected by most AD frameworks—has been explored [5]. Most general-purpose differentiable rendering systems (e.g., Redner [45] and Mitsuba 2 [62]) utilize automatic differentiation, and ours is no exception. In this dissertation, we mainly focus on differentiable rendering theory and algorithms, which are orthogonal to the choice of automatic differentiation method.

Chapter 2

Preliminaries

In this chapter, we briefly review mathematical preliminaries on forward rendering and the differentiation of integrals—both of which will be used to devise the main results of this dissertation in later chapters.

Specifically, we first revisit in §2.1 important equations that specify steady-state radiance distributions under interfacial and volumetric light transport. These equations have led to the development of widely adopted forward-rendering methods like unidirectional path tracing.

Additionally, to facilitate the design of more advanced rendering methods such as bidirectional path tracing (BDPT) [82], techniques that formulate forward rendering as estimating high-dimensional *path integrals* have been introduced [81, 65]. We will present these formulations in §2.2.

Table 2.1 summarizes some commonly used notations in forward rendering.

Lastly, we present in §2.3.1 the *Reynolds transport theorem*, which has been the foundation of several recent differentiable rendering algorithms [45, 95]. We will derive many theoretical

Symbol	Definition
f_s	bidirectional scattering distribution function (BSDF)
σ_s	scattering coefficient
σ_t	extinction coefficient
f_p	single-scattering phase function
$T(\boldsymbol{x} \leftrightarrow \boldsymbol{y})$	transmittance between $\boldsymbol{x}$ and $\boldsymbol{y}$
W_e	sensor importance
$\mathcal{M}$	union of all object surfaces in the scene
$\mathcal{V}$	3D volume encapsulating the virtual scene
$\mathcal{V}_0$	volume interior $\mathcal{V}_0 := \mathcal{V} \setminus \mathcal{M}$
$\boldsymbol{x}_\mathcal{M}(\boldsymbol{x}, \boldsymbol{\omega})$	ray-casting function
$\boldsymbol{\Omega}$	path space/generalized path space
$G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})$	geometric term/generalized geometric term
$\bar{\boldsymbol{x}}$	light transport path
$f(\bar{\boldsymbol{x}})$	measurement contribution function generalized measurement contribution function

Table 2.1: List of commonly used symbols in forward rendering

results—including the establishment of the *differential path integral* formulation—using this theorem. Additionally, we provide a brief discussion in §2.3.2 about *automatic differentiation* which, as the bedrock of differentiable programming, is an important tool for evaluating the derivatives of a function specified by a computer program. We will use it to implement our path-space differentiable rendering algorithms.

2.1 Light Transport Equations

We now review the key equations that model the propagation of light in virtual scenes. Specifically, we first present the *rendering equation* (§2.1.1) that governs light interaction with surfaces. Then, we cover the *radiative transfer equation* (§2.1.2) that captures light propagation in participating media.

2.1.1 Rendering Equation

In physisc-based rendering, the **bidirectional scttering distribution function** (BSDF) is a mathematical description of light-scattering properties of a surface. At a specific surface point $\boldsymbol{x}$, BSDF is a 4D function of the incident and scattered radiance:

$$f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{\mathrm{d}L_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{o}})}{L_{\mathrm{i}}(\boldsymbol{\omega}_{\mathrm{i}})\,\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathrm{i}}),} \tag{2.1}$$

where $\sigma^{\perp}$ is the measure of projected solid angle. Integrating both sides of the equation $\mathrm{d}L_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{o}}) = L_{\mathrm{i}}(\boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}})\,\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathrm{i}})$ over the unit sphere $\mathbb{S}^2$ yields the **scattering equation** [14, 81]

$$L_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{o}}) = \int_{\mathbb{S}^2} L_{\mathrm{i}}(\boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{s}}(\boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}})\,\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathrm{i}}), \tag{2.2}$$

which describes the *local* scattering of light at a single point.

Rendering equation. Provided Eq. (2.2), one can express the outgoing radiance L_{o} as the sum of emitted radiance L_{e} and scattered radiance L_{s}:

$$\begin{aligned} L_{\mathrm{o}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) &= L_{\mathrm{e}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) + L_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) \\ &= L_{\mathrm{e}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) + \int_{\mathbb{S}^2} L_{\mathrm{i}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{i}})\,\mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathrm{i}}). \end{aligned} \tag{2.3}$$

In geometric optics, radiance remains constant along rays of light through empty space:

$$L_{\mathrm{i}}(\boldsymbol{x}, \boldsymbol{\omega}) = L_{\mathrm{o}}(\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}), -\boldsymbol{\omega}), \tag{2.4}$$

where $\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega})$ is the **ray-casting function** that returns the first intersection (or "hit") between the ray originated at $\boldsymbol{x}$ with direction $\boldsymbol{\omega}$ and **union of all object surfaces** $\mathcal{M}$.

That is, $\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}) := \boldsymbol{x} + t_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega})\, \boldsymbol{\omega}$, where

$$t_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}) := \inf\{t > 0 \ : \ \boldsymbol{x} + t\, \boldsymbol{\omega} \in \mathcal{M}\}. \tag{2.5}$$

Based on Eq. (2.4), Eq. (2.3) can be rewritten as

$$L(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}) + \int_{\mathbb{S}^2} L(\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}}), \boldsymbol{\omega}_{\mathrm{i}})\, f_{\mathrm{s}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{i}})\, \mathrm{d}\sigma^{\perp}(\boldsymbol{\omega}_{\mathrm{i}}), \tag{2.6}$$

which is the well known **rendering equation** [36]. Since L_{i} does not appear in this equation, we drop the subscript of L_{o} for notational simplicity. Even though the spherical integral in the rendering equation (2.6) appears similar to the scattering equation (2.2), the former is conditioned on the global distribution of radiance (at equilibrium) and is an integral equation where the unknown quantity L appears on both sides. The form of this equation lends itself to recursive numerical solutions using Monte Carlo methods.

2.1.2 Radiative Transfer Equation

The radiative transfer theory (RTT) [10] has been used in natural science and application areas, including biomedical imaging [30], neutron transport [67], material scence [80], remote sensing and astrophysics [54]. This framework was later introduced to computer graphics [7, 37, 71] to render participating media and translucent materials. At the core of RTT is the **radiative transfer equation** (RTE). In its integral form, the RTE states that

$$\begin{aligned} L(\boldsymbol{x}, \boldsymbol{\omega}) = {} & T(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x})\, L(\boldsymbol{x}_0, \boldsymbol{\omega}) \\ & + \int_0^R T(\boldsymbol{x}' \leftrightarrow \boldsymbol{x})\, \sigma_{\mathrm{s}}(\boldsymbol{x}') \int_{\mathbb{S}^2} f_{\mathrm{p}}(\boldsymbol{x}', \boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega})\, L(\boldsymbol{x}', \boldsymbol{\omega}_{\mathrm{i}})\, \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{i}})\, \mathrm{d}r. \end{aligned} \tag{2.7}$$

where σ denotes the solid-angle measure, $\boldsymbol{x}' := \boldsymbol{x} - r\, \boldsymbol{\omega}$, and $\boldsymbol{x}_0 := \boldsymbol{x} - R\, \boldsymbol{\omega}$ with $R = t_{\mathcal{M}}(\boldsymbol{x}, -\boldsymbol{\omega})$ being the distance along the ray originated at $\boldsymbol{x}$ with the direction $-\boldsymbol{\omega}$ before

intersecting[1] the medium's boundary $\mathcal{M}$. The radiance $L(\boldsymbol{x}_0, \boldsymbol{\omega})$ is considered a *boundary condition* of the RTE and typically estimated using the rendering equation (2.6) in practice.

Lastly, in Eq. (2.7), σ_s and f_p denote, respectively, the medium's **scattering coefficient** and **single-scattering phase function**, and the **transmittance** $T(\boldsymbol{x}', \boldsymbol{x})$ specifies the fraction of light that travels from $\boldsymbol{x}'$ to $\boldsymbol{x}$ without being absorbed or scattered and equals

$$T(\boldsymbol{x}' \leftrightarrow \boldsymbol{x}) = \exp\left(-\int_0^\tau \sigma_\mathrm{t}(\boldsymbol{x} - \tau'\boldsymbol{\omega})\,\mathrm{d}\tau'\right),$$

where σ_t is the medium's **extinction coeffient** (or **optical density**) and describes how frequently light interacts with the medium (by being absorbed or scattered).

Numerical solutions. In general, neither the rendering equation (2.6) nor the radiative transfer equation (2.7) has analytical solutions. Instead, these equations are typically solved *numerically* using Monte Carlo methods—a long-standing research topic on forward rendering. One of the most widely adopted solutions is *unidirectional path tracing*.

2.2 Path Integral Formulations

Physics-based rendering typically amounts to estimating the **response of a radiometric sensor** formulated as an integral of incident radiance over the surface of the sensor, modulated by a **sensor importance** function W_e:

$$I = \int_{\mathcal{M}\times\mathbb{S}^2} W_\mathrm{e}(\boldsymbol{x}, \boldsymbol{\omega})\, L_\mathrm{i}(\boldsymbol{x}, \boldsymbol{\omega})\,\mathrm{d}A(\boldsymbol{x})\,\mathrm{d}\sigma^\perp(\boldsymbol{\omega}). \tag{2.8}$$

[1]When the medium in unbounded and an intersection does not exist, $R = +\infty$ and the first term on the right-hand side of Eq. (2.7) vanishes.

In practice, rendering an image amounts to estimating one response I_j for each pixel j with a distinct importance W_{e}^j.

In the following, we present mathematical formulations that express the radiometric responses I in Eq. (2.8) as high-dimensional *path integrals*. Specifically, we will first revisit the interfacial variant [81] in §2.2.1, and then provide the generalized version [65] capable of handling volumetric light transport in §2.2.2. A main result of this dissertation—which we will introduce in Chapter 4—is the *differential* counterparts of these integrals with respect to arbitrary scene parameters.

2.2.1 Interfacial Path Integral

We now show how interfacial (i.e., surface-only) light transport governed by the rendering equation (2.6) can be re-expressed as an integration problem. Our presentation roughly follows Chapter 8 of Eric Veach's Ph.D. thesis [81].

Rendering equation in three-point form. The rendering equation (2.6) can be reformulated as an integral over object surfaces $\mathcal{M}$ as opposed to the unit sphere $\mathbb{S}^2$. To be specific, for any surface points $\boldsymbol{x}, \boldsymbol{x}', \boldsymbol{x}'' \in \mathcal{M}$, we have

$$L(\boldsymbol{x}' \to \boldsymbol{x}'') = L_{\mathrm{e}}(\boldsymbol{x}' \to \boldsymbol{x}'') + \int_{\mathcal{M}} L(\boldsymbol{x} \to \boldsymbol{x}')\, f_{\mathrm{s}}(\boldsymbol{x} \to \boldsymbol{x}' \to \boldsymbol{x}'')\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}')\, \mathrm{d}A(\boldsymbol{x}). \tag{2.9}$$

In Eq. (2.9), $L(\boldsymbol{x} \to \boldsymbol{x}')$ denotes the radiance leaving $\boldsymbol{x}$ toward $\boldsymbol{x}'$. That is, $L(\boldsymbol{x} \to \boldsymbol{x}') := L(\boldsymbol{x}, \overrightarrow{\boldsymbol{x}\,\boldsymbol{x}'})$ where $\overrightarrow{\boldsymbol{x}\,\boldsymbol{x}'} := \frac{\boldsymbol{x}'-\boldsymbol{x}}{\|\boldsymbol{x}'-\boldsymbol{x}\|}$ indicates the unit vector pointing from $\boldsymbol{x}$ to $\boldsymbol{x}'$. $L(\boldsymbol{x}' \to \boldsymbol{x}'')$ is defined in a similar fashion. Additionally, $f_{\mathrm{s}}(\boldsymbol{x} \to \boldsymbol{x}' \to \boldsymbol{x}'') := f_{\mathrm{s}}(\boldsymbol{x}, \overrightarrow{\boldsymbol{x}'\,\boldsymbol{x}}, \overrightarrow{\boldsymbol{x}'\,\boldsymbol{x}''})$. Lastly, the **geometric term** G captures the change of measure from projected solid angle

in Eq. (2.6) to the surface area in Eq. (2.9):

$$G(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) := \mathbb{V}(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) \frac{\left|\boldsymbol{n}(\boldsymbol{y}) \cdot \overrightarrow{\boldsymbol{x}\boldsymbol{y}}\right| \left|\boldsymbol{n}(\boldsymbol{y}) \cdot \overrightarrow{\boldsymbol{x}\boldsymbol{y}}\right|}{\|\boldsymbol{x} - \boldsymbol{y}\|^2}, \tag{2.10}$$

where $\mathbb{V}(\boldsymbol{x}, \boldsymbol{y})$ is the **mutual visibility function** that equals to one when $\boldsymbol{x}$ and $\boldsymbol{y}$ are mutually visible and zero otherwise.

Applying a similar change of measure to Eq. (2.8) yields:

$$I = \int_{\mathcal{M} \times \mathcal{M}} W_\mathrm{e}(\boldsymbol{x} \to \boldsymbol{x}')\, L(\boldsymbol{x} \to \boldsymbol{x}')\, G(\boldsymbol{x} \leftrightarrow \boldsymbol{x}')\, \mathrm{d}A(\boldsymbol{x})\, \mathrm{d}A(\boldsymbol{x}'), \tag{2.11}$$

where $W_\mathrm{e}(\boldsymbol{x} \to \boldsymbol{x}') := W_\mathrm{e}(\boldsymbol{x}', \overrightarrow{\boldsymbol{x}'\boldsymbol{x}})$. We note that, in Eq. (2.11), $\boldsymbol{x}'$ resides on the surface of the sensor and $\boldsymbol{x} \to \boldsymbol{x}'$ follows the direction of light flow.

Path integral formulation. By recursively expanding $L(\boldsymbol{x} \to \boldsymbol{x}')$ in Eq. (2.11) using the three-point rendering equation (2.9), the sensor response can be represented as

$$I = \sum_{N=1}^{\infty} \int_{\boldsymbol{\Omega}_N} f(\bar{\boldsymbol{x}})\, \mathrm{d}\mu_N(\bar{\boldsymbol{x}}), \tag{2.12}$$

where $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ denotes a **light path** with $\boldsymbol{x}_0$ on a light source and $\boldsymbol{x}_N$ on the sensor, $\boldsymbol{\Omega}_N := \mathcal{M}^{N+1}$, and $\mathrm{d}\mu_N(\bar{\boldsymbol{x}}) := \prod_{n=0}^{N} \mathrm{d}A(\boldsymbol{x}_n)$ is the area-product measure. Additionally, f is the **measurement contribution function** given by

$$\begin{aligned} f(\bar{\boldsymbol{x}}) := {} & L_\mathrm{e}(\boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_0 \leftrightarrow \boldsymbol{x}_1) \\ & \left(\prod_{n=1}^{N-1} f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})\, G(\boldsymbol{x}_n \leftrightarrow \boldsymbol{x}_{n+1}) \right) W_\mathrm{e}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N). \end{aligned} \tag{2.13}$$

By expressing the emission L_e, the sensor importance W_e, and the BSDF f_s in a unified

fashion via:

$$f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}) := \begin{cases} f_{\mathrm{s}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}), & (0 < n < N) \\ L_{\mathrm{e}}(\boldsymbol{x}_0 \to \boldsymbol{x}_1), & (n = 0) \\ W_{\mathrm{e}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N), & (n = N) \end{cases} \tag{2.14}$$

the measurement contribution in Eq. (2.13) can be simplified to

$$f(\bar{\boldsymbol{x}}) := \left(\prod_{n=0}^{N} f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}) \right) \left(\prod_{n=1}^{N} G(\boldsymbol{x}_{n-1} \leftrightarrow \boldsymbol{x}_n) \right). \tag{2.15}$$

Further, the **path space** $\boldsymbol{\Omega}$ is defined as the space containing all light paths with finite lengths

$$\boldsymbol{\Omega} := \bigcup_{N=1}^{\infty} \boldsymbol{\Omega}_N, \tag{2.16}$$

with the associated **area-product measure** μ provided by

$$\mu(D) := \sum_{N=1}^{\infty} \mu_N(\boldsymbol{\Omega}_N \cap D), \tag{2.17}$$

for any $D \subset \boldsymbol{\Omega}$. Then, the sum over different path lengths in Eq. (2.12) can be omitted, yielding

$$I = \int_{\boldsymbol{\Omega}} f(\bar{\boldsymbol{x}}) \, \mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{2.18}$$

which is well known **path integral** formulation introduced by Veach [81].

2.2.2 Generalized Path Integral

To capture volumetric light transport governed by the radiative transfer equation (RTE) (2.7), Pauly et al. [65] have introduced a generalization of the path integral formulation presented in §2.2.1. In the following, we provide a brief derivation of this generalized formulation.

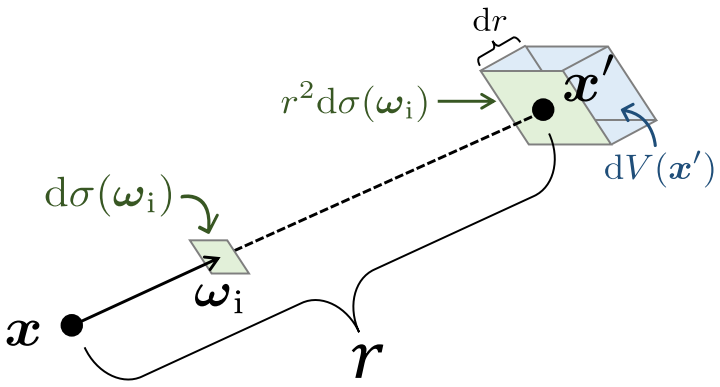


Figure 2.1: Illustration for the change of variables given by Eq. (2.19).

Change of measure. Similar to the derivation of the interfacial path integral (2.18), it is desired to rewrite the double integral on the right-hand side of the RTE (2.7) as a single *volume integral*, which can be achieved based on the relation

$$\mathrm{d}V(\boldsymbol{x}') = (r^2\,\mathrm{d}\sigma(\boldsymbol{\omega}_\mathrm{i}))\,\mathrm{d}r, \tag{2.19}$$

as illustrated in Figure 2.1.

Then, *generalized path integrals* can be obtained by recursively expanding the radiance term $L(\boldsymbol{x}', \boldsymbol{\omega}_\mathrm{i})$ in the RTE (2.7) and applying the change of measure in Eq. (2.19). Please refer to Chapter 3 of Wenzel Jacob's Ph.D. thesis [34] for a detailed explanation of this process.

Generalized path integral. Let $\mathcal{V} \subset \mathbb{R}^3$ be a **3D volume** that encapsulates the virtual scene, with its boundary being the scene surfaces $\mathcal{M}$ and its **interior** denoted by $\mathcal{V}_0 := \mathcal{V} \backslash \mathcal{M}$. At a high level, a **generalized path integral** takes the same form as the interfacial variant in Eq. (2.18):

$$I = \int_{\boldsymbol{\Omega}} f(\bar{\boldsymbol{x}})\,\mathrm{d}\mu(\bar{\boldsymbol{x}}), \tag{2.20}$$

with a major distinction that, in a light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, each $\boldsymbol{x}_n$ can be either a **surface vertex** (i.e., $\boldsymbol{x}_n \in \mathcal{M}$) or a **volume vertex** (i.e., $\boldsymbol{x}_n \in \mathcal{V}_0$). Due to this distinc-

tion, definitions of the path space $\mathbf{\Omega}$, the measure $\mu(\bar{\boldsymbol{x}})$ and the measurement contribution function $f(\bar{\boldsymbol{x}})$ need to be generalized accordingly as follows.

Generalized path space and measure. The type of individual vertices in a light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ can be characterized using the **path characteristic** l, an $(N+1)$-bit integer that encodes the type of individual vertices. Specifically, the n-th bit of the binary representation of l, which we denote as $b_n(l)$, equals one if $\boldsymbol{x}_n$ is a surface vertex and zero if it is a volume vertex.

For all $N \geq 1$ and $0 \leq l < 2^{N+1}$, the set of all paths with N segments and characteristic l is

$$\mathbf{\Omega}_N^l := \left\{ (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N) \ : \begin{array}{ll} \boldsymbol{x}_n \in \mathcal{M} & \text{if } b_n(l) = 1 \\ \boldsymbol{x}_n \in \mathcal{V}_0 & \text{if } b_n(l) = 0 \end{array} \right\}, \tag{2.21}$$

and the Lebesgue measure μ_N^l on $\mathbf{\Omega}_N^l$ is defined by

$$\mathrm{d}\mu_N^l(\bar{\boldsymbol{x}}) := \prod_{n=0}^{N} \mathrm{d}\mu_{N,n}^l(\boldsymbol{x}_n), \tag{2.22}$$

where

$$\mathrm{d}\mu_{N,n}^l(\boldsymbol{x}_n) := \begin{cases} \mathrm{d}A(\boldsymbol{x}_n), & (b_n(l) = 1) \\ \mathrm{d}V(\boldsymbol{x}_n). & (b_n(l) = 0) \end{cases} \tag{2.23}$$

Provided Eqs. (2.21) and (2.22), the **generalized path space** $\mathbf{\Omega}$ and the **associated measure** μ in Eq. (2.20) can be defined, respectively, as

$$\mathbf{\Omega} := \bigcup_{N \geq 1} \bigcup_{l=0}^{2^{N+1}-1} \mathbf{\Omega}_N^l, \tag{2.24}$$

$$\mu(D) := \sum_{N \geq 1} \sum_{l=0}^{2^{N+1}-1} \mu_N^l \left(D \cap \mathbf{\Omega}_N^l \right) \quad \text{for any } D \subset \mathbf{\Omega}. \tag{2.25}$$

Measurement contribution. Similar to the interfacial case, in generalized path integrals (2.20), the **(generalized) measurement contribution** f is still expressed as the product of per-vertex contributions f_{v} and per-segment ones G:

$$f(\bar{\boldsymbol{x}}) := \left(\prod_{n=0}^{N} f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})\right)\left(\prod_{n=1}^{N} G(\boldsymbol{x}_{n-1} \leftrightarrow \boldsymbol{x}_n)\right). \tag{2.26}$$

To handle volume light transport, the per-vertex term $f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ is generalized for volume vertices:

$$f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}) := \begin{cases} f_{\mathrm{s}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}), & (0 < n < N \quad \text{and} \quad \boldsymbol{x}_n \in \mathcal{M}) \\ \sigma_{\mathrm{s}}(\boldsymbol{x}_n)\, f_{\mathrm{p}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}), & (0 < n < N \quad \text{and} \quad \boldsymbol{x}_n \in \mathcal{V}_0) \\ L_{\mathrm{e}}(\boldsymbol{x}_0 \to \boldsymbol{x}_1), & (n = 0) \\ W_{\mathrm{e}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N). & (n = N) \end{cases} \tag{2.27}$$

Moreover, the per-segment $G(\boldsymbol{x} \leftrightarrow \boldsymbol{y})$ becomes the **generalized geometric term** defined as

$$G(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) := \mathbb{V}(\boldsymbol{x} \leftrightarrow \boldsymbol{y})\, T(\boldsymbol{x} \leftrightarrow \boldsymbol{y}) \frac{D_{\boldsymbol{x}}(\boldsymbol{y})\, D_{\boldsymbol{y}}(\boldsymbol{x})}{\|\boldsymbol{x} - \boldsymbol{y}\|^2}, \tag{2.28}$$

where

$$D_{\boldsymbol{x}}(\boldsymbol{y}) := \begin{cases} \left|\boldsymbol{n}(\boldsymbol{x}) \cdot \overrightarrow{\boldsymbol{xy}}\right|, & (\boldsymbol{x} \in \mathcal{M}) \\ 1. & (\boldsymbol{x} \in \mathcal{V}_0) \end{cases} \tag{2.29}$$

Significance of path integrals. The path integrals in Eqs. (2.18) and (2.20) operate on entire light transport paths—as opposed to radiance fields in the rendering (2.6) and radiative transfer equations (2.7)—and essentially provide a "global" perspective of light transport. Consequently, the path integral formulations have enabled the development of a wide array of advanced forward-rendering techniques such as bidirectional path tracing (BDPT) [82], vertex connection and merging (VCM) [18], as well as various Markov-Chain

Monte Carlo methods [81, 32, 65, 39, 12, 24].

2.3 Differentiating Integrals

Physics-based forward rendering, as demonstrated in §2.1 and §2.2, largely amounts to solving integral equations like Eqs. (2.6) and (2.7), or estimating path integrals defined in Eqs. (2.18) and (2.20). Consequently, a crucial ingredient for physics-based differentiable rendering is the differentiation of various forms (e.g., spherical, area, and path) of integrals.

In what follows, we present *Reynolds transport theorem*—a general mathematical tool for differentiating integrals in §2.3.1. Additionally, in §2.3.2, we provide a brief discussion on *automatic differentiation*—an important computational tool for differentiable programming.

2.3.1 Reynolds Transport Theorem

A problem we will encounter repeatedly in this dissertation is calculating derivatives of integrals with respect to some parameter θ:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\int_{\Omega} h(\boldsymbol{u})\,\mathrm{d}\mu(\boldsymbol{u}) = ? \tag{2.30}$$

where both the domain of integration Ω and the integrand h can depend on θ in general.

Conventionally, Eq. (2.30) is usually calculated by exchanging the differentiation and integration operations:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\int_{\Omega} h(\boldsymbol{u})\,\mathrm{d}\mu(\boldsymbol{u}) \overset{?}{=} \int_{\Omega} \frac{\mathrm{d}}{\mathrm{d}\theta} h(\boldsymbol{u})\,\mathrm{d}\mu(\boldsymbol{u}), \tag{2.31}$$

where the right-hand side can be computed based on standard numerical methods (e.g., Monte Carlo) and differentiable evaluation of the integrand h.

Although Eq. (2.31) works adequately in many cases, it can break when the integrand h contains (jump) discontinuities that depend on the parameter θ. For example, we consider the problem of differentiating the following Riemann integral:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\int_0^1 \mathcal{H}(u-\theta)\,\mathrm{d}u = ? \tag{2.32}$$

where $\mathcal{H}$ is the Heaviside step function defined as, for any $x \in \mathbb{R}$,

$$\mathcal{H}(x) := \begin{cases} 1, & (x > 0) \\ 0. & (x \leq 0) \end{cases} \tag{2.33}$$

Noting that $\int_0^1 \mathcal{H}(u-\theta)\,\mathrm{d}u = \int_\theta^1 \mathrm{d}u = (1-\theta)$, Eq. (2.32) can be calculated analytically:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\int_0^1 \mathcal{H}(u-\theta)\,\mathrm{d}u = \frac{\mathrm{d}}{\mathrm{d}\theta}(1-\theta) = -1. \tag{2.34}$$

On the contrary, since $\frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{H}(u-\theta) \equiv 0$, differentiating the integrand leads to an *incorrect* result:

$$\int_0^1 \frac{\mathrm{d}}{\mathrm{d}\theta}\mathcal{H}(u-\theta)\,\mathrm{d}u = 0 \neq \frac{\mathrm{d}}{\mathrm{d}\theta}\int_0^1 \mathcal{H}(u-\theta)\,\mathrm{d}u. \tag{2.35}$$

We note that many, if not most, integrals in physics-based rendering—including those in the rendering (2.6) and radiative transfer equations (2.7) as well as path integrals (2.18, 2.20)—have discontinuous integrands. These discontinuities emerge partially from *occlusion*. The ray-casting function $\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}_\mathrm{i})$ in the rendering equation (2.6), for instance, generally contains jump discontinuous with respect to $\boldsymbol{\omega}_\mathrm{i}$ when there is an object occluding another (see Figure 2.2).

To correctly handle discontinuities integrands, we resort to the *Reynolds transport theorem* [70]—which originated in fluid mechanics and is a generalization of Leibniz's rule for differentiation [17]. In the following, we provide a brief description of this theorem.

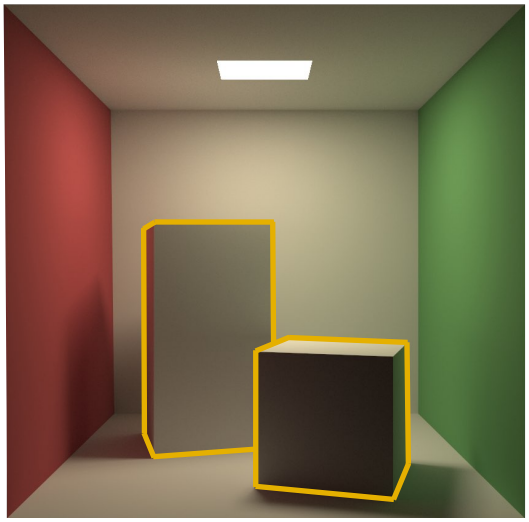


Figure 2.2: In the rendering equation (2.6), the discontinuities partially come from the mutal occlusion between scene objects. In this example, the ray-casting function $\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}, \boldsymbol{\omega}_{\mathrm{i}})$ is discontinuous with respect to $\boldsymbol{\omega}_{\mathrm{i}}$ at the visibility boundary (denoted in yellow).

Theorem 1: Reynolds transport theorem

Let h be a scalar-valued function defined on some n-dimensional manifold $\Omega(\theta)$ parameterized with some $\theta \in \mathbb{R}$. Additionally, let $\Gamma(\theta) \subset \Omega(\theta)$ be an $(n-1)$-dimensional manifold, which we term as the **extended boundary** of $\Omega(\theta)$, given by the union of the *external* boundary $\partial\Omega(\theta)$ and the *internal* one comprised of jump discontinuity points of the integrand h. Then, it holds that

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\left(\int_{\Omega(\theta)} h \,\mathrm{d}\Omega(\theta)\right) = \int_{\Omega(\theta)} \frac{\mathrm{d}h}{\mathrm{d}\theta}\,\mathrm{d}\Omega(\theta) + \int_{\Gamma(\theta)} \left(\boldsymbol{n} \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\theta}\right) \Delta h \,\mathrm{d}\Gamma(\theta), \tag{2.36}$$

where $\mathrm{d}\Omega$ and $\mathrm{d}\Gamma$ respectively denote the standard measures associated with Ω and Γ; and $\boldsymbol{n}$ is the unit-normal field associated with $\Gamma(\theta)$. Further, for all $\boldsymbol{x} \in \Gamma(\theta)$, Δh equals[a]

$$\Delta h(\boldsymbol{x}) := \lim_{\epsilon \to 0^-} h(\boldsymbol{x} + \epsilon\, \boldsymbol{n}) - \lim_{\epsilon \to 0^+} h(\boldsymbol{x} + \epsilon\, \boldsymbol{n}). \tag{2.37}$$

[a] When $\boldsymbol{x}$ approaches $\Gamma(\theta)$ from the exterior of the domain $\Omega(\theta)$, the corresponding one-sided limit in Eq. (2.37) is set to zero.

Intuitively, Theorem 1 states that the derivative of the integral of a function h over some domain $\Omega(\theta)$ equals the sum of:

1. *(The first integral)* The derivative of h itself integrated over the same domain $\Omega(\theta)$;
2. *(The second integral)* The scalar change rate, or "normal velocity", of the extended boundary $\Gamma(\theta)$ along the normal direction, modulated by the differences between function values across $\Gamma(\theta)$.

In the rest of the dissertation, we refer to the first integral on the right-hand side of Eq. (2.36) as the **interior** term and the second one as the **boundary** term.

As a special case, for classical Riemann integrals where Ω is an interval $(a, b) \subset \mathbb{R}$ and the integrand h is differentiable everywhere, it holds that $\Gamma = \{a, b\}$, and the boundary integral in Eq. (2.36) reduces to the sum of the integrand evaluated at a and b. Assuming the boundary normal $\boldsymbol{n}$ to point toward the positive direction of the x-axis, the scalar normal velocity $\left(\boldsymbol{n} \cdot \frac{\mathrm{d}\boldsymbol{x}}{\mathrm{d}\theta}\right)$ in Eq. (2.36) becomes $\frac{\mathrm{d}a}{\mathrm{d}\theta}$ at $x = a$ and $\frac{\mathrm{d}b}{\mathrm{d}\theta}$ at $x = b$. Then, it is easy to verify that $\Delta h(b(\theta), \theta) = h(b(\theta), \theta)$ and $\Delta h(a(\theta), \theta) = -h(a(\theta), \theta)$, yielding

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}\int_{a(\theta)}^{b(\theta)} h(x,\theta)\,\mathrm{d}x = &\int_{a(\theta)}^{b(\theta)} \frac{\mathrm{d}}{\mathrm{d}\theta} h(x,\theta)\,\mathrm{d}x \\ &+ \left(\frac{\mathrm{d}}{\mathrm{d}\theta} b(\theta)\right) h(b(\theta),\theta) - \left(\frac{\mathrm{d}}{\mathrm{d}\theta} a(\theta)\right) h(a(\theta),\theta),\end{aligned}$$

which is known as Leibniz's rule for differentiation [17].

We will utilize Theorem 1 to devise *differential* variants of forward-rendering path integrals of Eq. (2.18) and (2.20) in Chapter 4. Furthermore, based on these *differential path integrals*, we will introduce new efficient Monte Carlo methods for path-space differentiable rendering in Chapter 5.

2.3.2 Automatic Differentiation

Automatic differentiation (AD) refers to a set of techniques that take a program which computes a value, and automatically construct a procedure for computing derivatives of that value. In an AD system, the computations in a program are decomposed into a sequence of *primitive operations* which have specified routine for computing derivatives. For a thorough review of automatic differentiation techniques, we refer to the book by Griewank and Walther [22].

According to the order in which the derivatives are calculated, conventional AD algorithms are typically categorized into two distinct types: **forward-mode** and **reverse-mode** AD.

Forward mode. *Forward-mode* AD algorithms propagate derivatives from inputs to outputs (i.e., following the flow of ordinary computations) by keeping track of derivatives with respect to every input variable through individual arithmetic operations (based on the primitive differentiation rules). In practice, *forward-mode* AD is most efficient for differentiating functions with low-dimensional inputs and high-dimensional outputs.

Reverse mode. On the other hand, *reverse-mode* AD algorithms evaluate the chain rule in the reversed order (i.e., from outputs to inputs). This is accomplished by recording a transcript of operations known as the **computation graph** in the *forward pass* that runs the ordinary program. Then, in a subsequent *backward pass*, gradients are back-propagated through the computation graph by applying the chain rule at individual graph nodes.

In contrast to the forward mode, *reverse-mode* AD is ideal for differentiating functions with high-dimensional inputs and low-dimensional outputs. This is the case for many applications in computer graphics, vision, and machine learning. Consequently, many widely adopted AD frameworks such as TensorFlow [1], PyTorch [64], Enoki [31], and Enzyme [55] focus on reverse-

mode AD. Our path-space differentiable rendering algorithms, which we will introduce in later chapters, are also better suited with reverse-mode AD.

Chapter 3

Material-Form Parameterization

In this chapter, to facilitate the differentiation of (forward-rendering) path integrals, we introduce the *material-form parameterization* (§3.1) that reformulates surface or volume integrals over evolving domains to those over constant (i.e., non-evolving) domains. We further show how generalized path integrals can be re-expressed in *material forms* (§3.2), which can be differentiated conveniently using Reynolds transport theorem. Lastly, we provide more details about our implementation of the *material-form parameterization* (§3.3). Table 3.1 summarizes the commonly used symbols and their definitions in the material-form parameterization.

Even though it is also possible to differentiate path integrals directly without any reparameterization, which we have discussed in one of our recent works [94], *material-form path integrals* can be differentiated more easily and requires fewer types of discontinuities to be handled. In this dissertation, we focus on the material-form formulation.

Symbol	Definition
θ	abstract scene parameter
$\mathcal{M}(\theta)$	evolving surface
$\mathcal{V}(\theta)$	evolving volume
$\mathcal{V}_0(\theta)$	$\mathcal{V}(\theta) \setminus \mathcal{M}(\theta)$
$\mathtt{X}$	motion
$\mathtt{P}$	reference map
$\boldsymbol{p}$	material point
$\boldsymbol{x}$	spatial point
$\mathcal{B}_{\mathcal{M}}$	reference surface
$\mathcal{B}_{\mathcal{V}}$	reference volume
$\mathcal{B}_{\mathcal{V}_0}$	$\mathcal{B}_{\mathcal{V}} \setminus \mathcal{B}_{\mathcal{M}}$
$\bar{\boldsymbol{p}}$	material light path
$\hat{\boldsymbol{\Omega}}$	material path space
$\bar{\mathtt{X}}$	path-level mapping
$\hat{f}$	material measurement contribution function

Table 3.1: List of commonly used symbols in material-form parameterization

3.1 Material-Form Integrals

In this section, we revisit some concepts (e.g., motion and references) that originated from fluid mechanics. Based on these concepts, we introduce the *material-form parameterization* that can be applied to a surface or volume integral with its integration domain evolving based on some **abstract scene parameter** θ.

3.1.1 Evolving Surfaces and Volumes

In §2.2.2, we describe the geoemtry of a 3D virtual scene using the encapsulating volume $\mathcal{V} \subset \mathbb{R}^3$, along with the union of object surfaces $\mathcal{M} \subset \mathcal{V}$. Now, we consider the case when both the volume $\mathcal{V}$ and the surfaces $\mathcal{M}$ evolve under some parameter $\theta \in \mathbb{R}$. In addition, we define $\mathcal{V}_0(\theta) := \mathcal{V}(\theta) \setminus \mathcal{M}(\theta)$ as the interior of the evolving volume $\mathcal{V}$.

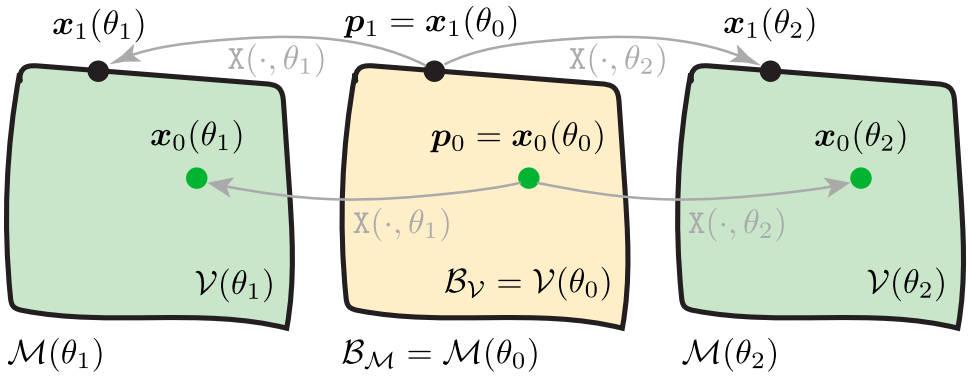


Figure 3.1: **Material-form parameterization** of a block whose horizontal location is controlled by a parameter θ. In this example, the reference surface $\mathcal{B}_\mathcal{M}$ and volume $\mathcal{B}_\mathcal{V}$ are selected as the block at some fixed $\theta = \theta_0$ (illustrated in yellow). Then, the motion $\mathtt{X}$ captures the motion of the block (hence the name) by mapping each point in the reference volume to the corresponding one in the "moving" block (illustrated in green) via $\mathtt{X}(\cdot, \theta)$ for any θ.

In fluid mechanics [9, 23], *material-form parameterization* has been introduced to efficiently express the evolution of volumes and surfaces. This is achieved using two key ingredients: (i) *reference configurations* that are considered fixed associated with (ii) *motions* that transform the reference configurations to their evolving counterparts. In what follows, we provide precise definitions of these concepts.

Reference configurations. The material-form parameterization involves fixed (i.e., independent of the parameter θ) **reference volume** $\mathcal{B}_\mathcal{V} \subset \mathbb{R}^3$ and **reference surface** $\mathcal{B}_\mathcal{M} \subset \mathcal{B}_\mathcal{V}$ that correspond, respectively, to the evolving volume $\mathcal{V}(\theta)$ and surface $\mathcal{M}(\theta)$. Additionally, we define $\mathcal{B}_{\mathcal{V}_0} := \mathcal{B}_\mathcal{V} \setminus \mathcal{B}_\mathcal{M}$—which is also fixed—to represent the interior of $\mathcal{B}_\mathcal{V}$.

To distinguish points in the reference configurations to those belonging to the evolving counterparts, we refer to $\boldsymbol{p} \in \mathcal{B}_\mathcal{V}$ as **material points** and $\boldsymbol{x} \in \mathcal{V}(\theta)$ as **spatial points**. Similarly, in the rest of this dissertation, we will use the terms "*material*" and "*spatial*" to refer to quantity defined with respect to the reference (e.g., $\mathcal{B}_\mathcal{M}$ and $\mathcal{B}_{\mathcal{V}_0}$) and the evolving (e.g., $\mathcal{M}$ and $\mathcal{V}_0$) domains, respectively.

Motion. Associated with the reference volume and surface is a **motion** $\mathtt{X}$ satisfying that:

- For any $\theta \in \mathbb{R}$, $\mathtt{X}(\cdot, \theta)$ maps the reference surface $\mathcal{B}_{\mathcal{M}}$ to its spatial counterpart $\mathcal{M}(\theta)$ and (the interior of) the reference volume $\mathcal{B}_{\mathcal{V}_0}$ to $\mathcal{V}_0(\theta)$ in a C^0-continuous and one-to-one fashion (see Figure 3.1).
- For any fixed material point $\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}}$, $\mathtt{X}(\boldsymbol{p}, \theta)$ is differentiable with respect to θ.

Additionally, we define the **reference map** $\mathtt{P}$ as the inverse of the motion $\mathtt{X}$: For any θ, $\mathtt{P}(\cdot, \theta) := \mathtt{X}^{-1}(\cdot, \theta)$ maps the evolving $\mathcal{M}(\theta)$ and $\mathcal{V}_0(\theta)$ back to their material counterparts $\mathcal{B}_{\mathcal{M}}$ and $\mathcal{B}_{\mathcal{V}_0}$, respectively.

We will discuss how the reference configurations and associated motion are specified under practical settings in §3.3.

3.1.2 Material-Form Integrals

The material-form parameterization described in §3.1.1 induces a change of variable from spatial points to material ones. In the following, we discuss how integrals can be rewritten using such changes of variables.

Material-form surface integral. Now we consider a surface integral defined on the evolving surface $\mathcal{M}(\theta)$:

$$I = \int_{\mathcal{M}(\theta)} \varphi(\boldsymbol{x}, \theta) \, \mathrm{d}A(\boldsymbol{x}), \tag{3.1}$$

where $\boldsymbol{x}$ is a spatial point on $\mathcal{M}(\theta)$ and φ is a scalar-valued function. Given the motion $\mathtt{X}(\cdot, \theta)$, we apply a change of variable from spatial points $\boldsymbol{x} \in \mathcal{M}(\theta)$ to material ones $\boldsymbol{p} \in \mathcal{B}_{\mathcal{M}}$ based

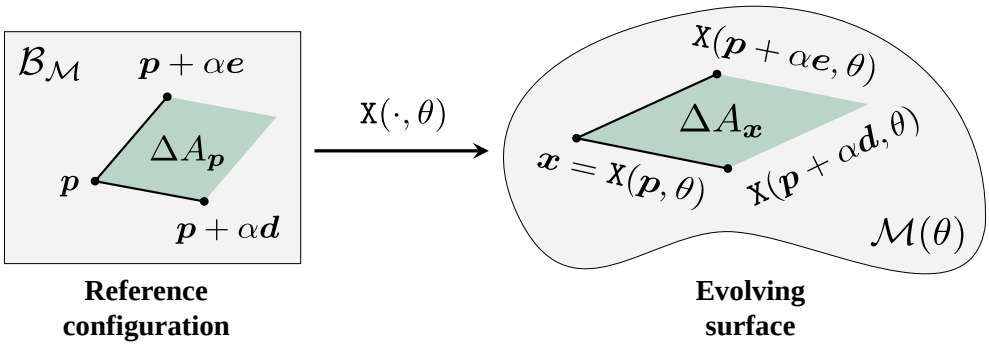


Figure 3.2: Given a deformation $\mathtt{X}(\cdot,\theta)$ with fixed $\theta\in\mathbb{R}$, the **Jacobian term** of Eq. (3.3) equals the ratio of the infinitesimal areas $|\Delta A_{\boldsymbol{x}}|$ and $|\Delta A_{\boldsymbol{p}}|$.

on the relation $\boldsymbol{x}=\mathtt{X}(\boldsymbol{p},\theta)$, yielding the following **material-form surface integral**:

$$I=\int_{\mathcal{B}_\mathcal{M}}\varphi(\boldsymbol{x},\theta)\,J(\boldsymbol{p})\,\mathrm{d}A(\boldsymbol{p}),\tag{3.2}$$

where $\boldsymbol{x}=\mathtt{X}(\boldsymbol{p},\theta)$, and J is the Jacobian determinant resulting from the change of variable.

Conceptually, the term J in Eq. (3.2) captures the stretching of infinitesimal surface area. Precisely, let $\Delta A_{\boldsymbol{p}}$ be an infinitesimal (material) parallelogram with sides $\alpha\boldsymbol{d}$ and $\alpha\boldsymbol{e}$ at some $\boldsymbol{p}\in\mathcal{B}_\mathcal{M}$. Under the mapping $\mathtt{X}(\cdot,\theta)$, $\Delta A_{\boldsymbol{p}}$ is transformed into another (spatial) parallelogram $\Delta A_{\boldsymbol{x}}$ with sides $\boldsymbol{d}_\alpha=\mathtt{X}(\boldsymbol{p}+\alpha\boldsymbol{d},\theta)-\boldsymbol{x}$ and $\boldsymbol{e}_\alpha=\mathtt{X}(\boldsymbol{p}+\alpha\boldsymbol{e},\theta)-\boldsymbol{x}$. Then, the Jacobian determinant J equals the ratio of the infinitesimal area $|\Delta A_{\boldsymbol{x}}|$ to $|\Delta A_{\boldsymbol{p}}|$:

$$J(\boldsymbol{p})=\left\|\frac{\mathrm{d}A(\boldsymbol{x})}{\mathrm{d}A(\boldsymbol{p})}\right\|=\lim_{\alpha\to0}\frac{|\Delta A_{\boldsymbol{x}}|}{|\Delta A_{\boldsymbol{p}}|}=\lim_{\alpha\to0}\frac{\|\boldsymbol{d}_\alpha\times\boldsymbol{e}_\alpha\|}{\|\alpha\boldsymbol{d}\times\alpha\boldsymbol{e}\|}\tag{3.3}$$

where "$\times$" denotes vector cross product (see Figure 3.2).

Material-form volume integral. Similarly, consider a volume integral of some scalar-valued function $\varphi(\boldsymbol{x},\theta)$ over some evolving domain $\mathcal{V}_0(\theta)$

$$I=\int_{\mathcal{V}_0(\theta)}\varphi(\boldsymbol{x},\theta)\,\mathrm{d}V(\boldsymbol{x}).\tag{3.4}$$

Given a motion $\mathtt{X}(\cdot, \theta)$ that maps material points $\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}_0}$ to spatial points $\boldsymbol{x} \in \mathcal{V}_0$, we can rewrite Eq. (3.4) as a **material-form volume integral** of the form:

$$I = \int_{\mathcal{B}_{\mathcal{V}_0}} \varphi(\boldsymbol{x}, \theta)\, J(\boldsymbol{p})\, \mathrm{d}V(\boldsymbol{p}). \tag{3.5}$$

where $\boldsymbol{x} = \mathtt{X}(\boldsymbol{p}, \theta)$. As we are dealing with a volume integral, the Jacobian determinant J now captures the ratio of the infinitesimal volume $|\Delta V_{\boldsymbol{x}}|$ to $|\Delta V_{\boldsymbol{p}}|$:

$$J(\boldsymbol{p}) = \left\| \frac{\mathrm{d}V(\boldsymbol{x})}{\mathrm{d}V(\boldsymbol{p})} \right\| = \lim_{\alpha \to 0} \frac{|\Delta V_{\boldsymbol{x}}|}{|\Delta V_{\boldsymbol{p}}|} = \lim_{\alpha \to 0} \frac{|\boldsymbol{a}_\alpha \times \boldsymbol{b}_\alpha \cdot \boldsymbol{c}_\alpha|}{|\alpha \boldsymbol{a} \times \alpha \boldsymbol{b} \cdot \alpha \boldsymbol{c}|}, \tag{3.6}$$

where $\Delta V_{\boldsymbol{p}}$ is a (material) tetrahedron with sides $\alpha \boldsymbol{a}$, $\alpha \boldsymbol{b}$ and $\alpha \boldsymbol{c}$; $\Delta V_{\boldsymbol{x}}$ is a (spatial) tetrahedron with sides $\boldsymbol{a}_\alpha = \mathtt{X}(\boldsymbol{p} + \alpha \boldsymbol{a}, \theta) - \boldsymbol{x}$, $\boldsymbol{b}_\alpha = \mathtt{X}(\boldsymbol{p} + \alpha \boldsymbol{b}, \theta) - \boldsymbol{x}$ and $\boldsymbol{c}_\alpha = \mathtt{X}(\boldsymbol{p} + \alpha \boldsymbol{c}, \theta) - \boldsymbol{x}$.

We will discuss the calculation of J given by Eqs. (3.3) and (3.6) under practical settings in §3.3.

Advantages. The material-form integrals in Eqs. (3.2) and (3.5) are defined over fixed reference surfaces or volumes. Compared with their spatial counterparts in Eqs. (3.1) and (3.4), the material-form integrals can be differentiated more easily using Reynolds transport theorem. For instance, all (topological) boundaries and jump discontinuities that are "static" (i.e., fixed) in the reference domains do not need to be handled by boundary integrals. We will further discuss the advantages of material-form parameterization in §4.1.2 when differentiating full path integrals.

Multiple parameters. So far, our discussion is limited to the differentiation with respect to a single parameter. In practice, the scene geometry $\mathcal{V}$ is usually controlled by multiple, say $m_{\boldsymbol{\theta}}$, parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ (e.g., positions of individual mesh vertices). In this case, the motion $\mathtt{X}$ also depends on the full parameter vector $\boldsymbol{\theta}$. That is, for any $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$, $\mathtt{X}(\cdot, \boldsymbol{\theta})$

establishes one-to-one mappings from the reference surface $\mathcal{B}_{\mathcal{M}}$ to $\mathcal{M}(\boldsymbol{\theta})$ and volume $\mathcal{B}_{\mathcal{V}_0}$ to $\mathcal{V}_0(\boldsymbol{\theta})$. Since this extension is straightforward, we will mainly focus on the single-parameter case in the rest of this dissertation for exposition simplicity.

3.2 Material-Form Path Integral

Previously, we introduced the formulation of generalized path integral (§2.2.2), whose integration domain may depend on geometric parameters (e.g., the shape and pose of an object). To facilitate the differentiation of path integrals, we now devise *material-form path integrals* based on the parameterization presented in §3.1.

In the rest of this chapter, unless otherwise specified, we use "path space" and "path integral" to indicate the generalized variants given by Eqs. (2.24) and Eq. (2.20), respectively.

Under the path-integral formulation, a light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ is *spatial* since all path vertices are spatial points belonging to the evolving domain $\mathcal{V}(\theta) = \mathcal{M}(\theta) \bigcup \mathcal{V}_0(\theta)$. Based on the material-form parameterization, we define a **material light path** as $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ where each vertex $\boldsymbol{p}_n \in \mathcal{B}_{\mathcal{V}} = \mathcal{B}_{\mathcal{M}} \bigcup \mathcal{B}_{\mathcal{V}_0}$ is a material point (for $n = 0, \ldots, N$). Following Eq. (2.21), we denote the set comprised of all material light paths $\bar{\boldsymbol{p}}$ with N segments and characteristic l as

$$\hat{\boldsymbol{\Omega}}_N^l := \left\{ (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N) \ : \ \begin{array}{ll} \boldsymbol{p}_n \in \mathcal{B}_{\mathcal{M}} & \text{if } b_n(l) = 1 \\ \boldsymbol{p}_n \in \mathcal{B}_{\mathcal{V}_0} & \text{if } b_n(l) = 0 \end{array} \right\}, \tag{3.7}$$

associated with the measure μ_N^l defined in Eq. (2.22), where the surface area and volume are with respect to the reference surface $\mathcal{B}_{\mathcal{M}}$ and volume $\mathcal{B}_{\mathcal{V}_0}$, respectively. Further, similar

to the ordinary path space $\boldsymbol{\Omega}$ (2.24), we define the **material path space** $\hat{\boldsymbol{\Omega}}$ as

$$\hat{\boldsymbol{\Omega}} := \bigcup_{N\geq 1} \bigcup_{l=0}^{2^{N+1}-1} \hat{\boldsymbol{\Omega}}_N^l, \tag{3.8}$$

with an associated measure μ defined in Eq. (2.25).

Additionally, for any $\theta \in \mathbb{R}$, the vertex-level mapping $\mathtt{X}(\cdot, \theta)$ induces a **path-level one** $\bar{\mathtt{X}}(\cdot, \theta)$ that maps material paths $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ to spatial ones $\bar{\boldsymbol{x}} = \bar{\mathtt{X}}(\bar{\boldsymbol{p}}, \theta) := (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ in a continuous and one-to-one fashion via $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \theta)$ for all n. By applying this path-level mapping to the ordinary path integral of Eq. (2.20), we obtain the **material-form path integral**:

$$\boxed{I = \int_{\hat{\boldsymbol{\Omega}}} \hat{f}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}}),} \tag{3.9}$$

where $\hat{f}$ is the **material measurement contribution function** defined as

$$\hat{f}(\bar{\boldsymbol{p}}) := f(\bar{\boldsymbol{x}}) \left\| \frac{\mathrm{d}\mu(\bar{\boldsymbol{x}})}{\mathrm{d}\mu(\bar{\boldsymbol{p}})} \right\| = f(\bar{\boldsymbol{x}}) \prod_n J(\boldsymbol{p}_n), \tag{3.10}$$

where the definition of $J(\boldsymbol{p})$ depends on the type of $\boldsymbol{p}$:

$$J(\boldsymbol{p}) := \begin{cases} \left\| \frac{\mathrm{d}A(\boldsymbol{x})}{\mathrm{d}A(\boldsymbol{p})} \right\|, & (\boldsymbol{p} \in \mathcal{B}_{\mathcal{M}}) \\ \left\| \frac{\mathrm{d}V(\boldsymbol{x})}{\mathrm{d}V(\boldsymbol{p})} \right\|. & (\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}_0}) \end{cases} \tag{3.11}$$

We note that, since the mapping $\boldsymbol{x} = \mathtt{X}(\boldsymbol{p}, \theta)$ depends on the parameter θ, so would $J(\boldsymbol{p})$. Eqs. (3.9—3.11) are key results of this dissertation, and we will discuss how Eq. (3.9) can be differentiated in §4.1.

3.3 Reference Configurations and Motion

Theoretically, our material-form formulation presented above allows the use of arbitrary reference surfaces and volumes as well as associated motions. In practice, a good parameterization allows the corresponding terms, such as the Jacobian determinants and "velocities" (that we will discuss later), to be calculated conveniently, benefiting the design of differentiable rendering algorithms. In this section, we introduce our choice of references and motion in details. We focus on the practical setting where the scene volume is expressed in a tetrahedral mesh with the surfaces given by triangle meshes comprised of (some subset of) tetrahedron faces.

In practice, to compute derivatives of a path integral at some user-specified $\theta = \theta_0$, we make the reference scene geometry $\mathcal{B}_\mathcal{V}$ coincide with the spatial one $\mathcal{V}(\theta)$ with the parameter θ fixed at θ_0. That is, we choose the reference surface as $\mathcal{B}_\mathcal{M} = \mathcal{M}(\theta_0)$ and volume as $\mathcal{B}_{\mathcal{V}_0} = \mathcal{V}_0(\theta_0)$. Further, as presented in §3.1.1, for any θ, the motion $\mathtt{X}$ produces a mapping $\mathtt{X}(\cdot, \theta)$ that transforms material points $\boldsymbol{p} \in \mathcal{B}_\mathcal{V}$ to spatial ones $\boldsymbol{x} \in \mathcal{V}$ in a continuous and one-to-one fashion. Based on our choice of reference configurations, the mapping $\mathtt{X}(\cdot, \theta_0)$ becomes the identity map, and $J(\theta_0) = 1$.

For calculating derivatives at $\theta = \theta_0$, we do not need to provide the full motion $\mathtt{X}$ explicitly. Instead, it suffices to specify, for any material point $\boldsymbol{p} \in \mathcal{B}_\mathcal{V}$, both its spatial counterpart $\boldsymbol{x}(\boldsymbol{p}) := \mathtt{X}(\boldsymbol{p}, \theta_0)$ and the derivative, or "**velocity**",

$$\boldsymbol{v}(\boldsymbol{p}) := \left.\frac{\mathrm{d}\mathtt{X}(\boldsymbol{p}, \theta)}{\mathrm{d}\theta}\right|_{\theta=\theta_0}. \tag{3.12}$$

Since $\mathtt{X}(\cdot, \theta_0)$ is the identity map, we have $\boldsymbol{x}(\boldsymbol{p}) = \boldsymbol{p}$. The velocity $\boldsymbol{v}$, on the other hand, is typically determined by how the parameter θ affects the scene geometry. In what follows, we discuss two common scenarios and derive the corresponding derivatives $\left.\frac{\mathrm{d}J}{\mathrm{d}\theta}\right|_{\theta=\theta_0}$ of the

Jacobian determinant J given by Eq. (3.11). We note that, although $J(\boldsymbol{p}, \theta_0) = 1$, this derivative is generally nonzero.

3.3.1 Affine Deformations

When the parameter θ specifies some affine transformation of an object, the corresponding motion can be expressed as

$$\mathtt{X}(\boldsymbol{p}, \theta) = \boldsymbol{R}(\theta)\,\boldsymbol{p} + \boldsymbol{t}(\theta), \tag{3.13}$$

where $\boldsymbol{R}(\theta)$ is an invertible (3×3)-matrix and $\boldsymbol{t}(\theta)$ is a vector satisfying that $\boldsymbol{R}(\theta_0)$ equals the identity matrix and $\boldsymbol{t}(\theta_0) = \boldsymbol{0}$. Assuming that $\dot{\boldsymbol{R}} := \frac{\mathrm{d}\boldsymbol{R}}{\mathrm{d}\theta}\big|_{\theta=\theta_0}$ and $\dot{\boldsymbol{t}} := \frac{\mathrm{d}\boldsymbol{t}}{\mathrm{d}\theta}\big|_{\theta=\theta_0}$ can be calculated analytically, we have

$$\boldsymbol{v}(\boldsymbol{p}) := \left.\frac{\mathrm{d}\mathtt{X}(\boldsymbol{p}, \theta)}{\mathrm{d}\theta}\right|_{\theta=\theta_0} = \dot{\boldsymbol{R}}\,\boldsymbol{p} + \dot{\boldsymbol{t}}. \tag{3.14}$$

Given Eq. (3.13), it is easy to verify that

$$J(\boldsymbol{p}) = \left|\det(\boldsymbol{R})\right|, \tag{3.15}$$

for all $\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}}$. It follows that the derivative $\left.\frac{\mathrm{d}J(\boldsymbol{p})}{\mathrm{d}\theta}\right|_{\theta=\theta_0}$ can be obtained via differentiable evaluation of Eq. (3.15).

3.3.2 Free-Form Deformations

When the scene geometry is expressed using tetrahedral meshes (or triangle meshes without participating media), general (non-affine) deformations are typically handled by moving individual mesh vertices. In this case, we first specify the velocity $\boldsymbol{v}_i$ at the i-th mesh vertex and *interpolate* the specified velocity continuously in the interior of the mesh. In

what follows, we first explain the interpolation process and then discuss how the per-vertex velocities $\boldsymbol{v}_i$ is formulated in practice.

Barycentric Interpolation

Provided the per-vertex velocities $\boldsymbol{v}_i$, we treat the mapping $\mathtt{X}(\cdot, \theta_0)$ at any vertex $\boldsymbol{p}_i$ as

$$\mathtt{X}(\boldsymbol{p}_i, \theta) = \boldsymbol{p}_i + (\theta - \theta_0)\, \boldsymbol{v}_i, \tag{3.16}$$

which satisfies that $\mathtt{X}(\boldsymbol{p}_i, \theta_0) = \boldsymbol{p}_i$ and $\frac{\mathrm{d}\mathtt{X}(\boldsymbol{p}_i, \theta)}{\mathrm{d}\theta}\Big|_{\theta=\theta_0} = \boldsymbol{v}_i$. We now specify the velocity $\boldsymbol{v}(\boldsymbol{p})$ at any material point $\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}}$ using barycentric interpolation.

Surface points. Let $\boldsymbol{p} \in \mathcal{B}_{\mathcal{M}}$ be a material point located on some triangle mesh and inside a triangle with vertices $\boldsymbol{p}_{\mathsf{A}}$, $\boldsymbol{p}_{\mathsf{B}}$, and $\boldsymbol{p}_{\mathsf{C}}$. Assume $\boldsymbol{p}$ to have barycentric coordinates (s, t): That is, $\boldsymbol{p} = (1 - s - t)\, \boldsymbol{p}_{\mathsf{A}} + s\, \boldsymbol{p}_{\mathsf{B}} + t\, \boldsymbol{p}_{\mathsf{C}}$. Then, we define the mapping $\mathtt{X}(\boldsymbol{p}, \theta)$ by treating s and t as constants:

$$\mathtt{X}(\boldsymbol{p}, \theta) = (1 - s - t)\, \boldsymbol{x}_{\mathsf{A}} + s\, \boldsymbol{x}_{\mathsf{B}} + t\, \boldsymbol{x}_{\mathsf{C}}, \tag{3.17}$$

where $\boldsymbol{x}_j = \mathtt{X}(\boldsymbol{p}_j, \theta) = \boldsymbol{p}_j + (\theta - \theta_0)\, \boldsymbol{v}_j$ for $j \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}\}$.

We now derive the Jacobian term J based on Eq. (3.3). Let $\boldsymbol{d} = \boldsymbol{p}_{\mathsf{B}} - \boldsymbol{p}_{\mathsf{A}}$ and $\boldsymbol{e} = \boldsymbol{p}_{\mathsf{C}} - \boldsymbol{p}_{\mathsf{A}}$, we have

$$\begin{aligned} \boldsymbol{d}_\alpha &:= \mathtt{X}(\boldsymbol{p} + \alpha \boldsymbol{d}, \theta) - \mathtt{X}(\boldsymbol{p}, \theta) \\ &= [(1 - s - t - \alpha)\, \boldsymbol{x}_{\mathsf{A}} + (s + \alpha)\, \boldsymbol{x}_{\mathsf{B}} + t\, \boldsymbol{x}_{\mathsf{C}}] - \\ &\quad [(1 - s - t)\, \boldsymbol{x}_{\mathsf{A}} + s\, \boldsymbol{x}_{\mathsf{B}} + t\, \boldsymbol{x}_{\mathsf{C}}] \\ &= \alpha\, (\boldsymbol{x}_{\mathsf{B}} - \boldsymbol{x}_{\mathsf{A}}), \end{aligned} \tag{3.18}$$

and, similarly,

$$\boldsymbol{e}_\alpha = \mathtt{X}(\boldsymbol{p} + \alpha \boldsymbol{e}, \theta) - \mathtt{X}(\boldsymbol{p}, \theta) = \alpha\,(\boldsymbol{x}_\mathsf{C} - \boldsymbol{x}_\mathsf{A}). \tag{3.19}$$

Then, the Jacobian determinant in Eq. (3.3) turns into

$$J(\boldsymbol{p}) = \lim_{\alpha \to 0} \frac{\|\boldsymbol{d}_\alpha \times \boldsymbol{e}_\alpha\|}{\|\alpha \boldsymbol{d} \times \alpha \boldsymbol{e}\|} = \frac{\|(\boldsymbol{x}_\mathsf{B} - \boldsymbol{x}_\mathsf{A}) \times (\boldsymbol{x}_\mathsf{C} - \boldsymbol{x}_\mathsf{A})\|}{\|(\boldsymbol{p}_\mathsf{B} - \boldsymbol{p}_\mathsf{A}) \times (\boldsymbol{p}_\mathsf{C} - \boldsymbol{p}_\mathsf{A})\|}, \tag{3.20}$$

which is independent of the barycentric coordinate (s, t). In other words, under the deformation of Eq. (3.17), J remains constant within the triangle. At $\theta = \theta_0$, it is easy to verify that J equals one, and the derivative $\frac{\mathrm{d}J(\boldsymbol{p})}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}$ can be obtained via differentiable evaluation of Eq. (3.20).

Volume points. Let $\boldsymbol{p} \in \mathcal{B}_{\mathcal{V}_0}$ be a volume vertex inside a tetrahedron with vertices $\boldsymbol{p}_j$ for $j \in \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$. When $\boldsymbol{p}$ has barycentric coordinates (u, v, w), it holds that $\boldsymbol{p} = (1 - u - v - w)\,\boldsymbol{p}_\mathsf{A} + u\,\boldsymbol{p}_\mathsf{B} + v\,\boldsymbol{p}_\mathsf{C} + w\,\boldsymbol{p}_\mathsf{D}$. Similar to the surface case, we formulate the mapping $\mathtt{X}(\boldsymbol{p}, \theta)$ as

$$\mathtt{X}(\boldsymbol{p}, \theta) = (1 - u - v - w)\,\boldsymbol{x}_\mathsf{A} + u\,\boldsymbol{x}_\mathsf{B} + v\,\boldsymbol{x}_\mathsf{C} + w\,\boldsymbol{x}_\mathsf{D}, \tag{3.21}$$

where $\boldsymbol{x}_j = \mathtt{X}(\boldsymbol{p}_j, \theta) = \boldsymbol{p}_j + (\theta - \theta_0)\,\boldsymbol{v}_j$ for all j. Similar to the derivation of Eq. (3.3), the Jacobian determinant defined in Eq. (3.6) becomes

$$J(\boldsymbol{p}) = \frac{\|(\boldsymbol{x}_\mathsf{B} - \boldsymbol{x}_\mathsf{A}) \times (\boldsymbol{x}_\mathsf{C} - \boldsymbol{x}_\mathsf{A}) \cdot (\boldsymbol{x}_\mathsf{D} - \boldsymbol{x}_\mathsf{A})\|}{\|(\boldsymbol{p}_\mathsf{B} - \boldsymbol{p}_\mathsf{A}) \times (\boldsymbol{p}_\mathsf{C} - \boldsymbol{p}_\mathsf{A}) \cdot (\boldsymbol{p}_\mathsf{D} - \boldsymbol{p}_\mathsf{A})\|}, \tag{3.22}$$

which is constant within the tetrahedron. Similar to the surface case, when $\theta = \theta_0$, $J(\boldsymbol{p})$ equals one for all $\boldsymbol{p}$ inside the tetrahedron. The derivative $\frac{\mathrm{d}J(\boldsymbol{p})}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}$, which is generally nonzero, can be obtained via differentiable evaluation of Eq. (3.22).

Per-Vertex Velocities

In practice, the per-vertex velocity $\boldsymbol{v}_i$ in Eq. (3.16) is implied based on how θ controls each mesh vertex. For example, if θ specifies a (global) translation along the x-axis, $\boldsymbol{v}_i = (1\ 0\ 0)^\top$ for all i.

On the other hand, if the tetrahedral or triangle mesh contains N vertices, and the differentiation is with respect to the position of each vertex, we effectively have a parameter vector $\boldsymbol{\theta} = (\theta^{1x}, \theta^{1y}, \theta^{1z}, \cdots, \theta^{Nx}, \theta^{Ny}, \theta^{Nz}) \in \mathbb{R}^{3N}$ with $\theta^{ix}, \theta^{iy}, \theta^{iz}$ indicating the x-, y-, and z-coordinates of the i-th vertex $\boldsymbol{x}_i$, respectively. Then, we formulate the mapping $\mathtt{X}(\cdot, \theta)$ at vertex $\boldsymbol{p}_i$ as

$$\mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta}) = \begin{bmatrix} \theta^{ix} \\ \theta^{iy} \\ \theta^{iz} \end{bmatrix}. \tag{3.23}$$

We note that, when computing derivatives at some $\boldsymbol{\theta} = \boldsymbol{\theta}_0$ with the reference configuration being the mesh given by $\boldsymbol{\theta}_0$, Eq. (3.23) satisfies $\mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta}_0) = \boldsymbol{p}_i$ trivially.

Under this multi-parameter setting, the "velocity" at $\boldsymbol{p}_i$ is essentially the partial derivatives of $\mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})$ with respect to the individual components of $\boldsymbol{\theta}$. Specifically, according to Eq. (3.23), it holds that

$$\frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{ix}} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \qquad \frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{iy}} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{iz}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{3.24}$$

and $\frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{jx}} = \frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{jy}} = \frac{\partial \mathtt{X}(\boldsymbol{p}_i, \boldsymbol{\theta})}{\partial \theta^{jz}} = \mathbf{0}$ for all $j \neq i$.

In practice, when computing derivatives at $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, the simple deformation in Eq. (3.23) can be implemented by representing the spatial representations of mesh vertices $\boldsymbol{x}_i$ as automatic-differentiation-enabled vectors. Further, the material counterpart $\boldsymbol{p}_i$ of $\boldsymbol{x}_i$ can be obtained

by simply "detaching" the latter: That is, $\boldsymbol{p}_i = \mathsf{detach}(\boldsymbol{x}_i)$. In this way, $\boldsymbol{p}_i$ takes the same value of $\boldsymbol{x}_i$ but is considered constant (i.e., independent of the parameters $\boldsymbol{\theta}$). The rest of computation can be accomplished using automatic differentiation, including the barycentric interpolations in Eqs. (3.17, 3.21) and Jacobian evaluations in Eqs. (3.20, 3.22).

Chapter 4

Differential Path Integrals

In this chapter, we devise a key result of this dissertation—*differential path integrals*—by applying Reynolds transport theorem (§2.3.1) to material-form path integrals (§3.2). This mathematical formulation will serve as the theoretical foundation for developing several unbiased and consistent Monte Carlo estimators in later chapters.

Specifically, our objective is to derive derivatives of material-form path integrals defined in Eq. (3.9) with respect to arbitrary scene parameters that can control, for example, the pose of an object, the position of a vertex, or the albedo of a surface.

Continuity assumptions. To facilitate the derivation of the derivative, we make the following assumptions:

A.1 There is no ideal specular surface (e.g., perfect mirror or smooth glass);

A.2 The source emission L_e, sensor importance W_e, BSDFs f_s, and phase functions f_p are C^0-continuous spatially and directionally;

A.3 The extinction coefficient σ_t and scattering coefficient σ_s are continuous in the interior

of all participating media;

A.4 Discontinuities of the Jacobian determinants J of Eq. (3.11), if they exist, are independent of the parameter with respect to which the differentiation is taken.

In practice, when computing derivatives at some $\theta = \theta_0$, Assumption **A.4** is satisfied under our choice of reference and motion discussed in §3.3. This is because J always equals one at $\theta = \theta_0$ (due to $\mathtt{X}(\cdot, \theta_0)$ reducing to the identity map).

Based on Assumptions **A.1**–**A.4**, we differentiate full material-form path integrals in §4.1. In addition, we slightly relax Assumption **A.2** by showing how zero-measure sources and sensors such as point lights and pinhole camera can be handled in §4.2.

4.1 Differentiating Material-Form Path Integrals

Now we derive derivatives of the material-form path integral in Eq. (3.9) with respect to some arbitrary scene parameter $\theta \in \mathbb{R}$.

Given the definition of the measure μ in Eq. (2.25), we start with rewriting Eq. (3.9) as

$$I = \sum_{N \geq 1} \sum_{l=0}^{2^{N+1}-1} \underbrace{\int_{\hat{\mathbf{\Omega}}^l_N} \hat{f}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu^l_N(\bar{\boldsymbol{p}})}_{=:\, I^l_N} , \tag{4.1}$$

where $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ is a material light path, and $\hat{\mathbf{\Omega}}^l_N$, μ^l_N are defined in Eqs. (3.7) and (2.22), respectively. Then, deriving the derivative $\frac{\mathrm{d}I}{\mathrm{d}\theta}$ amounts to differentiating I^l_N for any fixed $N > 0$ and path characteristic $0 \leq l < 2^{N+1}$.

To this end, we first express I_N^l in a recursive fashion by defining

$$h_N(\boldsymbol{p}_N; \boldsymbol{p}_{N-1}) := J(\boldsymbol{p}_N)\, W_{\mathrm{e}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N), \tag{4.2}$$

and, for $0 < K \leq N$,

$$h_{K-1}(\boldsymbol{p}_{K-1}; \boldsymbol{p}_{K-2}) := \int_{\mathcal{B}_K} h_K(\boldsymbol{p}_K; \boldsymbol{p}_{K-1})\, g(\boldsymbol{x}_K; \boldsymbol{x}_{K-2}, \boldsymbol{x}_{K-1})\, \mathrm{d}\mu_{N,K}^l(\boldsymbol{p}_K), \tag{4.3}$$

where $\mathrm{d}\mu_{N,n}^l$ is defined in Eq. (2.23) and

$$g(\boldsymbol{x}_K; \boldsymbol{x}_{K-2}, \boldsymbol{x}_{K-1}) := J(\boldsymbol{p}_{K-1})\, f_{\mathrm{v}}(\boldsymbol{x}_{K-2} \to \boldsymbol{x}_{K-1} \to \boldsymbol{x}_K)\, G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K). \tag{4.4}$$

In Eqs. (4.2) and (4.4), the function f_{v}, geometric term G and Jacobian determinant J are defined in Eqs. (2.27), (2.28) and (3.11), respectively. In Eq. (4.3), the domain of integration $\mathcal{B}_K$ is determined by $b_K(l)$—the K-th bit of the given path characteristic l:

$$\mathcal{B}_K = \begin{cases} \mathcal{B}_{\mathcal{M}}, & (b_K(l) = 1) \\ \mathcal{B}_{\mathcal{V}_0}. & (b_K(l) = 0) \end{cases} \tag{4.5}$$

Given Eqs. (4.2–4.5), it is easy to verify that

$$I_N^l = \int_{\mathcal{B}_0} h_0(\boldsymbol{p}_0)\, \mathrm{d}\mu_{N,0}^l(\boldsymbol{p}_0). \tag{4.6}$$

Then, deriving $\frac{\mathrm{d}I_N^l}{\mathrm{d}\theta}$ amounts to differentiating Eq. (4.3). Under Assumptions **A.1**–**A.4**, discontinuities of the integrand of Eq. (4.3) emerge solely from the mutual visibility function $\mathbb{V}$ that is a factor of the g function defined in Eq. (4.4). Thus, applying Reynolds transport theorem yields:

$$\frac{\mathrm{d}h_{K-1}}{\mathrm{d}\theta} = \int_{\mathcal{B}_K} \frac{\mathrm{d}}{\mathrm{d}\theta}(h_K\, g)\, \mathrm{d}\mu_{N,K}^l + \int_{\partial\mathcal{B}_K} (h_K\, g)\, \frac{\Delta G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)}{G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)}\, v_\perp(\boldsymbol{p}_K)\, \mathrm{d}\dot{\mu}_{N,K}^l, \tag{4.7}$$

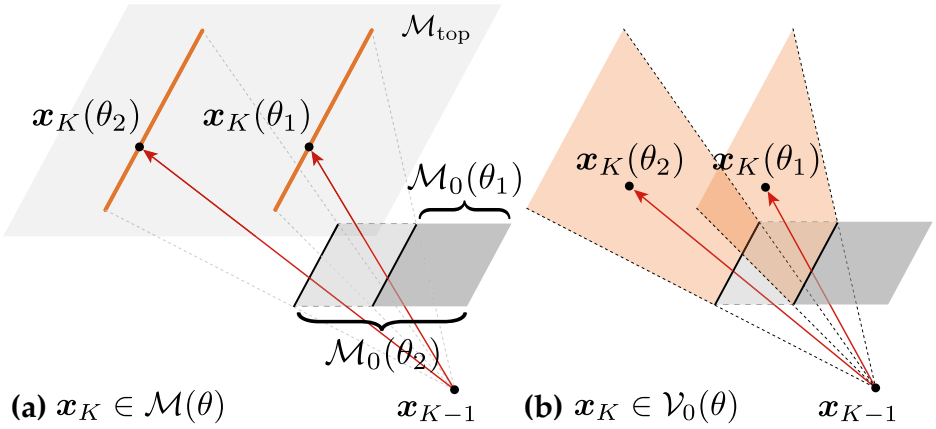


Figure 4.1: **Evolution of discontinuity boundaries:** when the scene geometry varies with the parameter θ, so will the visibility boundaries. In this example, θ controls the width of the surface $\mathcal{M}_0$. Given a fixed point $\boldsymbol{x}_{K-1}$ (on a surface or inside a medium), the discontinuity points $\boldsymbol{x}_K$ can lay (a) on a curve within an object surface ($\mathcal{M}_{\text{top}}$ in this example); and (b) within a surface determined by $\boldsymbol{x}_{K-1}$ and the left edge (in solid black lines) of $\mathcal{M}_0$. The discontinuity curves and surfaces (illustrated in orange) generally depend on the parameter θ. After transforming these curves and surfaces back to the reference configuration using the reference map $\mathtt{P}(\cdot, \theta)$, $v_\perp(\boldsymbol{p}_K)$ in Eq. (4.7) captures the change rate (with respect to θ) of $\boldsymbol{p}_K$ along the normal direction of the discontinuity curve or surface (under reference configurations).

where:

- $\partial\mathcal{B}_K \subset \mathcal{B}_K$ consists of jump discontinuity points of $g(\boldsymbol{x}_K; \boldsymbol{x}_{K-2}, \boldsymbol{x}_{K-1})$ with respect to $\boldsymbol{p}_K = \mathtt{P}(\boldsymbol{x}_K, \theta)$ (viz. the spatial counterpart of $\boldsymbol{x}_K$) when $\boldsymbol{x}_{K-2}$ and $\boldsymbol{x}_{K-1}$ are fixed. Specifically, when $b_K(l) = 1$, the integral in Eq. (4.3) is over all object surfaces $\mathcal{B}_\mathcal{M}$, and $\partial\mathcal{B}_n$ is a set of curves (see Figure 4.1-a). On the other hand, when $b_K(l) = 0$, the right-hand side of Eq. (4.3) becomes a volume integral, and $\partial\mathcal{B}_K$ takes the form of a collection of surfaces (see Figure 4.1-b).

- $v_\perp(\boldsymbol{p}_K)$ is the scalar normal velocity of the discontinuity boundary $\partial\mathcal{B}_K$ at $\boldsymbol{p}_K$ (see Figure 4.1) and can be calculated via

$$v_\perp(\boldsymbol{p}_K) := \frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\theta} \cdot \boldsymbol{n}(\boldsymbol{p}_K), \tag{4.8}$$

where $\boldsymbol{n}$ is the unit-normal field associated with the discontinuity boundary $\partial\mathcal{B}_K$ (as a set of curves when $\boldsymbol{p}_K$ is a surface vertex or surfaces when $\boldsymbol{p}_K$ is a volume vertex). Further, evaluating the change rate $\frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\theta}$ in Eq. (4.8) requires locally parameterizing the discontinuity boundary $\partial\mathcal{B}_K$ in a neighborhood of $\boldsymbol{p}_K$. We will discuss in practice how this can be done in §5.2.5. We note that the scalar normal velocity $v_\perp$ is known to be *parameterization independent*. In other words, all (valid) local parameterizations of $\partial\mathcal{B}_K$ will lead to the same $\boldsymbol{p}_K$.

- ΔG indicates the difference in $G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)$ across the discontinuity boundaries.
- The measure $\dot{\mu}^l_{N,K}$ is defined as

$$\mathrm{d}\dot{\mu}^l_{N,K} := \begin{cases} \mathrm{d}\ell, & (b_K(l) = 1) \\ \mathrm{d}A, & (b_K(l) = 0) \end{cases} \tag{4.9}$$

where ℓ and A are, respectively, curve-length and surface-area measures.

With Eqs. (4.3) and (4.7), we can now differentiate I^l_N defined in Eq. (4.6) by repeatedly expanding h_{K-1} and $\frac{\mathrm{d}h_{K-1}}{\mathrm{d}\theta}$ for $K = 1, ..., N$, resulting in:

$$\frac{\mathrm{d}I^l_N}{\mathrm{d}\theta} = \overbrace{\int_{\hat{\boldsymbol{\Omega}}^l_N} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta}\,\mathrm{d}\mu^l_N(\bar{\boldsymbol{p}})}^{\text{interior}} + \overbrace{\sum_{K=1}^{N}\left[\int_{\partial\hat{\boldsymbol{\Omega}}^l_{N,K}} \Delta\hat{f}_K(\bar{\boldsymbol{p}})\,V_K(\boldsymbol{p}_K)\,\mathrm{d}\dot{\mu}^l_{N,K}(\bar{\boldsymbol{p}})\right]}^{\text{boundary}}, \tag{4.10}$$

where

$$\partial\hat{\boldsymbol{\Omega}}^l_{N,K} := \left(\prod_{n=0}^{K-1}\mathcal{B}_n\right) \times \partial\mathcal{B}_K \times \left(\prod_{n=K+1}^{N}\mathcal{B}_n\right), \tag{4.11}$$

$$\mathrm{d}\dot{\mu}^l_{N,K}(\bar{\boldsymbol{p}}) := \mathrm{d}\dot{\mu}^l_{N,K}(\boldsymbol{p}_K) \prod_{\substack{0\le n\le N \\ n\neq K}} \mathrm{d}\mu^l_{N,n}(\boldsymbol{p}_n), \tag{4.12}$$

$$\Delta\hat{f}_K(\bar{\boldsymbol{p}}) := \hat{f}(\bar{\boldsymbol{p}})\,\frac{\Delta G(\boldsymbol{x}_K \leftrightarrow \boldsymbol{x}_{K-1})}{G(\boldsymbol{x}_K \leftrightarrow \boldsymbol{x}_{K-1})}. \tag{4.13}$$

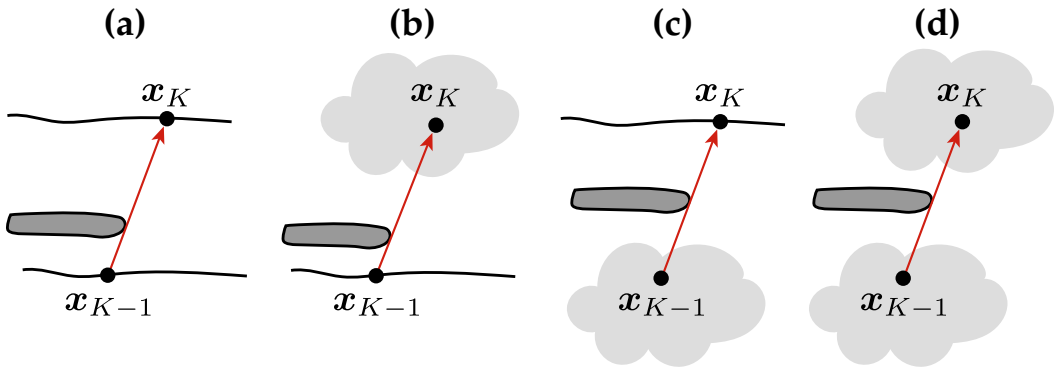


Figure 4.2: A **boundary segment** $\overline{\boldsymbol{x}_{K-1}\boldsymbol{x}_K}$ has property that its interior intersects with another surface in the scene at exactly one point. This causes the endpoint $\boldsymbol{x}_K$ to lie on the discontinuity boundary of the visibility function $\mathbb{V}(\boldsymbol{x}_{K-1} \leftrightarrow \cdot)$ with $\boldsymbol{x}_{K-1}$ fixed (and vise versa). With the presence of participating media, a boundary segment can connect two surface vertices (a), two volume vertices (d), or one surface and one volume vertices (b, c).

Lastly, we can sum up Eq. (4.10) for all $N \geq 1$ and $0 \leq l \leq 2^{N+1}-1$ to obtain our path-space differentiable rendering formulation.

4.1.1 Differential Path Integral

Based on the derivations above, we obtain a key result of this dissertation:

Differential path integral

The derivative of the material-form path integral (3.9) with respect to some arbitrary scene parameter $\theta \in \mathbb{R}$ is, in general, a *differential path integral* of the form:

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\int_{\hat{\Omega}} \hat{f}(\bar{\boldsymbol{p}})\,\mathrm{d}\mu(\bar{\boldsymbol{p}}) = \underbrace{\int_{\hat{\Omega}} \frac{\mathrm{d}}{\mathrm{d}\theta}\hat{f}(\bar{\boldsymbol{p}})\,\mathrm{d}\mu(\bar{\boldsymbol{p}})}_{\text{interior}} + \underbrace{\int_{\partial\hat{\Omega}} \Delta\hat{f}_K(\bar{\boldsymbol{p}})\,v_\perp(\boldsymbol{p}_K)\,\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}_{\text{boundary}}. \tag{4.14}$$

In the differential path integral (4.14), the *interior* term—which is essentially given by exchanging the differentiation and integration operations on the left-hand side—has the integrand $\frac{\mathrm{d}\hat{f}}{\mathrm{d}\theta}$ being the derivative of the material measurement contribution $\hat{f}$.

The *boundary* term of the differential path integral (4.14) is unique to our formulation. In this term, $\partial\hat{\boldsymbol{\Omega}}$ is the **boundary path space** defined as

$$\partial\hat{\boldsymbol{\Omega}} := \bigcup_{N\geq 1}\bigcup_{l=0}^{2^{N+1}-1}\bigcup_{K=1}^{N}\partial\hat{\boldsymbol{\Omega}}^{l}_{N,K}, \tag{4.15}$$

where $\partial\hat{\boldsymbol{\Omega}}^{l}_{N,K}$ is given by Eq. (4.11). The elements of $\partial\hat{\boldsymbol{\Omega}}$ are **material boundary paths** $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ that are identical to ordinary material paths except for containing exactly one **material boundary segment** $\overline{\boldsymbol{p}_{K-1}\,\boldsymbol{p}_K}$ (for some $0 < K \leq N$) such that the interior of its spatial counterpart $\overline{\boldsymbol{x}_{K-1}\,\boldsymbol{x}_K}$ given by $\boldsymbol{x}_{K-1} = \mathtt{X}(\boldsymbol{p}_{K-1}, \theta)$ and $\boldsymbol{x}_K = \mathtt{X}(\boldsymbol{p}_K, \theta)$ intersects the object surfaces $\mathcal{M}(\theta)$ at exactly one point (see Figure 4.2). We term $\overline{\boldsymbol{x}_{K-1}\,\boldsymbol{x}_K}$ a **boundary segment** and the spatial representation $\bar{\boldsymbol{x}} = \bar{\mathtt{X}}(\bar{\boldsymbol{p}}, \theta)$ a **boundary path**.

Associated with the boundary path space $\partial\hat{\boldsymbol{\Omega}}$ is the measure $\dot{\mu}$ satisfying that, given a material boundary path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ with characteristic l and boundary segment $\overline{\boldsymbol{p}_{K-1}\,\boldsymbol{p}_K}$, it holds that

$$\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}) = \left(\prod_{n\neq K}\mathrm{d}\mu^{l}_{N,n}(\boldsymbol{p}_n)\right)\begin{cases}\mathrm{d}\ell(\boldsymbol{p}_K), & (b_K(l)=1)\\ \mathrm{d}A(\boldsymbol{p}_K), & (b_K(l)=0)\end{cases} \tag{4.16}$$

where $\mathrm{d}\mu^{l}_{N,n}$ is defined in Eq. (2.23).

Additionally, for each boundary path $\bar{\boldsymbol{p}}$ with a material boundary segment $\overline{\boldsymbol{p}_{K-1}\,\boldsymbol{p}_K}$, $v_\perp(\boldsymbol{p}_K) \in \mathbb{R}$ follows the definition in Eq. (4.8) and captures how "fast" (with respect to θ) the discontinuity boundary evolves at $\boldsymbol{p}_K$ along the normal direction.

Lastly, the term $\Delta\hat{f}_K(\bar{\boldsymbol{p}})$ in Eq. (4.14) denotes the difference in material measurement contribution $\hat{f}$ across the discontinuity boundary specified by the boundary segment $\overline{\boldsymbol{p}_{K-1}\,\boldsymbol{p}_K}$. Based on Assumptions **A.1**–**A.4**, it holds that

$$\Delta\hat{f}_K(\bar{\boldsymbol{p}}) = \hat{f}(\bar{\boldsymbol{p}})\,\frac{\Delta G(\boldsymbol{x}_{K-1}\leftrightarrow\boldsymbol{x}_K)}{G(\boldsymbol{x}_{K-1}\leftrightarrow\boldsymbol{x}_K)}, \tag{4.17}$$

where $\Delta G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)$ equals $-G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)$ if the normal $\boldsymbol{n}(\boldsymbol{p}_K)$ of the discontinuity boundary at $\boldsymbol{p}_K$ points toward a region visible to $\boldsymbol{x}_{K-1}$, or $G(\boldsymbol{x}_{K-1} \leftrightarrow \boldsymbol{x}_K)$ if otherwise. In other words, $\Delta \hat{f}_K(\bar{\boldsymbol{p}}) = \pm \hat{f}(\bar{\boldsymbol{p}})$ with the sign based on the direction of the boundary normal $\boldsymbol{n}(\boldsymbol{p}_K)$.

4.1.2 Discussions

Relation with Reynolds transport theorem. Mathematically, our differential path integral presented in Eq. (4.14) is essentially a generalization of Reynolds transport theorem of Eq. (2.36) to material-form path integrals.

Non-geometric differentiation. In the special case when a scene parameter θ does not affect scene geometry (i.e., no visibility boundary depends on θ), the mapping $\mathtt{X}(\cdot, \theta)$ reduces to the identity map for all θ. This causes (i) the material path space $\hat{\boldsymbol{\Omega}}$ to become identical to the ordinary (spatial) one $\boldsymbol{\Omega}$; and (ii) the *boundary* integral in Eq. (4.14) to vanish.

Advantages of material-form path integrals. Although we have demonstrated in a recent work [94] that it is possible to differentiate path integrals directly (i.e., with no reparameterization) using a more general transport relation [9], the resulting derivative is more complicated and requires handling more types of discontinuities such as topological boundaries of object surfaces and discontinuities of surface normals or UV coordinates. In contrast, our material-form path integrals are defined over parameter-independent material path spaces $\hat{\boldsymbol{\Omega}}$. This allows us to differentiate material-form path integrals directly using Reynolds transport theorem. Additionally, when handling discontinuities (of measurement contributions), only discontinuities emerging from visibilities need to be tracked (as other discontinuities tend to be independent of the parameter θ in the reference configurations).

Compared with previous methods based on Monte Carlo edge sampling [45, 95], our formulation allows the *interior* integral to be estimated using path sampling techniques beyond unidirectional path tracing. Moreover, for Monte Carlo estimation of the *boundary* term that requires drawing boundary segments, our formulation leads to new algorithms that do not rely on silhouette detection—which is required by edge sampling methods but can be prohibitively expensive for complex scenes. We will present our new Monte Carlo estimators in the later chapters.

4.2 Supporting Zero-Measure Sources and Detectors

Our derivations of the differential path integral above assume that both the emission L_e and the sensor importance W_e to be defined over some surfaces—which is the case for area lights and cameras with finite (but nonzero) apertures.

On the other hand, zero-measure detectors and sources such as perspective pinhole cameras and point lights are ubiquitous in computer graphics and vision. In the following, we discuss how these models can be incorporated in our framework.

Perspective pinhole cameras and point lights. For a pinhole camera with the **center of projection** $\boldsymbol{x}_\mathrm{cam} \in \mathcal{V}$, any light transport path must terminate at $\boldsymbol{x}_\mathrm{cam}$ to have a nonzero measurement contribution. For a light path $(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \boldsymbol{x}_\mathrm{cam})$, the sensor importance of the pinhole camera equals[1]

$$W_\mathrm{e}^\mathrm{pinhole}(\boldsymbol{x}_N) := \frac{G(\boldsymbol{x}_N \leftrightarrow \boldsymbol{x}_\mathrm{cam})\, \mathcal{P}(\boldsymbol{x}_N^\perp)}{(\boldsymbol{n}_\mathrm{cam} \cdot \overrightarrow{\boldsymbol{x}_\mathrm{cam}\, \boldsymbol{x}_N})^3}, \tag{4.18}$$

where G is the geometric term of Eq. (2.28), $\boldsymbol{n}_\mathrm{cam}$ is the camera's **axis of projection**, and

[1] The geometric term G resulting from the change of measure in Eq. (2.11) is absorbed into Eq. (4.18) for defining the sensor importance of a pinhole camera.

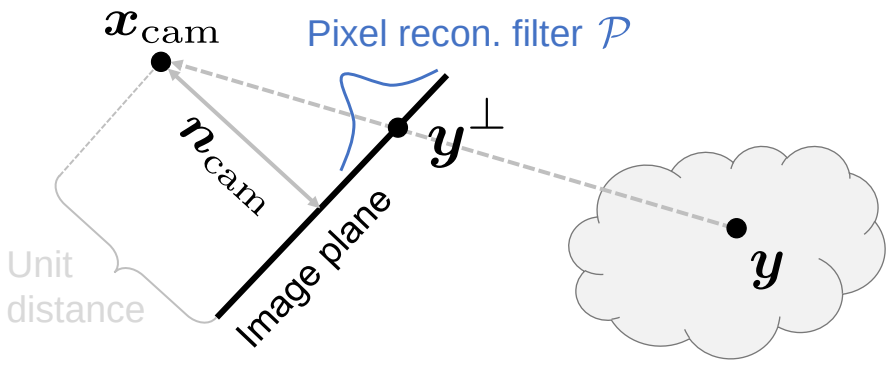


Figure 4.3: A **perspective pinhole camera** is specified by its center of projection $\boldsymbol{x}_{\text{cam}} \in \mathcal{V}$, axis of projection $\boldsymbol{n}_{\text{cam}} \in \mathbb{S}^2$, and per-pixel reconstruction filters h defined over an imaginary image plane.

$\overrightarrow{\boldsymbol{x}_{\text{cam}}\,\boldsymbol{x}_N}$ denotes the unit vector pointing from $\boldsymbol{x}_{\text{cam}}$ toward $\boldsymbol{x}_N$. In addition, as illustrated in Figure 4.3, $\boldsymbol{x}_N^{\perp}$ is the projection of $\boldsymbol{x}_N$ on the image plane (that is assumed to be unit-distance from $\boldsymbol{x}_{\text{cam}}$ and perpendicular to $\boldsymbol{n}_{\text{cam}}$), and $\mathcal{P}$ indicates the **pixel reconstruction filter**—which we assume to be C^0.

Since restricting the last vertex of a light path to be exactly at $\boldsymbol{x}_{\text{cam}}$ requires introducing Dirac delta functions to the sensor importance, we do not treat $\boldsymbol{x}_{\text{cam}}$ as an endpoint of all light paths. Instead, we encode its contributions in the sensor importance as follows. For any path $(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ with $\boldsymbol{x}_N$ being a standard surface or volume vertex, we set

$$W_{\text{e}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N) := f_{\text{v}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N \to \boldsymbol{x}_{\text{cam}})\, W_{\text{e}}^{\text{pinhole}}(\boldsymbol{x}_N), \tag{4.19}$$

where f_{v} is defined in Eq. (2.27). Then, as demonstrated in Figure 4.4–(a), the measurement contribution of a light path $(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N)$ with the detector importance of Eq. (4.19) equals that of $(\boldsymbol{x}_0, \ldots, \boldsymbol{x}_N, \boldsymbol{x}_{\text{cam}})$ with Eq. (4.18).

Similarly, with a (uniform) point light located at $\boldsymbol{x}_{\text{src}}$, we encode the contributions related to $\boldsymbol{x}_{\text{src}}$ in the emission L_{e} by setting

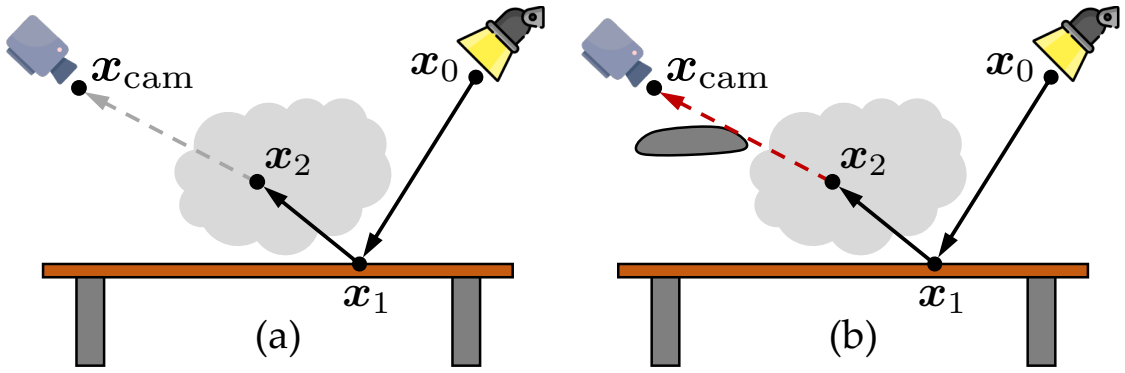


Figure 4.4: **Perspective pinhole camera:** (a) To support this camera model, we encode the contribution of the segment $\overline{\boldsymbol{x}_N\,\boldsymbol{x}_{\text{cam}}}$ illustrated as gray dashed arrows in the detector importance function via Eq. (4.19). (b) We also allow $\overline{\boldsymbol{x}_N\,\boldsymbol{x}_{\text{cam}}}$ to be a boundary segment to capture the additional discontinuities introduced by this segment.

$$L_{\text{e}}(\boldsymbol{x}_0 \to \boldsymbol{x}_1) := f_{\text{v}}(\boldsymbol{x}_{\text{src}} \to \boldsymbol{x}_0 \to \boldsymbol{x}_1)\, G(\boldsymbol{x}_{\text{src}} \leftrightarrow \boldsymbol{x}_0)\, I_{\text{src}}, \tag{4.20}$$

where I_{src} denotes the intensity of the point light.

Our formulations of Eqs. (4.19) and (4.20) essentially make all surface or volume points on the detector and the source, respectively, allowing pinhole cameras and point lights to be handled without introducing Dirac deltas to measurement contribution functions.

Handling discontinuities. The inclusion of the geometric term $G(\boldsymbol{x}_N \leftrightarrow \boldsymbol{x}_{\text{cam}})$ in Eq. (4.19) and $G(\boldsymbol{x}_{\text{src}} \leftrightarrow \boldsymbol{x}_0)$ in Eq. (4.20) can violate Assumption **A.2** (that requires W_{e} and L_{e} to be continuous). Fortunately, this can be handled easily by including a new set of material boundary paths in the *boundary* term of the differential path integral (4.14).

Specifically, for pinhole cameras, we consider $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ such that $\boldsymbol{p}_N$ is a discontinuity point of $G(\boldsymbol{x}_N \leftrightarrow \boldsymbol{x}_{\text{cam}})$. In other words, we allow $\overline{\boldsymbol{x}_N\,\boldsymbol{x}_{\text{cam}}}$ to be a boundary segment (see Figure 4.4-b). Similarly, when handling point lights, we track discontinuities of $\boldsymbol{x}_0$ such that $\overline{\boldsymbol{x}_{\text{src}}\,\boldsymbol{x}_0}$ is effectively a boundary segment.

Other zero-measure detectors and sources. Using the formulations outlined in Eqs. (4.19) and (4.20), other zero-measure detectors (e.g., orthographic cameras) and sources (e.g., directional lights) can be handled in a similar manner. In case of a directional light with incident direction $\boldsymbol{\omega}_{\text{src}}$, we can encode its contributions in the emission by letting

$$L_{\text{e}}(\boldsymbol{x}_0 \to \boldsymbol{x}_1) \coloneqq f_{\text{v}}(\boldsymbol{x}_0, \boldsymbol{\omega}_{\text{src}}, \overrightarrow{\boldsymbol{x}_0\boldsymbol{x}_1}) \, \mathbb{V}(\boldsymbol{x}_0, \boldsymbol{\omega}_{\text{src}}) \, I_{\text{src}}, \tag{4.21}$$

where $\mathbb{V}(\boldsymbol{x}_0, \boldsymbol{\omega}_{\text{src}})$, which can be discontinuous with respect to $\boldsymbol{x}_0$, indicates whether a ray with origin $\boldsymbol{x}_0$ and direction $\boldsymbol{\omega}_{\text{src}}$ can reach infinity without being occluded.

Chapter 5

Monte Carlo Estimation of Differential Path Integrals

Similar to the path integral formulation that has led to advanced Monte Carlo solutions (e.g., bidirectional path tracing [82]) for forward rendering, our differential path integral established in Chapter 4 serves as a mathematical foundation for the development of Monte Carlo differentiable rendering techniques.

In this chapter, we introduce new unbiased and consistent Monte Carlo estimators for differential path integrals. We focus on the problem of estimating $\frac{\mathrm{d}I}{\mathrm{d}\theta}\big|_{\theta=\theta_0}$ for some user-specified $\theta_0 \in \mathbb{R}$ with the reference configurations and motion as described in §3.3.

Because of the completely separated *interior* and *boundary* components of the differential path integral (4.14), we compute these terms independently using Monte Carlo estimators that we present in §5.1 and §5.2, respectively. Additionally, we validate and demonstrate the effectiveness of our Monte Carlo estimators in §5.3.

5.1 Estimating the Interior Integral

When estimating derivatives at some $\theta = \theta_0$, the *interior* component of the differential path integral expressed in Eq. (4.14) has the form

$$\int_{\hat{\mathbf{\Omega}}} \frac{\mathrm{d}}{\mathrm{d}\theta} \hat{f}(\bar{\boldsymbol{p}})\Big|_{\theta=\theta_0} \mathrm{d}\mu(\bar{\boldsymbol{p}}). \tag{5.1}$$

We estimate this term using Monte Carlo integration by (i) sampling a material path $\bar{\boldsymbol{p}} \in \hat{\mathbf{\Omega}}$ with some probability density $\mathrm{pdf}(\bar{\boldsymbol{p}})$; and (ii) evaluating the (single-sample) estimator $\frac{1}{\mathrm{pdf}(\bar{\boldsymbol{p}})} \left(\frac{\mathrm{d}}{\mathrm{d}\theta} \hat{f}(\bar{\boldsymbol{p}})\Big|_{\theta=\theta_0} \right)$. In the following, we discuss each step in more details.

Path sampling. With our choice of reference and motion discussed in §3.3, the material path space $\hat{\mathbf{\Omega}}$ coincides with the ordinary one $\mathbf{\Omega}(\theta_0)$. This allows us to sample the material path $\bar{\boldsymbol{p}}$ by re-purposing forward-rendering techniques. In practice, we use *unidirectional* or *bidirectional* path tracing for interfacial light transport and *volumetric path tracing* algorithm for volume rendering. We note that, since the path $\bar{\boldsymbol{p}}$ is considered independent of the parameter θ, the sampling process does not needs to be differentiated.

Differentiating measurement contributions. With the material path $\bar{\boldsymbol{p}}$ constructed, we compute the corresponding spatial path $\bar{\boldsymbol{x}} \in \mathbf{\Omega}$ by setting $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \theta)$ for each vertex $\boldsymbol{p}_n$ of $\bar{\boldsymbol{p}}$. At $\theta = \theta_0$, since the mapping $\mathtt{X}(\cdot, \theta_0)$ reduces to the identity map, $\boldsymbol{x}_n$ takes the same value as $\boldsymbol{p}_n$ for all n. On the other hand, the derivative $\frac{\mathrm{d}}{\mathrm{d}\theta}\boldsymbol{x}_n\big|_{\theta=\theta_0}$—which can be obtained by differentiating $\mathtt{X}(\boldsymbol{p}_n, \theta)$ with respect to θ—is generally nonzero. In practice, as discussed in §3.3.2, the $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \theta)$ can be implemented using barycentric interpolation with differentiable (triangle or tetrahedral) mesh vertex positions and constant (i.e., "detached") barycentric coordinates. Lastly, we obtain $\frac{\mathrm{d}}{\mathrm{d}\theta} \hat{f}(\bar{\boldsymbol{p}})\Big|_{\theta=\theta_0}$ using differentiable evaluation of the material measurement contribution $\hat{f}(\bar{\boldsymbol{p}})$.

Further Challenges

Although the process discussed above allows unbiased and consistent estimation of the interior integral of Eq. (5.1), several challenges remain.

Glossy BSDFs and pixel reconstruction filters. Although Eq. (5.1) can be estimated by sampling material paths $\bar{\boldsymbol{p}}$ using re-purposed forward-rendering techniques, the estimates can suffer from high variance with the presence of, for example, glossy materials. We will introduce new Monte Carlo methods that leverage *antithetic sampling* to efficiently handle derivatives of glossy BSDFs and pixel reconstruction filters in §6.2 and §6.3, respectively.

Efficient differentiation. Many, if not most, practical problems involves many (e.g., 10^6–10^9) parameters. One commonly resorts to the *reverse-mode* automatic differentiation to compute all requested derivatives at once. Unfortunately, a severe limitation of reverse-mode approaches is the requirement of storing detailed transcripts of intermediate computation steps. In physics-based differentiable rendering, the output is an image consisting of many pixels, making the storage of computation graph expensive or even infeasible. In Chapter 7, we will present a technique that allows efficient differentiation of material measurement contributions without storing full computation graphs.

5.2 Estimating the Boundary Integral

We now consider the estimation of the *boundary* component of our differential path integral (4.14) given by

$$\int_{\partial\hat{\boldsymbol{\Omega}}} \Delta\hat{f}_K(\bar{\boldsymbol{p}})\, v_\perp(\boldsymbol{p}_K)\, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}}). \tag{5.2}$$

As explained in §4.1, this integral is over the boundary path space $\partial\hat{\mathbf{\Omega}}$—which is unique to our formulation—comprised of material boundary paths $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ that are identical to the ordinary ones except for containing exactly one material boundary segment $\overline{\boldsymbol{p}_{K-1}\,\boldsymbol{p}_K}$. The two endpoints of a material boundary segment must satisfy that the corresponding spatial points $\boldsymbol{x}_{K-1} = \mathtt{X}(\boldsymbol{p}_{K-1}, \theta)$ and $\boldsymbol{x}_{K-1} = \mathtt{X}(\boldsymbol{p}_{K-1}, \theta)$ are located on visibility boundaries with respect to each other.

This requirement, unfortunately, makes the sampling of material boundary paths challenging. For instance, provided one endpoint $\boldsymbol{p}_{K-1}$ of a material boundary segment, sampling the other one $\boldsymbol{p}_K$ requires identifying silhouette with respect to $\boldsymbol{x}_{K-1} = \mathtt{X}(\boldsymbol{p}_{K-1}, \theta)$. To this end, several previous methods [45, 95] rely on explicit silhouette detection, which scales poorly to virtual scenes with complex geometries.

In what follows, we introduce a new Monte Carlo method to efficiently estimate Eq. (5.2) while avoiding explicit silhouette detections.

5.2.1 Multi-Dimensional Form of the Boundary Integral

To efficiently sample a material boundary path $\bar{\boldsymbol{p}} \in \partial\hat{\mathbf{\Omega}}$, we introduce a *multi-directional* process that reconstructs $\bar{\boldsymbol{p}}$ from the boundary segment. For notational convenience, we rename the vertices of $\bar{\boldsymbol{p}}$ as:

$$\bar{\boldsymbol{p}} = (\boldsymbol{p}_s^{\mathrm{S}}, \boldsymbol{p}_{s-1}^{\mathrm{S}}, \ldots, \boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}, \boldsymbol{p}_1^{\mathrm{D}}, \ldots, \boldsymbol{p}_t^{\mathrm{D}}), \tag{5.3}$$

such that $\boldsymbol{p}_s^{\mathrm{S}}$ and $\boldsymbol{p}_t^{\mathrm{D}}$ are located, respectively, on the source and the detector;[1] and $\overline{\boldsymbol{p}_0^{\mathrm{S}}\,\boldsymbol{p}_0^{\mathrm{D}}}$ is the material boundary segment. Similarly, we rename vertices of the corresponding (spatial) boundary path as $\bar{\boldsymbol{x}} = (\boldsymbol{x}_s^{\mathrm{S}}, \boldsymbol{x}_{s-1}^{\mathrm{S}}, \ldots, \boldsymbol{x}_0^{\mathrm{S}}, \boldsymbol{x}_0^{\mathrm{D}}, \boldsymbol{x}_1^{\mathrm{D}}, \ldots, \boldsymbol{x}_t^{\mathrm{D}})$, as illustrated in Figure 5.1.

[1] When the detector is a pinhole camera, as discussed in §4.2, $\boldsymbol{p}_t^{\mathrm{D}}$ is further connected to the camera's center of projection $\boldsymbol{p}_{\mathrm{cam}}^{(0)}$ (instead of being on the sensor).

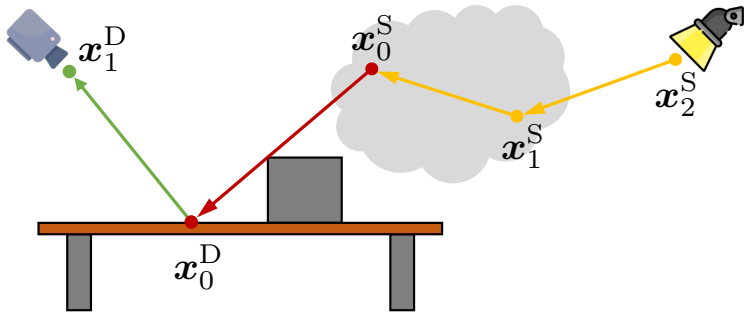


Figure 5.1: We **rename the vertices** of a boundary path such that $\overline{\boldsymbol{x}_0^{\mathrm{S}}\,\boldsymbol{x}_0^{\mathrm{D}}}$ is the boundary segment (illustrated in red). The source and detector subpaths are shown in yellow and green, respectively. The arrows in this figure illustrate the physical flow of light (that is from the source to the detector) and do not indicate how the subpaths are constructed.

Further, we factorize the integrand of the *boundary* integral in Eq. (5.2) into contributions of the segment $\overline{\boldsymbol{p}_0^{\mathrm{S}}\,\boldsymbol{p}_0^{\mathrm{D}}}$, the **source subpath** $\bar{\boldsymbol{p}}^{\mathrm{S}} := (\boldsymbol{p}_s^{\mathrm{S}}, \ldots, \boldsymbol{p}_0^{\mathrm{S}})$, and the **detector subpath** $\bar{\boldsymbol{p}}^{\mathrm{D}} := (\boldsymbol{p}_0^{\mathrm{D}}, \ldots, \boldsymbol{p}_t^{\mathrm{D}})$, respectively:

$$\Delta\hat{f}_K(\bar{\boldsymbol{p}})\, v_\perp(\boldsymbol{p}_K) = \underbrace{\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})}_{\text{boundary seg.}}\, \underbrace{\hat{f}^{\mathrm{S}}(\bar{\boldsymbol{p}}^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{D}})}_{\text{src. subpath}}\, \underbrace{\hat{f}^{\mathrm{D}}(\bar{\boldsymbol{p}}^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}})}_{\text{det. subpath}}, \tag{5.4}$$

where

$$\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}) := \Delta G(\boldsymbol{x}_0^{\mathrm{S}} \leftrightarrow \boldsymbol{x}_0^{\mathrm{D}})\, v_\perp(\boldsymbol{p}_0^{\mathrm{D}}), \tag{5.5}$$

$$\hat{f}^{\mathrm{S}}(\bar{\boldsymbol{p}}^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{D}}) := \hat{f}_{\mathrm{v}}(\boldsymbol{p}_1^{\mathrm{S}} \to \boldsymbol{p}_0^{\mathrm{S}} \to \boldsymbol{p}_0^{\mathrm{D}}) \textstyle\prod_{n=1}^{s} \hat{f}_{\mathrm{v}}(\boldsymbol{p}_{n+1}^{\mathrm{S}} \to \boldsymbol{p}_n^{\mathrm{S}} \to \boldsymbol{p}_{n-1}^{\mathrm{S}})\, G(\boldsymbol{x}_n^{\mathrm{S}} \leftrightarrow \boldsymbol{x}_{n-1}^{\mathrm{S}}), \tag{5.6}$$

$$\hat{f}^{\mathrm{D}}(\bar{\boldsymbol{p}}^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}}) := \hat{f}_{\mathrm{v}}(\boldsymbol{p}_0^{\mathrm{S}} \to \boldsymbol{p}_0^{\mathrm{D}} \to \boldsymbol{p}_1^{\mathrm{D}}) \textstyle\prod_{n=1}^{t} \hat{f}_{\mathrm{v}}(\boldsymbol{p}_{n-1}^{\mathrm{D}} \to \boldsymbol{p}_n^{\mathrm{D}} \to \boldsymbol{p}_{n+1}^{\mathrm{D}})\, G(\boldsymbol{x}_{n-1}^{\mathrm{D}} \leftrightarrow \boldsymbol{x}_n^{\mathrm{D}}). \tag{5.7}$$

In Eqs. (5.6) and (5.7), G is the generalized geometric term. Additionally, $\hat{f}_{\mathrm{v}}$ captures both per-vertex contribution f_{v} of Eq. (2.27) and the Jacobian determinant J of Eq. (3.11). That is, for any $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3 \in \mathcal{B}_{\mathcal{V}}$ and $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \theta)$ for $n = 1, 2, 3$:

$$\hat{f}_{\mathrm{v}}(\boldsymbol{p}_1 \to \boldsymbol{p}_2 \to \boldsymbol{p}_3) := f_{\mathrm{v}}(\boldsymbol{x}_1 \to \boldsymbol{x}_2 \to \boldsymbol{x}_3)\, J(\boldsymbol{p}_2). \tag{5.8}$$

With its integrand expressed using Eq. (5.4), we establish the *boundary* integral of Eq. (5.2) in its *multi-directional form* as:[2]

$$\iint \hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}) \left[\int \hat{f}^{\mathrm{S}}(\bar{\boldsymbol{p}}^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{D}}) \, \mathrm{d}\bar{\boldsymbol{p}}_0^{\mathrm{S}} \right] \left[\int \hat{f}^{\mathrm{D}}(\bar{\boldsymbol{p}}^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}}) \, \mathrm{d}\bar{\boldsymbol{p}}_0^{\mathrm{D}} \right] \mathrm{d}\boldsymbol{p}_0^{\mathrm{D}} \, \mathrm{d}\boldsymbol{p}_0^{\mathrm{S}}, \tag{5.9}$$

where the outer integral is over the material boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$. Additionally, $\bar{\boldsymbol{p}}_0^{\mathrm{S}}$ and $\bar{\boldsymbol{p}}_0^{\mathrm{D}}$ denote the source and detector subpaths with endpoints $\boldsymbol{p}_0^{\mathrm{S}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$ of the boundary segment excluded, respectively. That is, $\bar{\boldsymbol{p}}_0^{\mathrm{S}} := (\boldsymbol{p}_s^{\mathrm{S}}, \ldots, \boldsymbol{p}_1^{\mathrm{S}})$ and $\bar{\boldsymbol{p}}_0^{\mathrm{D}} := (\boldsymbol{p}_1^{\mathrm{D}}, \ldots, \boldsymbol{p}_t^{\mathrm{D}})$.

In the case of interfacial light transport, both $\boldsymbol{p}_0^{\mathrm{S}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$ would always be surface vertices (Figure 4.2-a). With the presence of participating media, on the other hand, they both can be either a surface or a volume vertex, leading to three extra combinations (Figure 4.2-bcd).

5.2.2 Change of Variables

To facilitate efficient sampling of the material boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$, we apply a series of changes of variables to Eq. (5.9) as follows. First, we use the predetermined differentiable mapping $\mathtt{X}(\cdot, \theta)$ to make the outer integral to be over the corresponding (spatial) boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$. In principle, this requires computing the Jacobian determinant $\left\| \frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{S}} \, \mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\boldsymbol{x}_0^{\mathrm{S}} \, \mathrm{d}\boldsymbol{x}_0^{\mathrm{D}}} \right\|$ based on the mapping $\mathtt{X}(\cdot, \theta)$. In practice, because of our choice of reference configurations, the Jacobian determinant simply equals one (i.e., $\left\| \frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{S}} \, \mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\boldsymbol{x}_0^{\mathrm{S}} \, \mathrm{d}\boldsymbol{x}_0^{\mathrm{D}}} \right\| \equiv 1$).

Then, let $\boldsymbol{x}^{\mathrm{B}}$ be the intersection point between the boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ and the union of all object surfaces and $\boldsymbol{\omega}_{\mathrm{B}} = \overrightarrow{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ be the direction of this segment. We apply another change of variable to make the outer integral of Eq. (5.9) to be with respect to $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$

[2] We omit the integral domains and measures in Eq. (5.9) for notational simplicity.

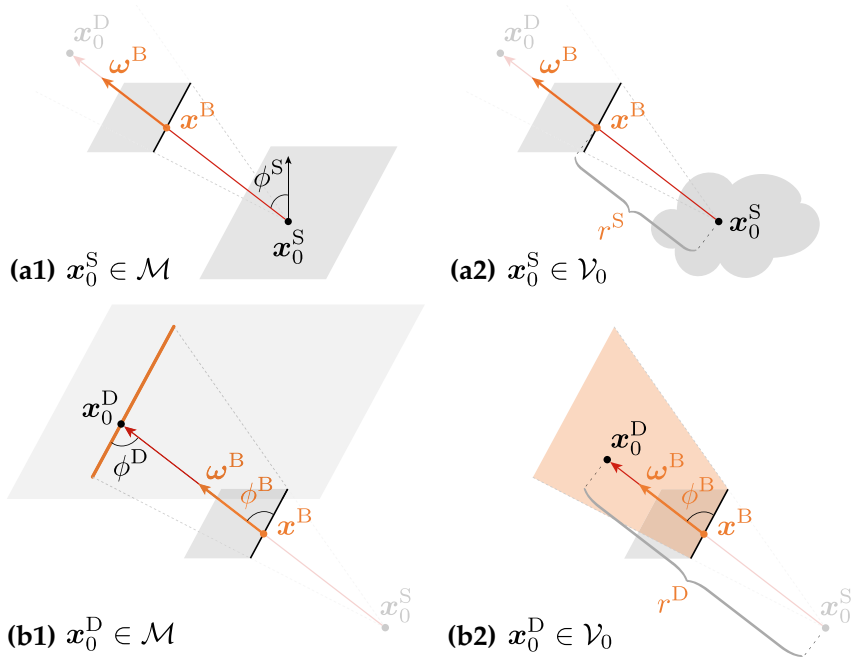


Figure 5.2: Illustrations for deriving the **change-of-variable** Jacobian determinants expressed in Eqs. (5.15)–(5.21).

(as opposed to $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$). That is, we aim to rewrite Eq. (5.9) as:

$$\iint \hat{F}^{\mathrm{B}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})\, \mathrm{d}\boldsymbol{\omega}_{\mathrm{B}}\, \mathrm{d}\boldsymbol{x}^{\mathrm{B}}. \tag{5.10}$$

We note that the point $\boldsymbol{x}^{\mathrm{B}}$ is not a vertex of the resulting boundary path—we will only use it for sampling $\overline{\boldsymbol{x}_0^{\mathrm{S}}\, \boldsymbol{x}_0^{\mathrm{D}}}$.

In what follows, we derive the integrand $\hat{F}^{\mathrm{B}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ of the rewritten *boundary* integral in Eq. (5.10). This term needs to contain the Jacobian determinant corresponding to the change of variable from $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$ to $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$. We base our derivations on the assumption that all surfaces in the scene are represented using triangle meshes. In this case, $\boldsymbol{x}^{\mathrm{B}}$ will always belong to an edge of some face of the mesh.

To derive the Jacobian determinant, we consider $\boldsymbol{x}_0^{\mathrm{S}}$ fixed and $\boldsymbol{x}_0^{\mathrm{D}}$ located on a visibility boundary with respect to $\boldsymbol{x}_0^{\mathrm{S}}$. Then, $\mathrm{d}\boldsymbol{x}_0^{\mathrm{S}}$ and $\mathrm{d}\boldsymbol{x}_0^{\mathrm{D}}$ can be related to $\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})$ and $\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})$, respectively, as follows.

- When $\boldsymbol{x}_0^{\mathrm{S}}$ is a surface vertex, as illustrated in Figure 5.2-a1, we have

$$\frac{\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{S}})\,|\cos\phi^{\mathrm{S}}|}{(r^{\mathrm{S}})^2} = \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}}), \tag{5.11}$$

  where $r^{\mathrm{S}} := \|\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}}\|$ is the distance between $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{x}_0^{\mathrm{S}}$; ϕ^{S} is the angle between $\boldsymbol{\omega}_{\mathrm{B}}$ and the surface normal at $\boldsymbol{x}_0^{\mathrm{S}}$.

- When $\boldsymbol{x}_0^{\mathrm{S}}$ is a volume vertex, as shown in Figure 5.2-a2, we have

$$\frac{\mathrm{d}V(\boldsymbol{x}_0^{\mathrm{S}})}{(r^{\mathrm{S}})^2} = \mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})\,\mathrm{d}r^{\mathrm{S}}. \tag{5.12}$$

- When $\boldsymbol{x}_0^{\mathrm{D}}$ is a surface vertex, for $\overline{\boldsymbol{x}_0^{\mathrm{S}}\,\boldsymbol{x}_0^{\mathrm{D}}}$ to be a boundary segment with $\boldsymbol{x}_0^{\mathrm{S}}$ fixed, $\boldsymbol{x}_0^{\mathrm{D}}$ must belong to a curve (see Figure 5.2-b1). In this case, we have

$$\frac{\mathrm{d}\ell(\boldsymbol{x}_0^{\mathrm{D}})\,\sin\phi^{\mathrm{D}}}{r^{\mathrm{D}}} = \frac{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\sin\phi^{\mathrm{B}}}{r^{\mathrm{S}}}, \tag{5.13}$$

  where $r^{\mathrm{D}} := \|\boldsymbol{x}_0^{\mathrm{D}} - \boldsymbol{x}_0^{\mathrm{S}}\|$ is the distance between $\boldsymbol{x}_0^{\mathrm{D}}$ and $\boldsymbol{x}_0^{\mathrm{S}}$, and ϕ^{D} is the angle between $\boldsymbol{\omega}_{\mathrm{B}}$ and the curve's tangent at $\boldsymbol{x}_0^{\mathrm{D}}$.

- When $\boldsymbol{x}_0^{\mathrm{D}}$ is a volume vertex, it resides on a surface determined by the point $\boldsymbol{x}^{\mathrm{B}}$ and direction $\boldsymbol{\omega}_{\mathrm{B}}$ (see Figure 5.2-b2). It follows that

$$\frac{\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{D}})}{r^{\mathrm{D}}} = \frac{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\sin\phi^{\mathrm{B}}}{r^{\mathrm{S}}}\,\mathrm{d}r^{\mathrm{D}}. \tag{5.14}$$

Based on the relations given by Eqs. (5.11)–(5.14), we derive the Jacobian determinants for changes of variables from $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$ to $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$ (as well as r^{S}, r^{D} when needed) and the corresponding integrand $\hat{F}^{\mathrm{B}}$. Specifically:

- When both $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$ are surface vertices, according to Eqs. (5.11) and (5.13), we

have

$$\left\| \frac{\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{S}})\,\mathrm{d}\ell(\boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})} \right\| = r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{\sin\phi^{\mathrm{D}}\,|\cos\phi^{\mathrm{S}}|}, \tag{5.15}$$

and

$$\hat{F}^{\,\mathrm{B}} = \hat{f}^{\,\mathrm{B}}\, r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{\sin\phi^{\mathrm{D}}\,|\cos\phi^{\mathrm{S}}|} \left[\int \hat{f}^{\,\mathrm{S}}\,\mathrm{d}\mu\right] \left[\int \hat{f}^{\,\mathrm{D}}\,\mathrm{d}\mu\right]. \tag{5.16}$$

- With $\boldsymbol{x}_0^{\mathrm{S}}$ being a surface vertex and $\boldsymbol{x}_0^{\mathrm{D}}$ a volume vertex, multiplying Eqs. (5.11) and (5.14) yields

$$\left\| \frac{\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{S}})\,\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})\,\mathrm{d}r^{\mathrm{D}}} \right\| = r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{|\cos\phi^{\mathrm{S}}|}, \tag{5.17}$$

and

$$\hat{F}^{\,\mathrm{B}} = \int_{r^{\mathrm{S}}}^{\infty} \hat{f}^{\,\mathrm{B}}\, r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{|\cos\phi^{\mathrm{S}}|} \left[\int \hat{f}^{\,\mathrm{S}}\,\mathrm{d}\mu\right] \left[\int \hat{f}^{\,\mathrm{D}}\,\mathrm{d}\mu\right] \mathrm{d}r^{\mathrm{D}}. \tag{5.18}$$

- With $\boldsymbol{x}_0^{\mathrm{S}}$ being a volume vertex and $\boldsymbol{x}_0^{\mathrm{D}}$ a surface vertex, multiplying Eqs. (5.12) and (5.13) gives

$$\left\| \frac{\mathrm{d}V(\boldsymbol{x}_0^{\mathrm{S}})\,\mathrm{d}\ell(\boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})\,\mathrm{d}r^{\mathrm{S}}} \right\| = r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{\sin\phi^{\mathrm{D}}}, \tag{5.19}$$

and

$$\hat{F}^{\,\mathrm{B}} = \int_{0}^{\infty} \hat{f}^{\,\mathrm{B}}\, r^{\mathrm{S}}\, r^{\mathrm{D}}\, \frac{\sin\phi^{\mathrm{B}}}{\sin\phi^{\mathrm{D}}} \left[\int \hat{f}^{\,\mathrm{S}}\,\mathrm{d}\mu\right] \left[\int \hat{f}^{\,\mathrm{D}}\,\mathrm{d}\mu\right] \mathrm{d}r^{\mathrm{S}}. \tag{5.20}$$

- Lastly, when both $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$ are volume vertices, according to Eqs. (5.12) and (5.14), we have

$$\left\| \frac{\mathrm{d}V(\boldsymbol{x}_0^{\mathrm{S}})\,\mathrm{d}A(\boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{d}\ell(\boldsymbol{x}^{\mathrm{B}})\,\mathrm{d}\sigma(\boldsymbol{\omega}_{\mathrm{B}})\,\mathrm{d}r^{\mathrm{S}}\,\mathrm{d}r^{\mathrm{D}}} \right\| = r^{\mathrm{S}}\, r^{\mathrm{D}}\, \sin\phi^{\mathrm{B}}, \tag{5.21}$$

and

$$\hat{F}^{\,\mathrm{B}} = \int_{0}^{\infty} \int_{r^{\mathrm{S}}}^{\infty} \hat{f}^{\,\mathrm{B}}\, r^{\mathrm{S}}\, r^{\mathrm{D}}\, \sin\phi^{\mathrm{B}} \left[\int \hat{f}^{\,\mathrm{S}}\,\mathrm{d}\mu\right] \left[\int \hat{f}^{\,\mathrm{D}}\,\mathrm{d}\mu\right] \mathrm{d}r^{\mathrm{D}}\,\mathrm{d}r^{\mathrm{S}}. \tag{5.22}$$

Algorithm 1: Monte Carlo estimator of the *boundary* integral (5.9)

1 `EstimateBoundaryIntegral()`
2 **begin**
  `/* Sample boundary segment */`
3 Draw $(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}}) \sim \mathbb{P}$;
4 $(\boldsymbol{x}_0^{\mathrm{S}}, \mathrm{pdf}^{\mathrm{S}}) \leftarrow$ `sampleInteraction`$(\boldsymbol{x}^{\mathrm{B}}, -\boldsymbol{\omega}_{\mathrm{B}})$;
5 $(\boldsymbol{x}_0^{\mathrm{D}}, \mathrm{pdf}^{\mathrm{D}}) \leftarrow$ `sampleInteraction`$(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$;
  `/* Compute Jacobian determinant */`
6 $r^{\mathrm{S}} \leftarrow \|\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}}\|$; $r^{\mathrm{D}} \leftarrow \|\boldsymbol{x}^{\mathrm{D}} - \boldsymbol{x}_0^{\mathrm{S}}\|$;
7 $J^{\mathrm{B}} \leftarrow r^{\mathrm{S}}\, r^{\mathrm{D}}\, \sin\phi^{\mathrm{B}}$;
8 **if** $\boldsymbol{x}_0^{\mathrm{S}}$ *is a surface vertex* **then**
9 $J^{\mathrm{B}} \leftarrow J^{\mathrm{B}} / |\cos\phi^{\mathrm{S}}|$;
10 **end**
11 **if** $\boldsymbol{x}_0^{\mathrm{D}}$ *is a surface vertex* **then**
12 $J^{\mathrm{B}} \leftarrow J^{\mathrm{B}} / \sin\phi^{\mathrm{D}}$;
13 **end**
  `/* Evaluate boundary segment */`
14 $\boldsymbol{p}_0^{\mathrm{S}} \leftarrow \boldsymbol{x}_0^{\mathrm{S}}$; $\boldsymbol{p}_0^{\mathrm{D}} \leftarrow \boldsymbol{x}_0^{\mathrm{D}}$;
15 $T^{\mathrm{B}} \leftarrow \dfrac{\hat{f}^{\mathrm{B}}(\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}})\, J^{\mathrm{B}}}{\mathbb{P}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})\, \mathrm{pdf}^{\mathrm{S}}\, \mathrm{pdf}^{\mathrm{D}}}$;
  `/* Sample and evaluate source & detector subpaths */`
16 $T^{\mathrm{S}} \leftarrow$ `EstimateSourcePath`$(\boldsymbol{p}_0^{\mathrm{S}}; \boldsymbol{p}_0^{\mathrm{D}})$;
17 $T^{\mathrm{D}} \leftarrow$ `EstimateDetectorPath`$(\boldsymbol{p}_0^{\mathrm{D}}; \boldsymbol{p}_0^{\mathrm{S}})$;
18 **return** $T^{\mathrm{B}}\, T^{\mathrm{S}}\, T^{\mathrm{D}}$;
19 **end**

5.2.3 Sampling Boundary Light Paths

Based on the rewritten form of Eq. (5.10), we develop a generalized multi-directional sampling algorithm (outlined in Algorithm 1) to estimate the *boundary* integral of Eq. (5.2).

Sampling boundary segment. Our algorithm starts with sampling a point $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{M}(\theta)$ and a direction $\boldsymbol{\omega}_{\mathrm{B}} \in \mathbb{S}^2$ from some predetermined probability density $\mathbb{P}$ (Line 3). Recall that $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$ will determine the boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}}\, \boldsymbol{x}_0^{\mathrm{D}}}$. When the surfaces $\mathcal{M}(\theta)$ are specified using triangle meshes, $\boldsymbol{x}^{\mathrm{B}}$ must belong to the union of of all face edges, which we denote as $\mathcal{E}(\theta) \subset \mathcal{M}(\theta)$. In addition, for each $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{E}(\theta)$, the direction $\boldsymbol{\omega}_{\mathrm{B}}$ needs to satisfy

the following two conditions:

- First, the line passing $\boldsymbol{x}^{\mathrm{B}}$ with direction $\boldsymbol{\omega}_{\mathrm{B}}$ does not penetrate $\mathcal{M}(\theta)$ at $\boldsymbol{x}^{\mathrm{B}}$. Precisely, as shown in Figure 5.3, if the face edge containing $\boldsymbol{x}^{\mathrm{B}}$ is shared by two faces with normal vectors $\boldsymbol{n}$ and $\boldsymbol{n}'$, we need to have $(\boldsymbol{\omega}_{\mathrm{B}} \cdot \boldsymbol{n})\,(\boldsymbol{\omega}_{\mathrm{B}} \cdot \boldsymbol{n}') < 0$.
- Second, `sampleInteraction`$(\boldsymbol{x}^{\mathrm{B}}, -\boldsymbol{\omega}_{\mathrm{B}})$ and `sampleInteraction`$(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ should successfully return some valid $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$, respectively. We will provide more details about `sampleInteraction` next.

Provided $\boldsymbol{x}^{\mathrm{B}} \in \mathcal{E}(\theta)$ and $\boldsymbol{\omega}_{\mathrm{B}} \in \mathbb{S}^2$, we obtain the two endpoints $\boldsymbol{x}_0^{\mathrm{S}}$ and $\boldsymbol{x}_0^{\mathrm{D}}$ of the boundary segment using the `sampleInteraction` function (Lines 4 and 5 of Algorithm 1). For any given $\boldsymbol{x}$ and $\boldsymbol{\omega}$, `sampleInteraction`$(\boldsymbol{x}, \boldsymbol{\omega})$ returns a randomly sampled volume or surface vertex at $\boldsymbol{x}+t\,\boldsymbol{\omega}$ (for some $t \in \mathbb{R}_{>0}$), accompanied with the corresponding probability density.

In practice, we follow the standard procedure in volumetric path tracing by first drawing a free-flight distance $t > 0$ from an exponential distribution with the PDF

$$p(t) = \sigma_{\mathrm{t}}(\boldsymbol{x} + t\boldsymbol{\omega}) \exp\left(-\int_0^t \sigma_{\mathrm{t}}(\boldsymbol{x} + s\boldsymbol{\omega})\,\mathrm{d}s\right)$$

where σ_{t} is the extinction coefficient. For heterogeneous media, this can be achieved using techniques like delta tracking [87] or differential ratio tracking [60] for strict unbiasedness. If the line segment connecting $\boldsymbol{x}$ and $(\boldsymbol{x} + t\,\boldsymbol{\omega})$ does not intersect any object surface—that is, $\{\boldsymbol{x} + s\,\boldsymbol{\omega} : 0 < s < t\} \cap \mathcal{M}(\theta) = \emptyset$—`sampleInteraction`$(\boldsymbol{x}, \boldsymbol{\omega})$ returns a volume vertex at $(\boldsymbol{x} + t\,\boldsymbol{\omega})$. Otherwise, the function returns the first intersection $(\boldsymbol{x} + t_0\,\boldsymbol{\omega})$ as a surface vertex where $t_0 = \inf\{0 < s < t : \boldsymbol{x} + s\,\boldsymbol{\omega} \in \mathcal{M}(\theta)\}$.

Sampling subpaths. With the boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}}\,\boldsymbol{x}_0^{\mathrm{D}}}$ drawn, we compute the corresponding Jacobian determinant J^{B} (Lines 6–12) based on Eqs. (5.15), (5.17), (5.19) and

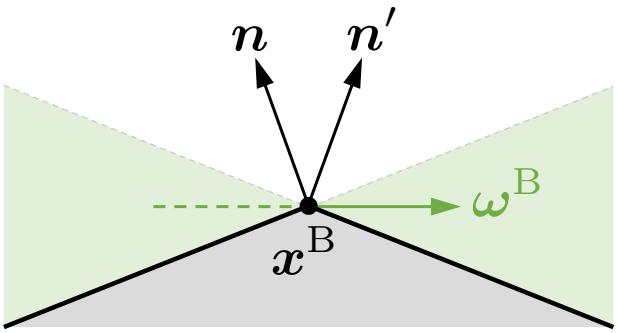


Figure 5.3: **Sampling the boundary segment:** When $\boldsymbol{x}^{\mathrm{B}}$ lies on an edge of a polygonal mesh that is shared by two faces with normals $\boldsymbol{n}$ and $\boldsymbol{n}'$, the direction $\boldsymbol{\omega}_{\mathrm{B}}$ needs to satisfy $(\boldsymbol{\omega}_{\mathrm{B}} \cdot \boldsymbol{n})(\boldsymbol{\omega}_{\mathrm{B}} \cdot \boldsymbol{n}') < 0$ (i.e., the green region) for the resulting segment $\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}$ to be a (spatial) boundary segment.

(5.21). This allows the contribution T^{B} of the boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}_0^{\mathrm{D}}}$ to be computed (Line 15). Lastly, we estimate the contributions of the source and detector subpaths $\bar{\boldsymbol{p}}^{\mathrm{S}}$ and $\bar{\boldsymbol{p}}^{\mathrm{D}}$, respectively, using standard techniques such as volumetric path tracing (Lines 16 and 17), completing our estimation of the *boundary* integral of Eq. (5.2).

Next-event estimation. To improve the efficiency of boundary-path sampling, we adopt next-event estimation (NEE), a technique widely used by forward rendering algorithms, as follows. We consider (material) **direct boundary paths** $\bar{\boldsymbol{p}} \in \bigcup_{N=2}^{\infty} \partial\hat{\boldsymbol{\Omega}}_{N,1}$, in analogy with *direct-illumination paths* in unidirectional path tracing, where $\partial\hat{\boldsymbol{\Omega}}_{N,1}$ is defined in Eq. (4.11). Then, a direct boundary path takes the form $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0^{\mathrm{S}}, \boldsymbol{p}_0^{\mathrm{D}}, \boldsymbol{p}_1^{\mathrm{D}}, \ldots)$ with the material boundary segment $\overline{\boldsymbol{p}_0^{\mathrm{S}} \boldsymbol{p}_0^{\mathrm{D}}}$. To construct a direct boundary path, instead of sampling $\boldsymbol{\omega}_{\mathrm{B}} \in \mathbb{S}^2$ after obtaining $\boldsymbol{x}^{\mathrm{B}}$ (Line 3), we sample the endpoint $\boldsymbol{x}_0^{\mathrm{S}}$ directly on the surface of a light source and set $\boldsymbol{\omega}_{\mathrm{B}} = \overrightarrow{\boldsymbol{x}_0^{\mathrm{S}} \boldsymbol{x}^{\mathrm{B}}}$. Further, since $\boldsymbol{p}_0^{\mathrm{S}} = \mathtt{P}(\boldsymbol{x}_0^{\mathrm{S}}, \theta)$ is already an endpoint of the path $\bar{\boldsymbol{p}}$, source subpaths do not need to be constructed from $\boldsymbol{p}_0^{\mathrm{S}}$.

Pinhole cameras and point lights. Given a boundary path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$, as discussed in §4.2, we allow $\overline{\boldsymbol{x}_N \boldsymbol{x}_{\mathrm{cam}}}$ to be an effective boundary segment—despite not treating $\boldsymbol{x}_{\mathrm{cam}}$ as a path vertex—to handle pinhole cameras (see Figure 4.4-b). In our multi-directional

sampling algorithm, the segment $\overline{\boldsymbol{x}_N\,\boldsymbol{x}_{\text{cam}}}$ can be obtained by connecting the sampled point $\boldsymbol{x}^{\text{B}} \in \mathcal{E}(\theta)$ to the camera's center of projection $\boldsymbol{x}_{\text{cam}}$. In this case, the direction $\boldsymbol{\omega}_{\text{B}}$ simply equals $\overrightarrow{\boldsymbol{x}^{\text{B}}\,\boldsymbol{x}_{\text{cam}}}$, and only the source subpath $\bar{\boldsymbol{p}}^{\text{S}}$ needs to be sampled. Similarly, when dealing with point lights, we allow $\overline{\boldsymbol{x}_{\text{src}}\,\boldsymbol{x}_0}$ to be an effective boundary segment. Then, $\boldsymbol{\omega}_{\text{B}} = \overrightarrow{\boldsymbol{x}_{\text{src}}\,\boldsymbol{x}^{\text{B}}}$, and only the detector subpath $\bar{\boldsymbol{p}}^{\text{D}}$ needs to be constructed.

5.2.4 Grid-Based Importance Sampling

In Algorithm 1, a key step is to sample $\boldsymbol{x}^{\text{B}} \in \mathcal{E}(\theta)$ and $\boldsymbol{\omega}_{\text{B}} \in \mathbb{S}^2$ (Line 3). To this end, a naive way to sample both $\boldsymbol{x}^{\text{B}}$ and $\boldsymbol{\omega}_{\text{B}}$ uniformly. Unfortunately, this can result in estimates of high variance, due to the complexity of the integrand $\hat{F}^{\text{B}}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}})$ defined in Eq. (5.10). Instead, we would like to importance sample $\boldsymbol{x}^{\text{B}}$ and $\boldsymbol{\omega}_{\text{B}}$ jointly, with a probability density $\mathbb{P}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}})$ proportional to the integrand. That is, $\mathbb{P}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}}) \propto \hat{F}^{\text{B}}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}})$.

Although this probability is difficult to compute analytically, we note that the total dimensionality of the domains from which $\boldsymbol{x}^{\text{B}}$ and $\boldsymbol{\omega}_{\text{B}}$ are drawn is only three. Specifically, the union of all face edges $\mathcal{E}(\theta)$ has dimensionality one; and the set of all valid directions $\boldsymbol{\omega}_{\text{B}}$, as illustrated in Figure 5.3, is a subset of $\mathbb{S}^2$ with dimensionality two. We take this advantage of low dimensionality to develop a simple method for importance sampling $\boldsymbol{x}^{\text{B}}$ and $\boldsymbol{\omega}_{\text{B}}$ as follows.

We represent the probability density $\mathbb{P}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}})$ as a piecewise constant function using a regular 3D grid, and precompute this grid during preprocessing. Then, for each cell C_i of the grid, the corresponding probability value $\mathbb{P}_i$ (before normalization) is given by

$$\mathbb{P}_i = \int_{C_i} \hat{F}^{\text{B}}(\boldsymbol{x}^{\text{B}}, \boldsymbol{\omega}_{\text{B}})\, \mathrm{d}\boldsymbol{\omega}_{\text{B}}\, \mathrm{d}\boldsymbol{x}^{\text{B}}, \tag{5.23}$$

where $\hat{F}^{\text{B}}$ is defined in Eqs. (5.16), (5.18), (5.20), and (5.22).

To efficiently evaluate Eq. (5.23), we approximate the two integrals $\int \hat{f}^{\mathrm{S}}\,\mathrm{d}\mu$ and $\int \hat{f}^{\mathrm{D}}\,\mathrm{d}\mu$—which are factors of the function $\hat{F}^{\mathrm{B}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$—by leveraging kernel density estimation using the pre-generated photon and importon maps. Specifically, given the boundary segment $\overline{\boldsymbol{x}_0^{\mathrm{S}}\,\boldsymbol{x}_0^{\mathrm{D}}}$, by performing a nearest-neighbor (NN) search in the photon map around $\boldsymbol{x}_0^{\mathrm{S}}$, we can approximate the contribution of the source subpath as

$$\int_{\hat{\boldsymbol{\Omega}}} \hat{f}^{\mathrm{S}}\,\mathrm{d}\mu \approx \frac{1}{A}\sum_{p} \hat{f}_{\mathrm{s}}(\boldsymbol{x}_0^{\mathrm{S}}, \boldsymbol{\omega}_p, \boldsymbol{\omega}_{\mathrm{B}})\,\Phi_p, \tag{5.24}$$

where A is the area/volume of the search neighborhood, Φ_p denotes the power of the p-th photon in the neighborhood, and $\boldsymbol{\omega}_p$ is the photon's incident direction. A similar estimate can be formed using the importon map for the contribution $\int_{\hat{\boldsymbol{\Omega}}} \hat{f}^{\mathrm{D}}\,\mathrm{d}\mu$ of the detector subpath.

We emphasize that, even though our estimate of $\mathbb{P}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ is biased, the resulting estimator of the *boundary* integral remains unbiased, as $\mathbb{P}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ is only used to importance sample $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$. In practice, we precompute two probability densities $\mathbb{P}_{\mathrm{direct}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ and $\mathbb{P}_{\mathrm{indirect}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$ for importance sampling the direct and the indirect boundary paths, respectively.

5.2.5 Computing Change Rates of Discontinuity Boundaries

A key term in Eq. (5.2) is the "normal velocity" $v_{\perp}(\boldsymbol{p}_K)$ that captures the scalar change rate of the discontinuity boundary along the normal direction (with respect to the scene parameter θ).

In practice, evaluating this term using Eq. (4.8) largely amounts to computing the derivative $\frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\theta}$ that, in turn, requires parameterizing the corresponding discontinuity curve or surface near $\boldsymbol{p}_K$. Under our multi-directional formulation described in §5.2.2, $\boldsymbol{p}_K$ is renamed as $\boldsymbol{p}_0^{\mathrm{D}}$. In what follows, we discuss the computation of $v_{\perp}(\boldsymbol{p}_0^{\mathrm{D}})$ at some user-specified $\theta = \theta_0$ (with

the reference configurations selected as described in §3.3.2) while treating $\boldsymbol{p}_0^{\mathrm{S}}$ as fixed.

To obtain $\boldsymbol{p}_0^{\mathrm{D}}$ and, more importantly, its derivative $\frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}$, we first compute $\boldsymbol{x}_0^{\mathrm{D}} \in \mathcal{V}(\theta_0)$ in a differentiable fashion and then transform it back to the reference surface or volume. Without loss of generality, assume that

$$\boldsymbol{x}^{\mathrm{B}} = \xi_1 \, \boldsymbol{x}_{\mathrm{P}} + (1 - \xi_1) \, \boldsymbol{x}_{\mathrm{Q}}, \qquad \boldsymbol{\omega}_{\mathrm{B}} = \overrightarrow{\boldsymbol{x}_0^{\mathrm{S}} \, \boldsymbol{x}^{\mathrm{B}}} = \frac{\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}}}{\|\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}}\|}, \tag{5.25}$$

where $\boldsymbol{x}_{\mathrm{P}}, \boldsymbol{x}_{\mathrm{Q}} \in \mathcal{M}(\theta_0)$ are positions of the two adjacent mesh vertices satisfying that the face edge $\overline{\boldsymbol{x}_{\mathrm{P}} \, \boldsymbol{x}_{\mathrm{Q}}}$ contains $\boldsymbol{x}^{\mathrm{B}}$, and $\xi_1 \in [0, 1)$ is some real number that is considered independent of θ. In Eq. (5.25), $\boldsymbol{x}_{\mathrm{P}}$, $\boldsymbol{x}_{\mathrm{Q}}$, and $\boldsymbol{x}_0^{\mathrm{S}}$ can all be expressed as automatic-differentiation-enabled vectors (as discussed in §3.3.2). We now discuss how $\boldsymbol{x}_0^{\mathrm{D}}$ and $\boldsymbol{p}_0^{\mathrm{D}}$—both of which depend on the scene parameter θ in general—can be computed in a differentiable fashion given $\boldsymbol{x}^{\mathrm{B}}$ and $\boldsymbol{\omega}_{\mathrm{B}}$. After obtaining the derivative $\frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}$, we can compute the scalar change rate $v_\perp(\boldsymbol{p}_0^{\mathrm{D}})$ using Eq. (4.8).

Surface case. When $\boldsymbol{x}_0^{\mathrm{D}}$ is a surface vertex, as illustrated in Figure 5.2-b1, $\boldsymbol{x}_0^{\mathrm{D}}$ and its derivative $\frac{\mathrm{d}\boldsymbol{x}_0^{\mathrm{D}}}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}$ can be computed via differentiable evaluation of the ray-casting function $\boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}})$:

$$\boldsymbol{x}_0^{\mathrm{D}} = \boldsymbol{x}_{\mathcal{M}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}}) = \boldsymbol{x}^{\mathrm{B}} + t_{\mathcal{M}}(\boldsymbol{x}^{\mathrm{B}}, \boldsymbol{\omega}_{\mathrm{B}}) \, \boldsymbol{\omega}_{\mathrm{B}}. \tag{5.26}$$

Then, we obtain $\boldsymbol{p}_0^{\mathrm{D}}$ by transforming $\boldsymbol{x}_0^{\mathrm{D}}$ back to the reference surface as follows. Assume that $\boldsymbol{x}_0^{\mathrm{D}}$ lies within a mesh triangle with vertices $\boldsymbol{x}_{\mathrm{A}}, \boldsymbol{x}_{\mathrm{B}}, \boldsymbol{x}_{\mathrm{C}}$. We compute the barycentric coordinate (u_1, u_2) of $\boldsymbol{x}_0^{\mathrm{D}}$ satisfying that

$$\boldsymbol{x}_0^{\mathrm{D}} = (1 - u_1 - u_2) \, \boldsymbol{x}_{\mathrm{A}} + u_1 \, \boldsymbol{x}_{\mathrm{B}} + u_2 \, \boldsymbol{x}_{\mathrm{C}}. \tag{5.27}$$

We note that, since $\boldsymbol{x}_0^{\mathrm{D}}$ is obtained with differentiable evaluation of Eq. (5.26) (as opposed to the material-form parameterization), both u_1 and u_2 generally depend on the parameter

θ.

To map $\boldsymbol{x}_0^{\mathrm{D}}$ back to the reference surface, given Eq. (5.27), it follows that

$$\boldsymbol{p}_0^{\mathrm{D}} = (1 - u_1 - u_2)\,\boldsymbol{p}_{\mathrm{A}} + u_1\,\boldsymbol{p}_{\mathrm{B}} + u_2\,\boldsymbol{p}_{\mathrm{C}}, \tag{5.28}$$

where $\boldsymbol{p}_* = \mathtt{P}(\boldsymbol{x}_*, \theta_0) = \mathsf{detach}(\boldsymbol{x}_*)$ for each $* \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}\}$.

Given Eqs. (5.25)–(5.28), we essentially parameterize the discontinuity curve locally near $\boldsymbol{p}_0^{\mathrm{D}}$ using ξ_1. Lastly, the derivative of $\boldsymbol{p}_0^{\mathrm{D}}$ is given by

$$\left.\frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\theta}\right|_{\theta=\theta_0} = -(\dot{u}_1 + \dot{u}_2)\,\boldsymbol{p}_{\mathrm{A}} + \dot{u}_1\,\boldsymbol{p}_{\mathrm{B}} + \dot{u}_2\,\boldsymbol{p}_{\mathrm{C}}, \tag{5.29}$$

where $\dot{u}_j := \left.\frac{\mathrm{d}u_j}{\mathrm{d}\theta}\right|_{\theta=\theta_0}$ for $j = 1, 2$.

Volume case. When $\boldsymbol{x}_0^{\mathrm{D}}$ is a volume vertex, as illustrated in Figure 5.2-b2, it must lie on the discontinuity plane determined by $\boldsymbol{x}_0^{\mathrm{S}}$ and the face edge $\overline{\boldsymbol{x}_{\mathrm{P}}\,\boldsymbol{x}_{\mathrm{Q}}}$ containing $\boldsymbol{x}^{\mathrm{B}}$. Assume that

$$\boldsymbol{x}_0^{\mathrm{D}} = \boldsymbol{x}_0^{\mathrm{S}} + \xi_2\,(\boldsymbol{x}^{\mathrm{B}} - \boldsymbol{x}_0^{\mathrm{S}}), \tag{5.30}$$

for some $\xi_2 \geq 1$. Then, the discontinuity plane containing $\boldsymbol{x}_0^{\mathrm{D}}$ is effectively parameterized with ξ_1 and ξ_2 via Eqs. (5.25) and (5.30).

When the motion $\mathtt{X}$ describes some affine transformation, as discussed in §3.3.1, we have

$$\boldsymbol{p}_0^{\mathrm{D}} = \boldsymbol{R}^{-1}\left(\boldsymbol{x}_0^{\mathrm{D}} - \boldsymbol{t}\right). \tag{5.31}$$

When a tetrahedral mesh is used to express the motion $\mathtt{X}$, as described in §3.3.2, assume that $\boldsymbol{x}_0^{\mathrm{D}}$ is located inside a tetrahedron with vertices $\boldsymbol{x}_{\mathrm{A}}$, $\boldsymbol{x}_{\mathrm{B}}$, $\boldsymbol{x}_{\mathrm{C}}$, $\boldsymbol{x}_{\mathrm{D}} \in \mathcal{V}(\theta_0)$. Similar to the surface case, we compute the barycentric coordinates (u_1, u_2, u_3) of $\boldsymbol{x}_0^{\mathrm{D}}$ in a differentiable

fashion (so that they all depend on θ in general). Then, it holds that

$$\boldsymbol{p}_0^{\mathrm{D}} = (1 - u_1 - u_2 - u_3)\,\boldsymbol{p}_{\mathrm{A}} + u_1\,\boldsymbol{p}_{\mathrm{B}} + u_2\,\boldsymbol{p}_{\mathrm{C}} + u_3\,\boldsymbol{p}_{\mathrm{D}}, \tag{5.32}$$

$$\left.\frac{\mathrm{d}\boldsymbol{p}_0^{\mathrm{D}}}{\mathrm{d}\theta}\right|_{\theta=\theta_0} = -(\dot{u}_1 + \dot{u}_2 + \dot{u}_3)\,\boldsymbol{p}_{\mathrm{A}} + \dot{u}_1\,\boldsymbol{p}_{\mathrm{B}} + \dot{u}_2\,\boldsymbol{p}_{\mathrm{C}} + \dot{u}_3\,\boldsymbol{p}_{\mathrm{D}}, \tag{5.33}$$

where $\boldsymbol{p}_* = \mathsf{detach}(\boldsymbol{x}_*)$ for all $* \in \{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$, and $\dot{u}_j := \left.\frac{\mathrm{d}u_j}{\mathrm{d}\theta}\right|_{\theta=\theta_0}$ for $j = 1, 2, 3$.

5.3 Results

Based on our path-space differentiable rendering algorithms described in §5.1 and §5.2, we implement the following two Monte Carlo estimators (in C++ on the CPU):

I.1 Our unidirectional estimator uses unidirectional path tracing (PT) for constructing not only material light paths $\bar{\boldsymbol{p}}$ for the interior term but also the source and detector subpaths $\bar{\boldsymbol{p}}^{\mathrm{S}}$ and $\bar{\boldsymbol{p}}^{\mathrm{D}}$ for the boundary term.

I.2 Our bidirectional estimator uses bidirectional path tracing (BDPT) for construct all these paths.

5.3.1 Validations

We validate our unidirectional and bidirectional estimators (**I.1**, **I.2**) in Figure 5.4 by comparing derivatives estimated with our method to those obtained using finite differences (FD). The following virtual scenes are used for the validation:

- The BRANCHES scene contains a tree-like object with fine structures (that is outside the field of view) casting soft shadows on the ground. This object is further embedded

within an optically thin heterogeneous medium.

- The BUST scene consists of a translucent bust with complex geometry and spatially varying scattering properties. The bust is optically thick and, thus, exhibits strong multiple scattering.

- The BUNNY scene contains a diffuse bunny inside a Cornell box. The scene is lit by two area light sources with the right one facing the ceiling.

- The VEACH-EGG scene is modeled after the well known scene created by Veach [81] for demonstracing the effectiveness of BDPT algorithms. This scene involves a large floor lamp, a small spot light, and a glass egg on a table, and we use a camera setting to focus on the egg.

- The BUMPY-SPHERE scene consists of a sphere made of rough glass inside a box filled with a homogeneous participating medium. The sphere is lit by a point light from above, yielding strong volumetric shadow and caustic effects.

We use perspective pinhole cameras (discussed in §4.2) for all these scenes.

For the BRANCHES scene, we compute derivatives with respect to the rotation angle of the object around the vertical axis. For the BUST scene, we differentiate the ordinary image with respect to the rotation angle of the translucent bust. Derivative images of these two examples generated using our unidirectional estimator (**I.1**) closely matches the reference obtained using finite differences.

For the BUNNY scene, the differentiation is with respect to the horizontal displayment of both area lights. For the VEACH-EGG scene, the derivatives are computed with respect to the vertical location of an outside-of-view spot light on the right. Both examples involve light transport effects that are challenging for unidirectional methods. Thus, we use our

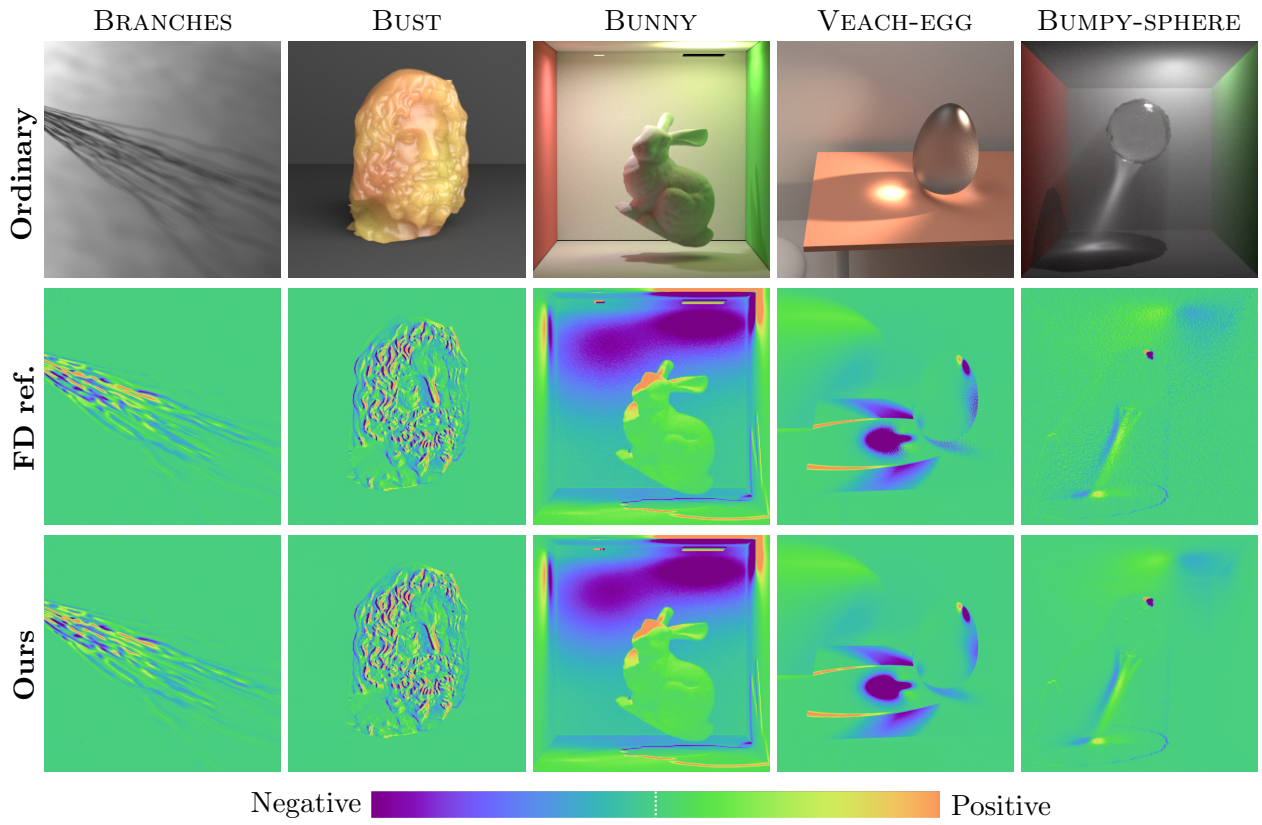


Figure 5.4: **Differentiable-rendering validations:** Comparison between the derivatives estimated using our path-space algorithms (**Ours**) and the references (**Ref.**) obtained using finite-differences method. The derivative images are displayed using the same colormap with different scales.

bidirectional estimator (**I.2**) for this scene. Again, our results and the references match closely.

Lastly, for the BUMPY-SPHERE scene, we estimate derivatives with respect to the horizontal translation of the light source. Our result obtained using the bidirectional estimator (**I.2**) matches the one obtained using finite difference method, which still contains some Monte Carlo noises even after being rendered for many hours.

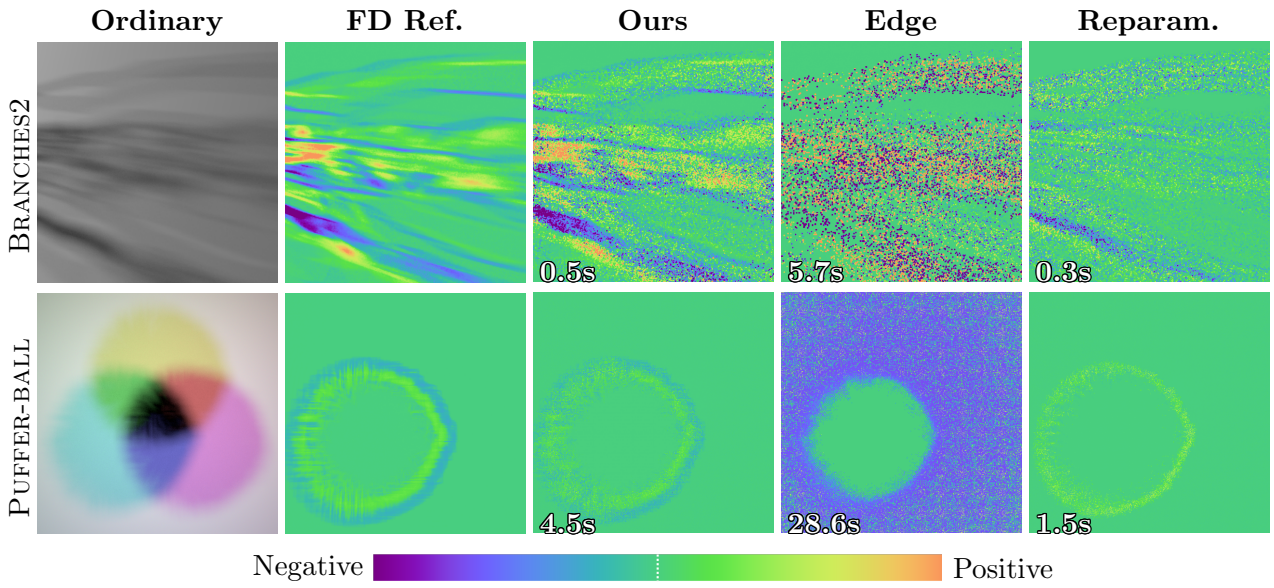


Figure 5.5: **Evaluation** of the effectiveness of our unidirectional estimator (**I.1**) for surface-only light transport with complex geometry. All derivative images (other than the finite-difference reference) are generated under *equal sample* with the exact running time shown on the bottom left corner of individual images. Our method runs much faster than edge sampling and produces a more accurate result than the reparamterization method.

5.3.2 Comparisons

Thanks to our differential path integral formulation introduced in Chapter 4, our Monte Carlo estimators (**I.1**, **I.2**) are capable of handling complex geometric discontinuities and light transport effects with high efficiency. In what follows, we evaluate the effectiveness of our method on both aspects. To this end, we compare our results to those generated using the (unbiased) edge-sampling methods [45, 95] and the (biased) reparameterization approach [49]. We use two kinds of configurations for these comparisons: (i) scenes with complex geometry and conclusion; and (ii) those with light transport effects that are known to make unidirectional methods inefficient (e.g., caustics).

Complex geometry. Previously, sampling points from *silhouette edges* of a surface point (i.e., *edge sampling*) was generally required to obtain unbiased derivative estimates [45, 95]

with respect to the scene geometry. This process, however, can be prohibitively expensive for scenes with complex geometries. Another solution is to trade unbiasedness for computational efficiency by applying a local reparameterization [49]. This method relies on a number of simplifying assumptions that can be violated in scenes with complex motions, making the resulting derivatives too biased for inverse rendering applications.

In Figure 5.5, we show *equal-sample*[3] comparisons of derivative images computed by our unidiretional algorithm and the other two baselines. The examples in this figure contain no volumetric light transport; and thus can be rendered using both baselines. The details of the two examples are as follows:

- The BRANCHES2 scene is a simplified version of *branches* scene by removing the embedding heterogeneous medium. Same as the *branches* scene, we differentiate with respect to the rotation angle of the object around the vertical axis.
- The PUFFER-BALL involves a highly-detailed mesh generated via physics-based simulation [98]. This model contains over one million faces and is illuminated by three emitters of red, green, and blue colors, creating the colored shadows on the ground. For each light, we use a single parameter to control its size and intensity such that the total power remians constant. The derivative images are computed with respect to the paprameter controlling the red light (which casts a blue shadow).

For both scenes, our results closely matches the references generated using the finite-difference (FD) method. Edge sampling, despite being unbiased, struggled to produce clean results. Compared to edge sampling, our method is both *faster* and provides derivative estimates with *much lower noise*. The reparameterization method, on the other hand, generates clean results but with high bias.

[3]We use CPU-based implementations of both our algorithms and the edge-sampling ones [45, 95]. The reparameterization method [49], on the other hand, replies on a GPU-based implementation. Due to this architectural difference, we opt for equal-sample instead of equal-time comparisons, as the former are more representative of different methods' relative performance.

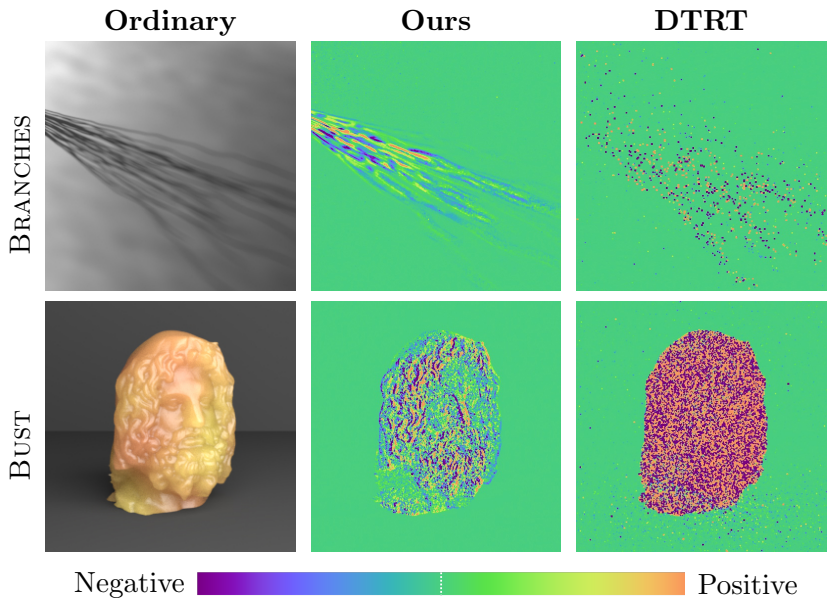


Figure 5.6: **Evaluation** of the effectiveness of our unidirectional estimator (**I.1**) with complex geometry and volumetric light transport. All derivative images are generated under *equal time*. Our method runs much faster than DTRT [95] and produces much cleaner derivative images.

In Figure 5.6, we reuse the Branches and Bust scene (see Figure 5.4), along with their derivative configurations, to evaluate the efficiency of our unidirectional estimator (**I.1**). We compare the derivative images estimated using our method (with low sample counts) to those obtained by DTRT [95], which is the only framework that supports geometric differentiation of volumetric light transport other than our generalized path-space formulation. As both DTRT and our methods are developed on CPU, the comparisons are conducted under equal-time condition. Because of the complex visibility, DTRT–which relies on explicit detection of object silhouettes–produces highly noisy derivative estimates for both scenes. Our method, on the other hand, does not require silhouette detection and can generate much cleaner reults in equal time.

Complex light transport effect. Another major benefit of our theory is to allow the *interior* term (and subpath contributions in the *boundary* term) to be estimated using sophisticated methods such as bidirectional path tracing (BDPT). We use the scene Veach-egg2

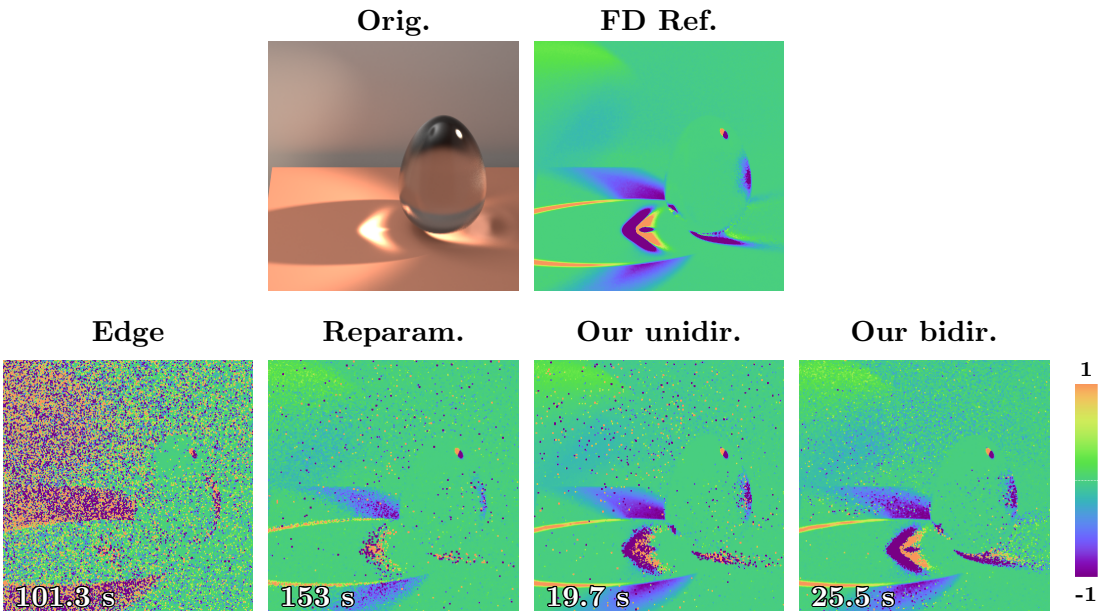


Figure 5.7: **Evaluation** of the effectiveness of our unidirectional and bidirectional estimators (**I.1**, **I.2**) using the VEACH-EGG2 scene. All derivative images (other than the reference) are generated under *equal sample*.

to evaluate the performance of our bidirectional estimator (**I.2**). This scene remains largely identical to the one used for validation in Figure 5.4, except for using a lower roughness for the glass egg.

In Figure 5.7, we show derivatives with respect to the vertical displacement of the spot light estimated using our unidirectional and bidirectional estimators (**I.1**, **I.2**), edge sampling, and biased reparameterization. All results (other than the finite-difference reference) are generated under *equal sample* per pixel. Our bidirectional algorithm outperforms the others significantly by producing accurate and clean derivatives estimates in the caustics area.

We will further demonstrate the practical advantages of our path-space differentiable rendering algorithms in Chapter 8 with more physics-based inverse rendering examples.

Chapter 6

Efficient Estimation of Interior Integrals Using Antithetic Sampling

The *interior* component (5.1) of our differential path integral (4.14), as discussed in §5.1, can be estimated by repurposing forward-rendering path sampling techniques. Although this works adequately in many cases, it can lead to poor performance for scenes containing, for example, glossy materials.

For efficient and robust estimation of the *interior* integral (5.1), we introduce new Monte Carlo techniques that leverage *antithetic sampling* in this chapter.

6.1 Antithetic Sampling Preliminaries

Being a classic variance reduction framework for Monte Carlo estimation, antithetic sampling [25, 19] has been studied in probabilistic inference and machine learning [69, 90]. In computer graphics, this technique has been explored by several previous works in forward rendering [77, 63, 76, 75]. In Monte Carlo differentiable rendering, Bangaru et al. [4] have

applied antithetic sampling to efficiently handle discontinuity boundaries under the warped-area formulation.

1D example. The core idea of antithetic sampling is to forgo independent samples and use (negatively) correlated ones instead. To demonstrate how antithetic sampling works, we consider the problem of estimating the following 1D integral:

$$I := \int_{-\infty}^{\infty} h(x)\,\mathrm{d}x, \tag{6.1}$$

where the integrand h is approximately an odd function with $h(x) \approx -h(-x)$. When h contains high-magnitude positive and negative regions, estimating I using ordinary Monte Carlo with independent samples can suffer from very slow convergence.

To address this problem, one can draw x from some predetermined probability density p and then set $x^* := -x$, resulting in an antithetic estimator

$$\langle I \rangle_{\text{antithetic}} := \frac{h(x) + h(x^*)}{p(x) + p(x^*)}. \tag{6.2}$$

Since $h(x) + h(x^*) \approx 0$, $\langle I \rangle_{\text{antithetic}}$ can offer significantly lower variance. Please see Appendix 6.A for a detailed analysis of this antithetic estimator (in a more general setting).

In the following, we introduce new Monte Carlo techniques that leverage antithetic sampling to efficiently differentiate BSDFs (§6.2) and pixel reconstruction filters (§6.3). Additionally, we show how these techniques can be applied to full light transport paths (§6.4). Lastly, we validate and demonstrate the effectiveness of our antithetic sampling techniques (§6.5).

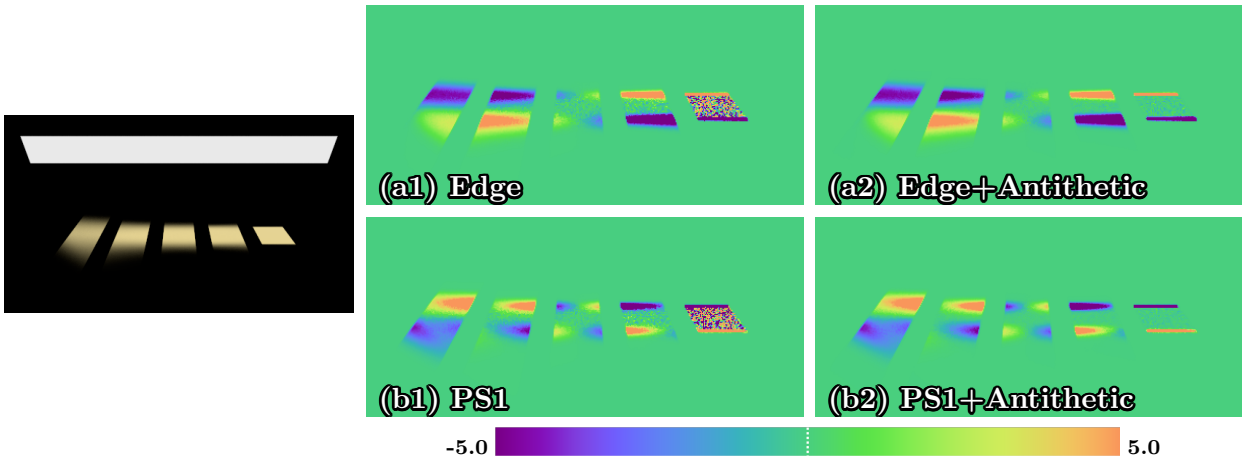


Figure 6.1: **BSDF antithetic sampling:** This example contains a simple scene where a row of reflectors—whose roughnesses decrease from left to right—are lit by a large area light. When estimating derivatives (with respect to the rotation angle around the horizontal axis), the computational efficiency of conventional sampling methods declines when the surface roughnesses decrease, as shown in (a1) and (b1). We use "Edge" and "PS1" to indicate, respectively, differentiable path tracing with edge sampling [45] and the unidirectional path-space method. Coupled with the same base methods, our BSDF antithetic sampling offers significant variance reduction for the *interior* term in equal time, as shown in (a2) and (b2).

6.2 Antithetic Sampling of BSDFs

BSDF sampling has been a key ingredient for constructing light paths in forward rendering. Although these techniques can be repurposed for differentiable rendering, the sampling efficiency can be unsatisfactory for glossy scenes—especially when differentiating with respect to scene geometries. With near-specular reflection and refraction, the estimated derivatives can even have unbounded variance. This issue exists in most, if not all, existing physics-based differentiable rendering frameworks, including edge-sampling methods [45, 95] and our path-space algorithms, as shown in Figure 6.1-(a1, b1).

To understand why this occurs, we examine the *interior* integral of Eq. (5.1) where the integrand of this term is given by the derivative $\frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta}$ of the material path contribution with respect to the scene parameter θ. Let $\boldsymbol{p}_n$ in the material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$ be a

surface vertex with glossy BSDF, we can rewrite $\frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta}$ using product rule:

$$\frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} = \frac{\mathrm{d}\hat{f}_0(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}) + \hat{f}_0(\bar{\boldsymbol{p}}) \left(\frac{\mathrm{d}}{\mathrm{d}\theta} f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n-1}) \right), \tag{6.3}$$

where $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \theta)$ and $\hat{f}_0$ consists of all the other terms in $\hat{f}$ except $f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$. For notational simplicity, we rewrite $f_\mathrm{s}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ as $f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o})$ with $\boldsymbol{\omega}_\mathrm{o} := \overrightarrow{\boldsymbol{x}_n \boldsymbol{x}_{n+1}}$ and $\boldsymbol{\omega}_\mathrm{i} := \overrightarrow{\boldsymbol{x}_n \boldsymbol{x}_{n-1}}$.

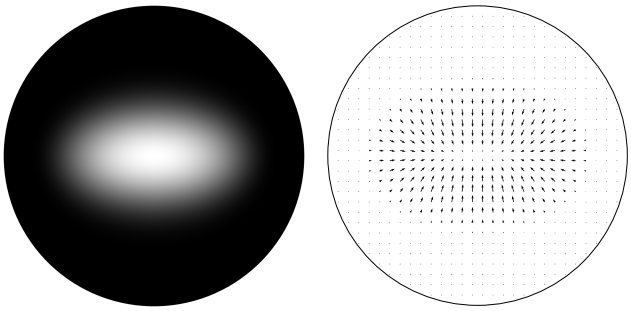
Given the outgoing direction $\boldsymbol{\omega}_\mathrm{o}$, traditional BSDF sampling techniques (developed for forward rendering) typically draw the incident direction $\boldsymbol{\omega}_\mathrm{i}$ with some probability density proportional to $f_\mathrm{s}(\cdot, \boldsymbol{\omega}_\mathrm{o})$. Unfortunately, this can be inefficient when handling the second term on the right-hand side of Eq. (6.3) due to the vast difference between f_s and its derivative $\frac{\mathrm{d}}{\mathrm{d}\theta} f_\mathrm{s}$—especially when the BSDF is glossy.

Our Method

As discussed in §6.1, integrals with integrands that are approximately odd and contain high-magnitude positive and negative regions can lead to slow convergence when estimated using independent samples. In the context of differentiable rendering, such functions emerge as (geometric) derivatives of glossy or near-specular BSDFs. To address this problem, we introduce *BSDF antithetic sampling*, as shown in Figure 6.1-(2a, 2b). We base our derivation on microfacet BSDFs that generally take the form

$$f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}) = D(\boldsymbol{\omega}_\mathrm{h})\, f_\mathrm{s}^{(0)}(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}), \tag{6.4}$$

where D is the **normal distribution function** (NDF) parameterized using the **half-way vector** $\boldsymbol{\omega}_\mathrm{h} := \frac{(\boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o})}{\|\boldsymbol{\omega}_\mathrm{i} + \boldsymbol{\omega}_\mathrm{o}\|}$, and $f_\mathrm{s}^{(0)}$ captures other factors such as Fresnel reflection/transmission and shadowing/masking terms.



(a) **NDF** $D(\boldsymbol{\omega}_\mathrm{h})$ (b) **NDF deriv.** $\mathrm{d}D/\mathrm{d}\boldsymbol{\omega}_\mathrm{h}$

Figure 6.2: Normal distributions $D(\boldsymbol{\omega}_\mathrm{h})$ usually exhibit point symmetry, causing their derivatives $\mathrm{d}D/\mathrm{d}\boldsymbol{\omega}_\mathrm{h}$—which is vector-valued—to possess point symmetry (with respect to the origin). We visualize an anisotropic normal distribution in (a) and its derivative in (b).

Differentiating Eq. (6.4) with respect to some scene parameter θ yields

$$\frac{\mathrm{d}f_\mathrm{s}(\boldsymbol{\omega}_\mathrm{i},\boldsymbol{\omega}_\mathrm{o})}{\mathrm{d}\theta} = \frac{\mathrm{d}D(\boldsymbol{\omega}_\mathrm{h})}{\mathrm{d}\theta}\, f_\mathrm{s}^{(0)}(\boldsymbol{\omega}_\mathrm{i},\boldsymbol{\omega}_\mathrm{o}) + D(\boldsymbol{\omega}_\mathrm{h})\,\frac{\mathrm{d}f_\mathrm{s}^{(0)}(\boldsymbol{\omega}_\mathrm{i},\boldsymbol{\omega}_\mathrm{o})}{\mathrm{d}\theta}, \tag{6.5}$$

where the NDF derivative $\frac{\mathrm{d}D}{\mathrm{d}\theta}$, according to the chain rule, equals

$$\frac{\mathrm{d}D}{\mathrm{d}\theta} = \frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}\frac{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}{\mathrm{d}\theta}, \tag{6.6}$$

where $\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}$ can be obtained by analytically differentiating the NDF (for parametric BSDF models), and the exact form of $\frac{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}{\mathrm{d}\theta}$ depends on the differentiable-rendering formulation.

In Eqs. (6.5) and (6.6), $f_\mathrm{s}^{(0)}$ and $\frac{\mathrm{d}f_\mathrm{s}^{(0)}}{\mathrm{d}\theta}$ usually vary slowly, while the normal distribution function D and its derivative $\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}$ can vary rapidly for glossy materials.

Symmetry in NDF derivatives. Most, if not all, commonly used normal distributions, including the Beckmann and the GGX models, are point symmetric. Specifically, under a local coordinate system with the surface normal aligned with the z-axis, it holds that

$$D([x_\mathrm{h}, y_\mathrm{h}, z_\mathrm{h}]) = D([-x_\mathrm{h}, -y_\mathrm{h}, z_\mathrm{h}]), \tag{6.7}$$

for all $x_\mathrm{h}^2 + y_\mathrm{h}^2 + z_\mathrm{h}^2 = 1$ and $z_\mathrm{h} > 0$. We note that this is the case even for anisotropic normal distributions. This point symmetry (with respect to the origin) causes the derivative $\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}$ to also be point symmetric:

$$\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}([x_\mathrm{h}, y_\mathrm{h}, z_\mathrm{h}]) = -\frac{\mathrm{d}D}{\mathrm{d}\boldsymbol{\omega}_\mathrm{h}}([-x_\mathrm{h}, -y_\mathrm{h}, z_\mathrm{h}]). \tag{6.8}$$

Figure 6.2 visualizes the NDF and its derivative.

BSDF antithetic sampling. Based on the point symmetry of the NDF derivative, we introduce *BSDF antithetic sampling* that exploits such symmetries. As shown in Algorithm 2, the process starts with drawing a half-way vector $\boldsymbol{\omega}_{\mathrm{h},1}$ (Line 3) the same way as in forward rendering based on the NDF [85] or visible NDF [28]. Then, we take the antithetic sample $\boldsymbol{\omega}_{\mathrm{h},2} = [-x_\mathrm{h}, -y_\mathrm{h}, z_\mathrm{h}]$ assuming $\boldsymbol{\omega}_{\mathrm{h},1} = [x_\mathrm{h}, y_\mathrm{h}, z_\mathrm{h}]$ under a local coordinate system where the surface normal is aligned with the z-axis (Line 4).

With the half-way directions $\boldsymbol{\omega}_{\mathrm{h},1}$ and $\boldsymbol{\omega}_{\mathrm{h},2}$ generated, we calculate the corresponding incident directions $\boldsymbol{\omega}_{\mathrm{i},1}$ and $\boldsymbol{\omega}_{\mathrm{i},2}$ (Line 6) as well as the probability densities p_1 and p_2 (Line 7). We note that p_1 and p_2 are computed solely based on the probability density p_h (from which the ordinary sample $\boldsymbol{\omega}_{\mathrm{h},1}$ is drawn). To be precise, when sampling microfacet BRDFs using $p_\mathrm{h} = D$, we have

$$p_j = \frac{D(\boldsymbol{\omega}_{\mathrm{h},j})}{4(\boldsymbol{\omega}_\mathrm{o} \cdot \boldsymbol{\omega}_{\mathrm{h},j})}, \qquad \text{for } j = 1, 2. \tag{6.9}$$

In practice, our BSDF antithetic sampling offers several benefits:

- It can provide significant variance reduction for estimating geometric gradients when the scene is glossy.
- It is very easy to implement (that is, using a few lines of code), as demonstrated in Algorithm 2.

Algorithm 2: Antithetic sampling of microfacet BSDFs

```
AntitheticBSDFSample(x, ω_o)
begin
    Draw ω_h,1 = [x_h, y_h, z_h] ~ p_h;                    // The ordinary sample
    Set ω_h,2 ← [−x_h, −y_h, z_h];                         // The antithetic sample
    for j ∈ {1, 2} do
        Compute incident direction ω_i,j based on ω_o and ω_h,j;
        Compute p_j := p(ω_i,j) based on p_h(ω_h,j);
    end
    return (ω_i,1, p_1, ω_i,2, p_2);
end
```

- Since we draw $\boldsymbol{\omega}_{\mathrm{h},1}$ the same way as in forward rendering, the resulting sampling pattern is well suited for forward rendering.
- Our BSDF antithetic sampling is not limited to microfacet BSDFs: The same algorithm can be applied to any BSDF and provide variance reduction as long as the BSDF derivative is similarly point symmetric.

Differentiable Rendering with BSDF Antithetic Sampling

Our BSDF antithetic sampling can be integrated into the Monte Carlo estimators of the *interior* integral of Eq. (5.1). For instance, it can be used to improve our unidirectional estimator (**I.1**) via a process outlined in Algorithm 3, which adopts unidirectional path tracing to sample material light paths $\bar{\boldsymbol{p}}$. For each path vertex associated with a glossy or near-specular BSDF, the algorithm draws two incident directions, causing the light path to branch at this vertex.

Algorithm 3 focuses on the interfacial (i.e., surface-only) case but can be extended to handle volumetric light transport easily. Further, ray intersections are applied using material points directly since we assume the reference configurations to be chosen as described in §3.3.

Algorithm 3: Estimate interior integrals (5.1) using BSDF antithetic sampling

1 `estimateInterior(`$\bar{\boldsymbol{p}}$, $\mathrm{pdf}_{\bar{\boldsymbol{p}}}$`)`
Input: Material path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots)$ associated with a probability $\mathrm{pdf}_{\bar{\boldsymbol{p}}}$
Output: Estimate $\dot{I}$ of the *interior* integral (5.1) at $\theta = \theta_0$
2 **begin**
3 **if** $\boldsymbol{p}_0$ *is located on a light source* **then**
4 $\dot{I} \leftarrow$ `evalMatContribDeriv(`$\bar{\boldsymbol{p}}$`)`$/\mathrm{pdf}_{\bar{\boldsymbol{p}}}$; `// Evaluate` $\left(\frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta}\Big|_{\theta=\theta_0}\right)/\mathrm{pdf}_{\bar{\boldsymbol{p}}}$
5 **else**
6 $\dot{I} \leftarrow 0$;
7 **end**
8 **if** $\boldsymbol{p}_0$ *lies on a glossy surface* **then** `// Antithetic sampling`
9 $(\boldsymbol{\omega}_{\mathrm{i},1}, \mathrm{pdf}_1^\sigma, \boldsymbol{\omega}_{\mathrm{i},2}, \mathrm{pdf}_2^\sigma) \leftarrow$ `AntitheticBSDFSample(`$\boldsymbol{p}_0, \overrightarrow{\boldsymbol{p}_0\boldsymbol{p}_1}$`)`;
10 $\boldsymbol{q}_1 \leftarrow \boldsymbol{x}_{\mathcal{M}}(\boldsymbol{p}_0, \boldsymbol{\omega}_{\mathrm{i},1})$; $\boldsymbol{q}_2 \leftarrow \boldsymbol{x}_{\mathcal{M}}(\boldsymbol{p}_0, \boldsymbol{\omega}_{\mathrm{i},2})$; `// Ray intersection`
11 Convert pdf_1^σ, pdf_2^σ under the solid-angle measure to pdf_1, pdf_2 under the area measure;
`/* Let` $\boldsymbol{q} \oplus \bar{\boldsymbol{p}} := (\boldsymbol{q}, \boldsymbol{p}_0, \boldsymbol{p}_1, \ldots)$ `for any` $\boldsymbol{q} \in \mathcal{B}_{\mathcal{M}}$ `*/`
12 $\dot{I} \leftarrow \dot{I} +$ `estimateInterior(`$\boldsymbol{q}_1 \oplus \bar{\boldsymbol{p}}, \mathrm{pdf}_{\bar{\boldsymbol{p}}} \cdot (\mathrm{pdf}_1 + \mathrm{pdf}_2)$`)`;
13 $\dot{I} \leftarrow \dot{I} +$ `estimateInterior(`$\boldsymbol{q}_2 \oplus \bar{\boldsymbol{p}}, \mathrm{pdf}_{\bar{\boldsymbol{p}}} \cdot (\mathrm{pdf}_1 + \mathrm{pdf}_2)$`)`;
14 **else** `// Standard sampling`
15 $(\boldsymbol{\omega}_{\mathrm{i}}, \mathrm{pdf}^\sigma) \leftarrow$ `BSDFSample(`$\boldsymbol{p}_0, \overrightarrow{\boldsymbol{p}_0\boldsymbol{p}_1}$`)`;
16 $\boldsymbol{q} \leftarrow \boldsymbol{x}_{\mathcal{M}}(\boldsymbol{p}_0, \boldsymbol{\omega}_{\mathrm{i}})$; ; `// Ray intersection`
17 Convert pdf^σ under the solid-angle measure to pdf under the area measure;
18 $\dot{I} \leftarrow \dot{I} +$ `estimateInterior(`$\boldsymbol{q} \oplus \bar{\boldsymbol{p}}, \mathrm{pdf}_{\bar{\boldsymbol{p}}} \cdot \mathrm{pdf}$`)`;
19 **end**
20 **return** $\dot{I}$;
21 **end**

Next-event estimation. The standard-sampling branch (Lines 15–18) of Algorithm 3 can be easily extended to utilize next-event estimation (NEE). Although this is also possible for our antithetic sampling (by generating $\boldsymbol{\omega}_{\mathrm{i},1}$ based on a position sample on a light source and $\boldsymbol{\omega}_{\mathrm{i},2}$ following the same symmetry), we find it unnecessary in practice since antithetic sampling is only applied when the surface is sufficiently glossy.

Varying parameterizations. Even though our derivation so far is limited to the path-space parameterization, our antithetic BSDF sampling technique is actually largely independent of any specific parameterization; and can also be applied to the edge sampling

methods [45, 95]. To be precise, the different parameterizations adopted in the two frameworks would only affect (i) how gradients of individual variables (such as $\boldsymbol{p}$ and $\boldsymbol{\omega}_{\mathrm{i}}$) are calculated; and (ii) how the *boundary* integral is handled.

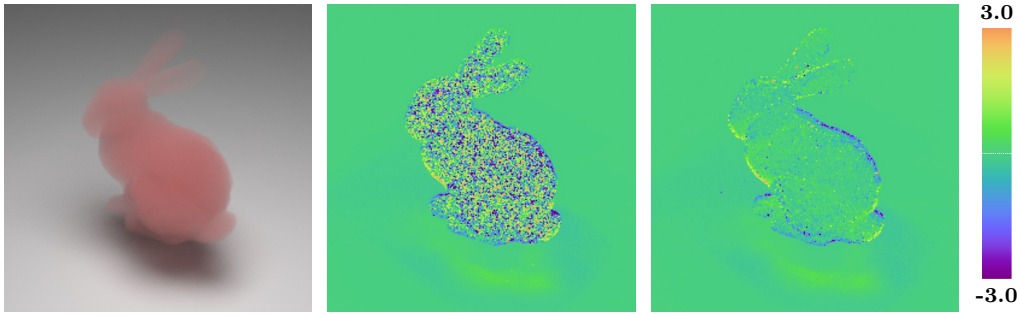
Correlating subpaths. By utilizing pairs of correlated samples, our BSDF antithetic sampling makes a light transport path to branch into two (Line 9 of Algorithm 3) that start with $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$, respectively. To ensure that the contributions of these two subpaths mostly cancel out when computing the gradient of L, we use correlated random samples to generate them. Conceptually, this is similar to computing finite differences using Monte Carlo methods.

Path branching. When only a small fraction of the scene is (highly) glossy, branching the light path at each vertex where BSDF antithetic sampling is performed has little impact on rendering performance. On the other hand, for mostly glossy scenes, frequent antithetic sampling can yield exponential branching of light paths and lowered performance. We will introduce a solution to this problem in §6.4.

6.3 Antithetic Sampling of Pixel Filters

Similar to BSDFs, pixel reconstruction filters can also lead to noisy estimates due to high-magnitude positive and negative regions in their derivatives (as demonstrated in Figure 6.3-b1). To address this problem, we introduce in the following *antithetic sampling of pixel reconstruction filters* (as demonstrated in Figure 6.3-b2).

When devising our antithetic sampling technique, we focus on the perspective pinhole camera model described in §4.2 with the *tent* reconstruction filter (to be defined soon). The results, on the other hand, can be easily generalized to other detector models (such as orthographic



(a) (b1) (b2)

Figure 6.3: **Antithetic sampling of pixel filter:** Sampling primary camera rays based on pixel reconstruction filters can produce high variance when estimating the *interior* integral (b1). With our pixel-filter antithetic sampling, significant variance reduction can be achieved (b2). In this example, the derivatives in (b1) and (b2) are computed with respect to the bunny's vertical position, and the ordinary image is shown in (a).

cameras) and pixel reconstruction filters (such as Gaussian).

Tent reconstruction filter. Assuming each pixel to be a square in the image plane $\mathcal{I}$ with edge length d (and surface area d^2), the tent reconstruction filter is defined as[1]

$$\mathcal{P}_{\text{tent}}(\boldsymbol{y}^\perp) := \frac{1}{d^2} \max\left(1 - \frac{|y_{\text{s}}^\perp|}{d}, 0\right) \max\left(1 - \frac{|y_{\text{t}}^\perp|}{d}, 0\right), \tag{6.10}$$

where $y_{\text{s}}^\perp := (\boldsymbol{y}^\perp - \boldsymbol{c}) \cdot \boldsymbol{s}$ and $y_{\text{t}}^\perp := (\boldsymbol{y}^\perp - \boldsymbol{c}) \cdot \boldsymbol{t}$ with: $\boldsymbol{c} \in \mathcal{I}$ denotes the center point of the pixel; $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{S}^2$ indicate the horizontal and vertical axes of the image plane, respectively.

Recall that, as discussed in section §4.2, we formulate perspective pinhole cameras by encoding their responses using the sensor importance via Eq. (4.19). Under this formulation, for any material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{q})$, the material-form path integral (3.9) can be rewritten as

$$I = \int_{\hat{\boldsymbol{\Omega}}} \underbrace{\hat{f}_0(\bar{\boldsymbol{p}})\, W_{\text{e}}^{\text{pinhole}}(\boldsymbol{y})\, J(\boldsymbol{q})}_{=\hat{f}(\bar{\boldsymbol{p}})} \, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{6.11}$$

[1] We intentionally let the supports of the tent filter from neighboring pixels to overlap, allowing the sum $\sum_i \mathcal{P}_{\text{tent}}^{(i)}$ over all pixels to be constant across the image plane.

where $\boldsymbol{y} = \mathtt{X}(\boldsymbol{q}, \theta)$; $J(\boldsymbol{q})$ is defined in Eq. (3.11); and $\hat{f}_0(\bar{\boldsymbol{p}})$ is a factor of the material measurement contribution $\hat{f}(\bar{\boldsymbol{p}})$. We denote the last vertex of the material path $\bar{\boldsymbol{p}}$ as $\boldsymbol{q}$ (instead of $\boldsymbol{p}_N$) to emphasize that it is directly "connected" to the pinhole camera at $\boldsymbol{x}_{\text{cam}}$.

When differentiating radiometric measurements of a perspective pinhole camera given by Eq. (6.11) with respect to some scene parameter θ, the *interior* component of the resulting derivative takes the form of

$$\int_{\hat{\boldsymbol{\Omega}}} \frac{\mathrm{d}}{\mathrm{d}\theta} \left(\hat{f}_0(\bar{\boldsymbol{p}})\, W_{\mathrm{e}}^{\text{pinhole}}(\boldsymbol{y})\, J(\boldsymbol{q}) \right) \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{6.12}$$

When the scene parameter θ affects the geometry of an object that is directly visible to the camera, for any spatial point $\boldsymbol{y}$ associated with the object, $\mathcal{P}(\boldsymbol{y}^\perp)$ depends on θ—even if the filter $\mathcal{P}$ itself is constant. This is because, under the material-form parameterization described in §3.2, we have $\boldsymbol{y} = \mathtt{X}(\boldsymbol{q}, \theta)$ for some fixed material point $\boldsymbol{q}$.

Attached sampling. Eq. (6.11) can be reparameterized by applying attached sampling [92] to the pixel reconstruction filter $\mathcal{P}$, allowing this term to be canceled out entirely (if perfect importance sampling is possible) before differentiation. Doing so, however, suffers from several problems which we will discuss in Appendix 6.B.

Our Method

In practice, the pixel reconstruction filter $\mathcal{P}$, as a function of image-plane positions $\boldsymbol{y}^\perp$, are usually point-symmetric with respect to the pixel center. This causes the (vector-valued) spatial derivative $\frac{\mathrm{d}\mathcal{P}(\boldsymbol{y}^\perp)}{\mathrm{d}\boldsymbol{y}^\perp}$ to exhibit the same type of symmetry, allowing us to apply antithetic sampling to reduce the variance introduced by this derivative (see Figure 6.3-b2).

Specifically, the basic idea is to generate pairs of material paths $\bar{\boldsymbol{p}} = (\boldsymbol{p}_{1,0}, \boldsymbol{p}_{1,1}, \ldots, \boldsymbol{q})$ and

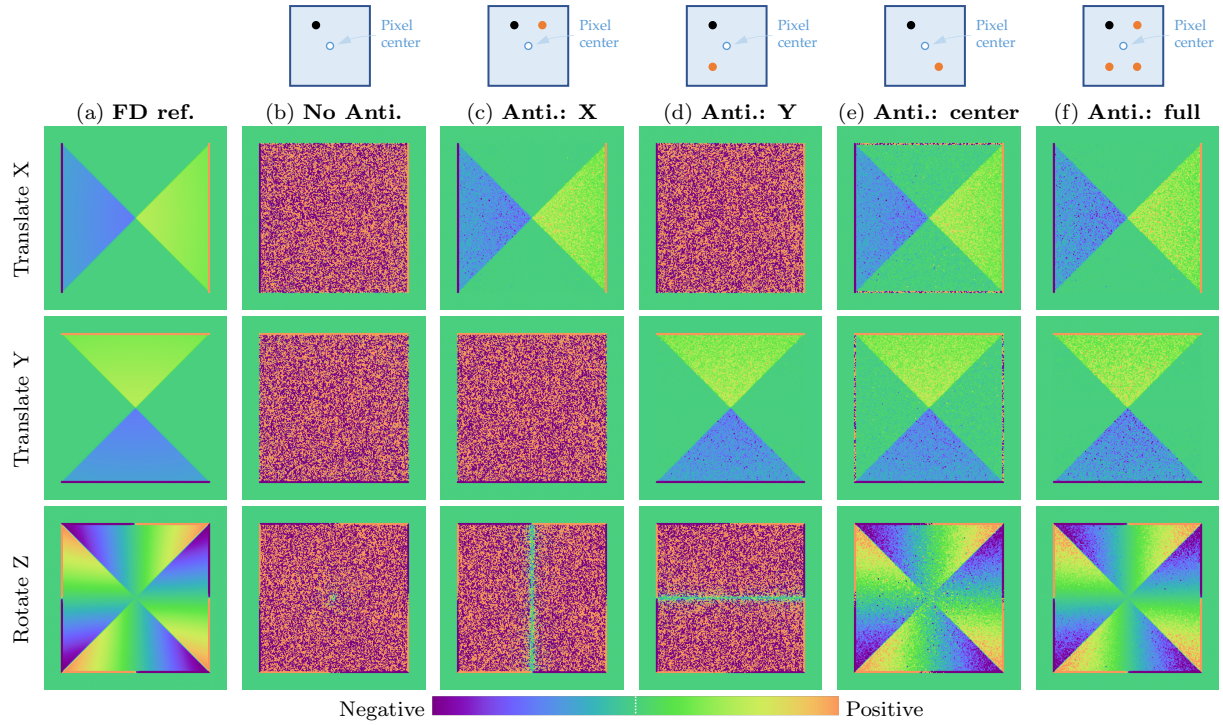


Figure 6.4: **Antithetic sampling pattern**: We evaluate the effectiveness of four different antithetic sampling patterns (c–f) using derivative images of a pyramid-like object viewed from the top using the tent pixel reconstruction filter (6.10). The derivatives are estimated with respect to translations along the x- and y-axes (that are within the image plane), and rotation about the Z axis (that is perpendicular to the image plane), respectively. All derivative images in (b–f) are generated in equal time.

$\bar{\boldsymbol{p}}^* = (\boldsymbol{p}_{2,0}, \boldsymbol{p}_{2,1}, \ldots, \boldsymbol{q}^*)$ such that the image-plane projections of $\boldsymbol{y} = \mathtt{X}(\boldsymbol{q}, \theta)$ and $\boldsymbol{y}^* = \mathtt{X}(\boldsymbol{q}^*, \theta)$ are point-symmetric. This can be achieved by (i) sampling image-plane location $\boldsymbol{y}_1^\perp$ based on the reconstruction filter $\mathcal{P}$ and setting $\boldsymbol{y}_2^\perp$ as the point reflection of $\boldsymbol{y}_1^\perp$; (ii) tracing two camera rays through $\boldsymbol{y}_1^\perp$ and $\boldsymbol{y}_2^\perp$ to obtain spatial points $\boldsymbol{y}$ and $\boldsymbol{y}^*$, respectively (which in turn give the corresponding material points $\boldsymbol{q}$ and $\boldsymbol{q}^*$); and (iii) constructing the remaining vertices of paths $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}*$ using standard methods like unidirectional path tracing (with correlated random samples).

Antithetic sampling pattern. The aforementioned process couples an ordinary path with one antithesis (by exploiting point symmetry). Alternatively, we can construct deterministically three antithetic paths using both edge and point symmetry as shown in Fig-

ure 6.4-f. In theory, the effectiveness of different pixel-filter antithetic sampling patterns depends on the target derivative. In practice, we found that the four-point pattern usually offers the best performance at equal time (see Figure 6.4). We will also use this four-point pattern in our path-level extension, which we will present in §6.4.2.

Limitations. Although the pixel-filter antithetic sampling by generating ordinary and antithetic paths using correlated samples works adequately in many cases, it suffers from several major problems. First, it introduces relatively high computational overhead since two (or even four) full paths $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}^*$ have to be constructed. Secondly, the process does not handle an important path-sampling scheme where next-event estimation is applied at $\boldsymbol{q}$ (by tracing a shadow ray toward the camera at $\boldsymbol{x}_{\text{cam}}$). This scheme is needed by, for example, adjoint particle tracing (APT) as well as bidirectional path tracing (BDPT). To address these problems, we introduce a path-level extension in §6.4.2.

6.4 Path-Level Antithetic Sampling

We now show how our BSDF and pixel-filter antithetic sampling (presented in §6.2 and §6.3, respectively) can be generalized to full light transport paths, allowing more efficient and robust estimations of the *interior* integral (5.1).

6.4.1 BSDF Antithetic Sampling At the Path Level

Based on the differential path integral formulation of Eq. (4.14), we introduce a new BSDF antithetic sampling technique that operates at the path level and enjoys (i) having no exponential branching even for mostly glossy scenes; and (ii) supporting both unidirectional and bidirectional path sampling methods.

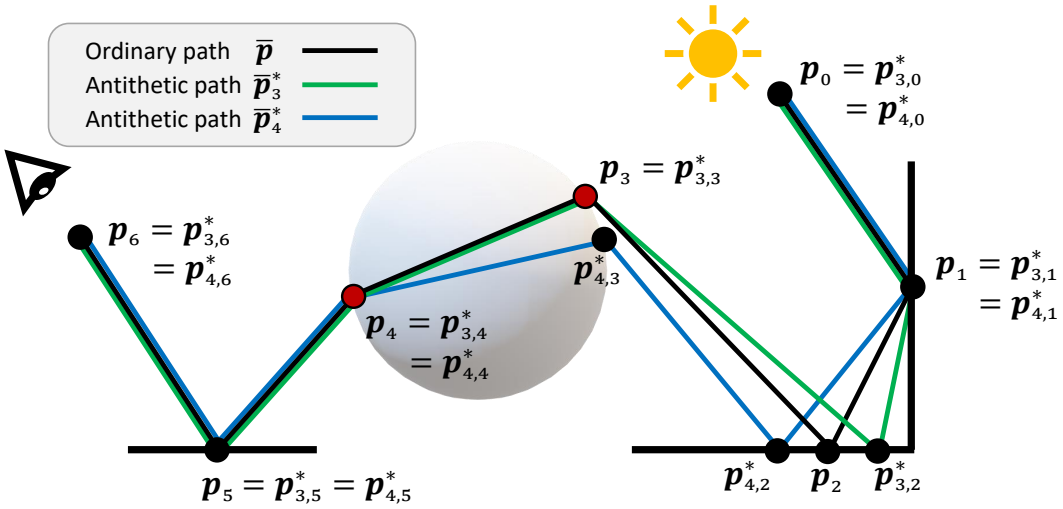


Figure 6.5: **Unidirectional construction of antithetic paths** (starting from the detector): In this example, we construct antithetic paths $\bar{\boldsymbol{p}}^*_i = (\boldsymbol{p}^*_{i,0}, \boldsymbol{p}^*_{i,1}, \ldots, \boldsymbol{p}^*_{i,6})$ for $i = 3, 4$ based on an *ordinary* one $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_6)$ that contains two vertices $\boldsymbol{p}_3$ and $\boldsymbol{p}_4$ (shown in red) with glossy BSDFs. To obtain the first antithetic path $\bar{\boldsymbol{p}}^*_3$, we apply BSDF antithetic sampling to $\boldsymbol{p}_3$ (by taking the antithetic sample), resulting in a new incident direction that in turn yields a new vertex $\boldsymbol{p}^*_{3,2}$. To obtain the second antithetic path $\bar{\boldsymbol{p}}^*_4$, we take the antithetic BSDF sample at $\boldsymbol{p}_4$, leading to a new vertex $\boldsymbol{p}^*_{4,3}$. Since this vertex has a glossy BSDF, we continue tracing (using standard BSDF sampling) and obtain $\boldsymbol{p}^*_{4,2}$ before mering back with the ordinary path.

Our basic idea is to decompose derivatives of the measurement contribution—by applying the product rule—as the sum of multiple terms each of which involves one BSDF derivative. In this way, we can apply BSDF antithetic sampling once per term, avoiding exponential branching. We describe how our technique works in the following and provide detailed derivations in Appendix 6.C.

Unidirectional sampling. The *unidirectional* variant of our technique starts with constructing an ordinary path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \boldsymbol{p}_1, \ldots, \boldsymbol{p}_N)$ using unidirectional path tracing. Assume that $\mathbb{I} \subseteq \{1, 2, \ldots, N-1\}$ denotes the indices of path vertices where BSDF antithetic sampling is needed. For each $i \in \mathbb{I}$, we accompany the same ordinary path $\bar{\boldsymbol{p}}$ with an antithesis $\bar{\boldsymbol{p}}^*_i$ generated by taking the antithetic incident direction sampled at the i-th vertex of the ordinary path $\bar{\boldsymbol{p}}$.

To maximize the consistency between $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}_i^*$, we adapt the gradient-domain path tracing (GDPT) [40], a forward-rendering technique. We note that the term "gradient-domain" in GDPT refers to image-space gradients that differ fundamentally from the *scene derivatives* with which differentiable rendering is concerned.

Specifically, given the ordinary path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$, our technique builds the antithetic path $\bar{\boldsymbol{p}}_i^* = (\boldsymbol{p}_{i,0}^*, \ldots, \boldsymbol{p}_{i,N}^*)$ as follows. The first $(i+1)$ vertices of the antithetic path coincide with those of the ordinary (that is, $\boldsymbol{p}_{i,j}^* = \boldsymbol{p}_j$ for all $0 \leq j \leq i$). The vertex $\boldsymbol{p}_{i,i+1}^*$ is obtained by tracing a ray from $\boldsymbol{p}_{i,i}^* = \boldsymbol{p}_i$ in the antithetic incident direction (given by our BSDF sampling). Then, starting from $\boldsymbol{p}_{i,i+1}^*$, we perform unidirectional path tracing with standard BSDF sampling until reaching a vertex $\boldsymbol{p}_{i,i'}^*$ with a non-glossy BSDF for some $i' \geq i+1$. Lastly, we merge the antithetic path $\bar{\boldsymbol{p}}_i^*$ back to the ordinary after $\boldsymbol{p}_{i,i'}^*$ by setting $\boldsymbol{p}_{i,k}^* = \boldsymbol{p}_k$ for all $k > i'$.

Further, for all $0 < j < N$, the vertex $\boldsymbol{p}_{i,j}^*$ of the antithetic path $\hat{\boldsymbol{\Omega}}_i^*$ and the vertex $\boldsymbol{p}_j$ of the ordinary $\bar{\boldsymbol{p}}$ must be either both glossy or both rough. If this requirement is not satisfied, the antithetic path is rejected and considered to have zero contribution.

We illustrate this process in Figure 6.5 and demonstrate its effectiveness in Figure 6.6.

Bidirectional sampling. Our ordinary and antithetic paths can also be generated in a *bidirectional* fashion. Specifically, we build two ordinary subpaths $\bar{\boldsymbol{p}}^{\mathrm{S}} = (\boldsymbol{p}_0^{\mathrm{S}}, \ldots, \boldsymbol{p}_N^{\mathrm{S}})$ and $\bar{\boldsymbol{p}}^{\mathrm{D}} = (\boldsymbol{p}_0^{\mathrm{D}}, \ldots, \boldsymbol{p}_M^{\mathrm{D}})$ originated at the source and the detector, respectively. Assume that BSDF antithetic sampling is needed at vertices with indices $\mathbb{I}^{\mathrm{S}}$ in the source subpath and $\mathbb{I}^{\mathrm{D}}$ in the detector subpath. Then, using the aforementioned unidirectional method, we build an antithetic source subpath $\bar{\boldsymbol{p}}_i^{\mathrm{S}*}$ for each $i \in \mathbb{I}^{\mathrm{S}}$ and an antithetic detector subpath $\bar{\boldsymbol{p}}_j^{\mathrm{D}*}$ for each $j \in \mathbb{I}^{\mathrm{D}}$.

With all ordinary and antithetic subpaths constructed, we then make bidirectional connec-

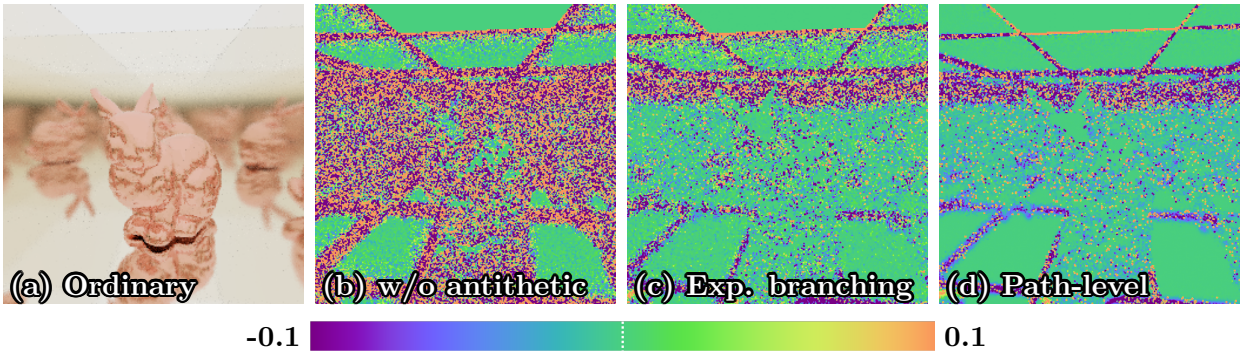


Figure 6.6: Our **path-level antithetic sampling** avoids exponential branching of light paths. In this example, we show a glossy scene with a bunny inside a box lit by an area light from the above, leading to many reflections of the bunny (a). Estimating derivatives with respect to the vertical position of the light using the unidirectional path-space method without antithetic sampling leads to very high variance (b). Using our BSDF sampling (§6.2) with the same base algorithm, much cleaner derivative estimates can be obtained (c). However, since each light path contains many vertices that require BSDF antithetic sampling, naïvely branching at each vertex has suboptimal performance. Our unidirectional path-level antithetic sampling (§6.4.1) addresses this problem and produces even lower variance (d). The derivative estimates in (b—d) are computed in equal time.

tions between the source and the detector subpaths as follows. That is, for each $i \notin \mathbb{I}^{\mathrm{S}}$ and $j \notin \mathbb{I}^{\mathrm{D}}$, we connect $\boldsymbol{p}_i^{\mathrm{S}}$ in the ordinary source subpath and $\boldsymbol{p}_j^{\mathrm{D}}$ in the ordinary detector subpath, resulting in a full ordinary path $\bar{\boldsymbol{p}}_{i,j}$. To obtain the antitheses of this ordinary path, we reuse the precomputed antithetic subpaths. Please refer to Figure 6.7 for an illustration of this process.

6.4.2 Pixel-Filter Antithetic Sampling with Vertices Reusing

We now address the limitations discussed in the end of §6.3 by introducing a new antithetic sampling technique for pixel reconstruction filters.

Given an *ordinary* path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{q})$, we construct (deterministically) three *antithetic* paths $\bar{\boldsymbol{p}}_i^* = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{q}_i^*)$ for $i = 1, 2, 3$. The ordinary and antithetic paths are identical except for the last vertices $\boldsymbol{q}$ and $\boldsymbol{q}_i^*$: Their image-plane projections of $\boldsymbol{y} = \mathtt{X}(\boldsymbol{q}, \theta)$

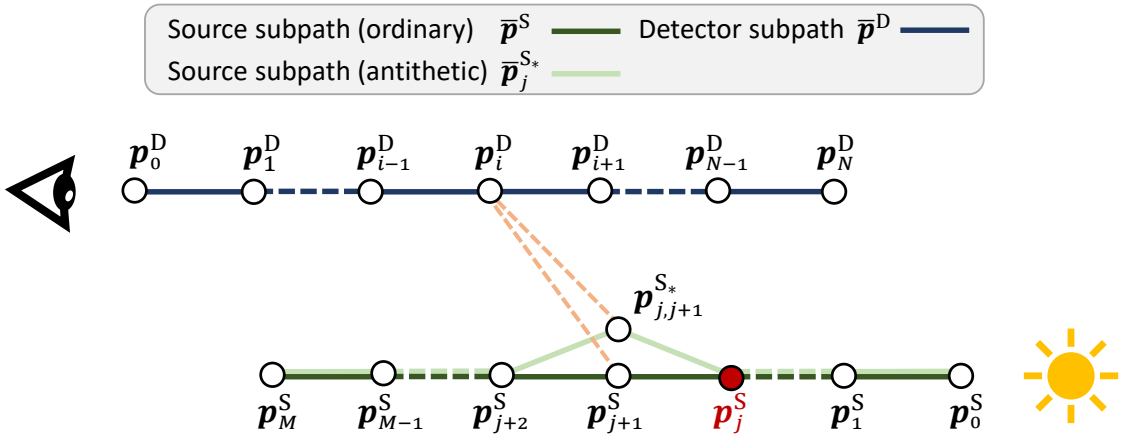


Figure 6.7: **Bidirectional construction of antithetic paths:** Let $\bar{\boldsymbol{p}}^{\mathrm{S}} = (\boldsymbol{p}_0^{\mathrm{S}}, \ldots, \boldsymbol{p}_M^{\mathrm{S}})$ be a pre-generated ordinary source subpath associated with antithesis $\bar{\boldsymbol{p}}_j^{\mathrm{S}*}$, and $\bar{\boldsymbol{p}}^{\mathrm{D}} = (\boldsymbol{p}_0^{\mathrm{D}}, \ldots, \boldsymbol{p}_N^{\mathrm{D}})$ be an ordinary detector subpath. Then, for any $0 \leq i \leq N$ and $j < j' \leq M$, connecting the i-th vertex $\boldsymbol{p}_i^{\mathrm{D}}$ on the ordinary detector subpath to, respectively, the j'-th vertex $\boldsymbol{p}_{j'}^{\mathrm{S}}$ on the ordinary source subpath and $\boldsymbol{p}_{j,j'}^{\mathrm{S}*}$ on its antithesis yields complete ordinary and antithetic paths $(\boldsymbol{p}_0^{\mathrm{S}}, \ldots, \boldsymbol{p}_{j'}^{\mathrm{S}}, \boldsymbol{p}_i^{\mathrm{D}}, \ldots, \boldsymbol{p}_0^{\mathrm{D}})$ and $(\boldsymbol{p}_{j,0}^{\mathrm{S}*}, \ldots, \boldsymbol{p}_{j,j'}^{\mathrm{S}*}, \boldsymbol{p}_i^{\mathrm{D}}, \ldots, \boldsymbol{p}_0^{\mathrm{D}})$. In this example, we have $j' = j + 1$. Similarly, we can connect a vertex $\boldsymbol{p}_j^{\mathrm{S}}$ from the ordinary source subpath to a pair of vertices $\boldsymbol{p}_{i'}^{\mathrm{D}}$ and $\boldsymbol{p}_{i,i'}^{\mathrm{D}*}$ from the detector subpaths (with $i < i' \leq N$) to form full ordinary-antithetic light paths.

and $\boldsymbol{y}_i^* = \mathtt{X}(\boldsymbol{q}_i^*, \theta)$ are symmetric around the pixel center, as illustrated in Figure 6.8. In comparison to the pixel-filter antithetic sampling process described in §6.3, the path antithetics $\boldsymbol{q}_i^*$ only differ from the ordinary path $\boldsymbol{q}_i^*$ by the last vertex. In other words, all vertices except for the last one (i.e., vertex $\boldsymbol{p}$) in the ordinary path $\bar{\boldsymbol{p}}$ are reused by the antithetic paths $\bar{\boldsymbol{p}}_i$ for all i.

Let $\mathrm{pdf}(\bar{\boldsymbol{p}})$ be the probability density of a material path $\bar{\boldsymbol{p}}$ sampled using standard techniques such as unidirectional and bidirectional path tracing as well as adjoint particle tracing. When the mapping (induced by aforementioned construction) between the original path $\bar{\boldsymbol{p}}$ and the antithetic one $\bar{\boldsymbol{p}}_i^*$ is *one-to-one* (for each $i = 1, 2, 3$), we can express the probability density $\mathrm{pdf}_i^*(\bar{\boldsymbol{p}}_i^*)$ of antithetic path $\bar{\boldsymbol{p}}_i^*$ analytically based on $\mathrm{pdf}(\bar{\boldsymbol{p}})$ as follows.

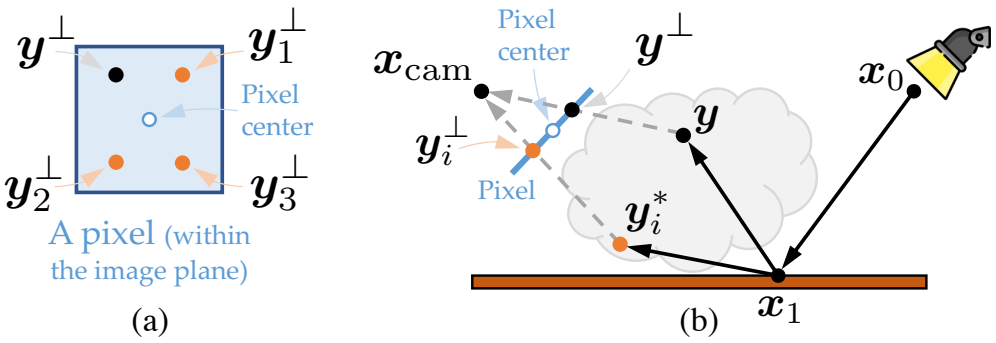


Figure 6.8: **Pixel-filter antithetic sampling with vertices reusing:** (a) Correlated sets of light paths are generated so that their intersections $\boldsymbol{y}^\perp$, $\boldsymbol{y}_1^\perp$, $\boldsymbol{y}_2^\perp$ and $\boldsymbol{y}_3^\perp$ with the image plane are symmetric around the pixel center. (b) Given an ordinary path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{y})$, we construct its antithesis $\bar{\boldsymbol{x}}_i^*$ by replacing only the last vertex with $\boldsymbol{y}_i^*$, yielding $\bar{\boldsymbol{x}}^* = (\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{y}_i^*)$ for $i = 1, 2, 3$.

Constructing Antithetic Paths

We now discuss, given an ordinary path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{q})$, how an antithetic path of the form $\bar{\boldsymbol{p}}^* = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{q}^*)$ can be constructed so that the induced mapping from $\bar{\boldsymbol{p}}$ to $\bar{\boldsymbol{p}}^*$ is *one-to-one*. We note that this process will be applied to obtain each antithetic path $\bar{\boldsymbol{p}}_i^*$ for $i = 1, 2, 3$.

Since $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}^*$ only differ by the last vertices, the problem amounts to constructing the last vertex $\boldsymbol{q}^*$ of the antithetic path $\bar{\boldsymbol{p}}^*$ in a one-to-one fashion. In the following, we discuss how $\boldsymbol{q}^*$ can be constructed based on the ordinary path $\bar{\boldsymbol{p}}$ and, more specifically, its last vertex $\boldsymbol{q}$.

Surface vertex. When $\boldsymbol{q}$ is a surface vertex (i.e., $\boldsymbol{q} \in \mathcal{B}_{\mathcal{M}}$), as illustrated in Figure 6.9-a, $\boldsymbol{q}^*$ can be constructed by:

1. Finding the image-plane projection $\boldsymbol{y}_1^\perp$ of $\boldsymbol{y} = \mathtt{X}(\boldsymbol{p}, \theta)$;
2. Obtaining the point $\boldsymbol{y}_2^\perp$ by mirroring $\boldsymbol{y}_1^\perp$ (based on antithetic sampling pattern);
3. Tracing a camera ray through the image-plane location $\boldsymbol{y}_2^\perp$ to obtain the first surface intersection $\boldsymbol{y}^*$ (while ignoring all media without refractive interfaces);

4. Letting $\boldsymbol{q}^* = \mathtt{X}^{-1}(\boldsymbol{y}^*, \theta)$.

Since the mapping between image-plane locations $\boldsymbol{y}_1^\perp$ and $\boldsymbol{y}_2^\perp$ is one-to-one, so is the mapping between $\boldsymbol{q}$ and $\boldsymbol{q}^*$.

To calculate the probability density $\mathrm{pdf}^*(\bar{\boldsymbol{p}}^*)$, we rely on the relation:

$$\mathrm{pdf}^*(\bar{\boldsymbol{p}}^*) = \mathrm{pdf}(\bar{\boldsymbol{p}}) \left\| \frac{\mathrm{d}A(\boldsymbol{q}^*)}{\mathrm{d}A(\boldsymbol{q})} \right\|. \tag{6.13}$$

We now derive the ratio $\|\frac{\mathrm{d}A(\boldsymbol{q}^*)}{\mathrm{d}A(\boldsymbol{q})}\|$ on the right-hand side of Eq. (6.13) due to the effective change of variable from $\boldsymbol{q}$ to $\boldsymbol{q}^*$. Let $\phi_{\boldsymbol{y}}$ and $\phi_{\boldsymbol{y}}^*$ denote, respectively, the angles from the camera's axis of projection $\boldsymbol{n}_{\mathrm{cam}}$ to directions $\overrightarrow{\boldsymbol{x}_{\mathrm{cam}}\,\boldsymbol{y}}$ and $\overrightarrow{\boldsymbol{x}_{\mathrm{cam}}\,\boldsymbol{y}^*}$. Then,

$$\begin{aligned} \mathrm{d}A(\boldsymbol{y}) &= \tfrac{\cos^3\phi_{\boldsymbol{y}}}{G_0(\boldsymbol{y}\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})}\,\mathrm{d}A(\boldsymbol{y}_1^\perp), \\ \mathrm{d}A(\boldsymbol{y}^*) &= \tfrac{\cos^3\phi_{\boldsymbol{y}}^*}{G_0(\boldsymbol{y}^*\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})}\,\mathrm{d}A(\boldsymbol{y}_2^\perp), \end{aligned} \tag{6.14}$$

where G_0 is the (standard) geometric term defined in Eq. (2.28). Further, due to the point symmetry between $\boldsymbol{y}_1^\perp$ and $\boldsymbol{y}_2^\perp$ on the image plane, we have $\mathrm{d}A(\boldsymbol{y}_1^\perp) = \mathrm{d}A(\boldsymbol{y}_2^\perp)$. Therefore, it holds that

$$\left\| \frac{\mathrm{d}A(\boldsymbol{y}^*)}{\mathrm{d}A(\boldsymbol{y})} \right\| = \frac{G_0(\boldsymbol{y}\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})\cos^3\phi_{\boldsymbol{y}}^*}{G_0(\boldsymbol{y}^*\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})\cos^3\phi_{\boldsymbol{y}}}. \tag{6.15}$$

It follows that

$$\left\| \frac{\mathrm{d}A(\boldsymbol{q}^*)}{\mathrm{d}A(\boldsymbol{q})} \right\| = \frac{G_0(\boldsymbol{y}\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})\cos^3\phi_{\boldsymbol{y}}^*}{G_0(\boldsymbol{y}^*\leftrightarrow\boldsymbol{x}_{\mathrm{cam}})\cos^3\phi_{\boldsymbol{y}}} \frac{J(\boldsymbol{q})}{J(\boldsymbol{q}^*)}. \tag{6.16}$$

In practice, as discussed in §3.3, when estimating derivatives at some $\theta = \theta_0$ with the reference surface set to $\mathcal{B}_{\mathcal{M}} = \mathcal{M}(\theta_0)$, both $\mathtt{X}(\cdot, \theta_0)$ and $\mathtt{X}^{-1}(\cdot, \theta_0)$ become identity maps. This causes $\boldsymbol{q}^*$ coincide with $\boldsymbol{y}^*$, and $\boldsymbol{q}$ with $\boldsymbol{y}$. Further, the factor $\frac{J(\boldsymbol{q})}{J(\boldsymbol{q}^*)}$ in Eq. (6.16) reduces to one.

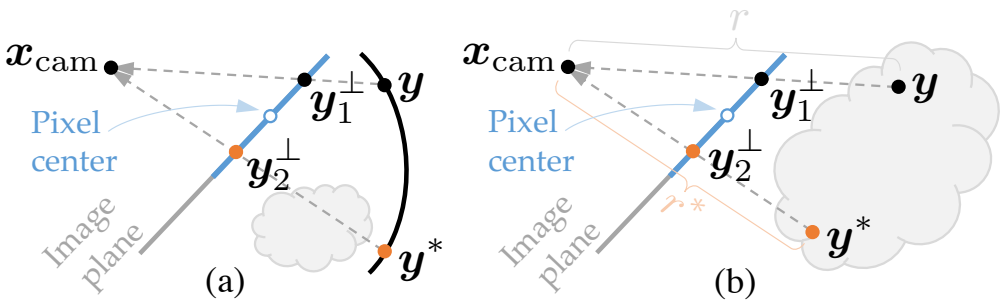


Figure 6.9: **Construction of antithetic path:** Given ordinary light path with the last vertex $\boldsymbol{y}$, we build the antithetic path that is identical to the ordinary except for the last vertex $\boldsymbol{y}^*$. We construct $\boldsymbol{y}^*$ such that the mapping between $\boldsymbol{y}$ and $\boldsymbol{y}^*$ is one-to-one based on the type of vertex $\boldsymbol{y}$: (a) When $\boldsymbol{y}$ is a surface vertex (i.e., $\boldsymbol{y} \in \mathcal{M}$), we trace a camera ray $\boldsymbol{x}_{\text{cam}} \to \boldsymbol{y}_2^\perp$ and set $\boldsymbol{y}^*$ as the first surface intersection while ignoring all medium (with index-matched interfaces); (b) When $\boldsymbol{y}$ is a volume vertex (i.e., $\boldsymbol{y} \in \mathcal{V} \setminus \mathcal{M}$), we set $\boldsymbol{y}^* = \boldsymbol{x}_{\text{cam}} + r^* \overrightarrow{\boldsymbol{x}_{\text{cam}}\, \boldsymbol{y}_2^\perp}$ with some $r^* > 0$ such that $T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y}) = T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y}^*)$.

Volume vertex. When $\boldsymbol{q}$ is a volume vertex (i.e., $\boldsymbol{q} \in \mathcal{B}_\mathcal{V} \setminus \mathcal{B}_\mathcal{M}$), we construct the vertex $\boldsymbol{q}^*$ using a three-step process identical to the aforementioned surface case except for the third step where we select the spatial point $\boldsymbol{y}^*$ along the ray $\boldsymbol{x}_{\text{cam}} \to \boldsymbol{y}_2^\perp$ such that, as illustrated in Figure 6.9-b, the transmittance between $\boldsymbol{x}_{\text{cam}}$ and $\boldsymbol{y}^*$ matches that between $\boldsymbol{x}_{\text{cam}}$ and $\boldsymbol{y}$: That is, $T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y}^*) = T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y})$.

Similar to Eq. (6.13) for the surface case, the probability density $\text{pdf}^*(\bar{\boldsymbol{p}}^*)$ satisfies that

$$\text{pdf}^*(\bar{\boldsymbol{p}}^*) = \text{pdf}(\bar{\boldsymbol{p}}) \left\| \frac{\mathrm{d}V(\boldsymbol{q}^*)}{\mathrm{d}V(\boldsymbol{q})} \right\|. \tag{6.17}$$

Let $r = \|\boldsymbol{y} - \boldsymbol{x}_{\text{cam}}\|$ and $r^* = \|\boldsymbol{y}^* - \boldsymbol{x}_{\text{cam}}\|$. Then, it is easy to verify:

$$\begin{aligned} {\mathrm{d}V(\boldsymbol{y})}/{r^2} &= \cos^3\phi_{\boldsymbol{y}}\, \mathrm{d}A(\boldsymbol{y}_1^\perp)\, \mathrm{d}r, \\ {\mathrm{d}V(\boldsymbol{y}^*)}/{(r^*)^2} &= \cos^3\phi^*_{\boldsymbol{y}}\, \mathrm{d}A(\boldsymbol{y}_2^\perp)\, \mathrm{d}r^*. \end{aligned} \tag{6.18}$$

Our construction of the vertex $\boldsymbol{y}^*$ implies that $\mathrm{d}A(\boldsymbol{y}_1^\perp) = \mathrm{d}A(\boldsymbol{y}_2^\perp)$ and ${\mathrm{d}r^*}/{\mathrm{d}r} = {\sigma_\mathrm{t}(\boldsymbol{y})}/{\sigma_\mathrm{t}(\boldsymbol{y}^*)}$. It follows that

$$\left\| \frac{\mathrm{d}V(\boldsymbol{q}^*)}{\mathrm{d}V(\boldsymbol{q})} \right\| = \left\| \frac{\mathrm{d}V(\boldsymbol{y}^*)}{\mathrm{d}V(\boldsymbol{y})} \right\| \frac{J(\boldsymbol{q})}{J(\boldsymbol{q}^*)} = \frac{\cos^3\phi^*_{\boldsymbol{y}}\, (r^*)^2\, \sigma_\mathrm{t}(\boldsymbol{y})}{\cos^3\phi_{\boldsymbol{y}}\, r^2\, \sigma_\mathrm{t}(\boldsymbol{y}^*)} \frac{J(\boldsymbol{q})}{J(\boldsymbol{q}^*)}. \tag{6.19}$$

Similar to the surface case, when estimating derivatives at $\theta = \theta_0$ with the reference volume set to $\mathcal{B}_\mathcal{V} = \mathcal{V}(\theta_0)$, the factor ${}^{J(\boldsymbol{q})}/_{J(\boldsymbol{q}^*)}$ in Eq. (6.19) reduces to one.

Failed constructions. Occasionally, for an ordinary path $\bar{\boldsymbol{p}}$, the construction of its antitheses $\bar{\boldsymbol{p}}_i^*$ (presented in §6.4.2) can fail. This happens when: (i) the camera ray for the antithetic path does not intersect any surface in the scene for the surface case; or (ii) there does not exist a point $\boldsymbol{y}^*$ along the ray satisfying the transmittance constraint $T(\boldsymbol{x}_\text{cam} \leftrightarrow \boldsymbol{y}^*) = T(\boldsymbol{x}_\text{cam} \leftrightarrow \boldsymbol{y})$. When the construction fails, we simply consider the antithetic path $\bar{\boldsymbol{p}}^*$ to be nonexistent and set its contribution to zero.

Similarly, when calculating the probability density $\text{pdf}_i^*(\bar{\boldsymbol{p}})$ for some path $\bar{\boldsymbol{p}}$, there may not exist an ordinary path whose i-th antithesis equals $\bar{\boldsymbol{p}}$. In this case, we set $\text{pdf}_i^*(\bar{\boldsymbol{p}}) = 0$.

Equal-transmittance vs. equal-distance. When $\boldsymbol{y}$ is a volume vertex (Figure 6.9-b), its antithesis $\boldsymbol{y}^*$ can also be constructed in a *equal-distance* fashion by setting $r^* = r$ rather than requiring $T(\boldsymbol{x}_\text{cam} \leftrightarrow \boldsymbol{y}^*)$ to match $T(\boldsymbol{x}_\text{cam} \leftrightarrow \boldsymbol{y})$. This, however, tends to produce higher variance (see Figure 6.10). On the other hand, when the aforementioned equal-transmittance construction is difficult—which can happen with heterogeneous media—the equal-distance variant can be used as a backup.

Handling multiple pixels. When (the supports of) reconstruction filters of neighboring pixels overlap, one light path $\bar{\boldsymbol{p}}$ can "intersect" with (i.e., contribute to) multiple pixels. In this case, for each intersecting pixel j, we construct one antithetic path $\bar{\boldsymbol{p}}_j^*$ for using the same ordinary path $\bar{\boldsymbol{p}}$, and use our MIS estimator of Eq. (6.20) with $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}_j^*$ for this pixel.

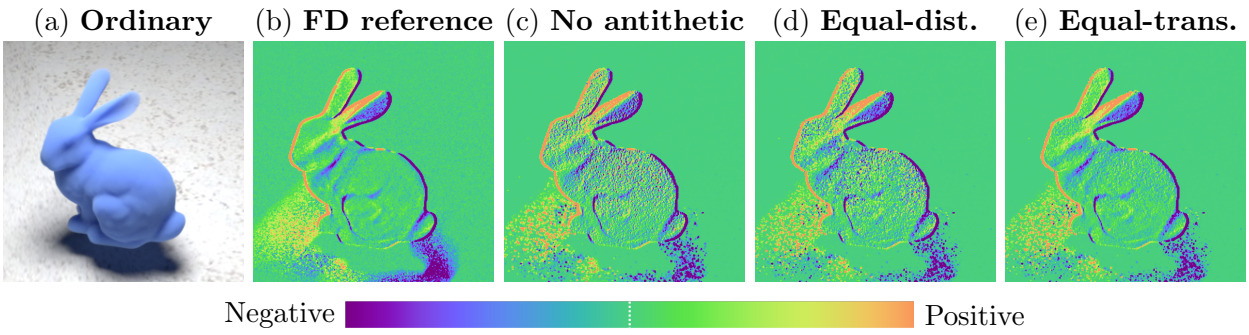


Figure 6.10: **Equal-time comparison** of derivate images rendered with volumetric adjoint particle tracing with (c) no antithetic sampling; (d) our *equal-distance* antithetic sampling with $r^* = r$; and (e) our *equal-transmittance* antithetic sampling with $T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y}^*) = T(\boldsymbol{x}_{\text{cam}} \leftrightarrow \boldsymbol{y})$. This example contains a homogeneous translucent bunny (without refractive interfaces) lit by an area light and uses the tent reconstruction filter (6.10). The derivatives (visualized using the same color map as Figure 6.4) are computed with respect to the horizontal translation of the bunny.

Multiple importance sampling. With both the ordinary path $\bar{\boldsymbol{p}}$ and its antitheses $\bar{\boldsymbol{p}}_i^*$ generated, we combine their contributions using multiple importance sampling (MIS) via:

$$w(\bar{\boldsymbol{p}}) \frac{\frac{\mathrm{d}}{\mathrm{d}\theta}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{pdf}(\bar{\boldsymbol{p}})} + \sum_i w_i^*(\bar{\boldsymbol{p}}_i^*) \frac{\frac{\mathrm{d}}{\mathrm{d}\theta}\hat{f}(\bar{\boldsymbol{p}}_i^*)}{\mathrm{pdf}_i^*(\bar{\boldsymbol{p}}_i^*)}, \tag{6.20}$$

where w and w^* are the MIS weighting functions which we set using the balanced heuristics [81]: $w(\bar{\boldsymbol{p}}) = \frac{\mathrm{pdf}(\bar{\boldsymbol{p}})}{\mathrm{pdf}(\bar{\boldsymbol{p}}) + \sum_j \mathrm{pdf}_j^*(\bar{\boldsymbol{p}})}$ and $w_i^*(\bar{\boldsymbol{p}}) = \frac{\mathrm{pdf}_i^*(\bar{\boldsymbol{p}})}{\mathrm{pdf}(\bar{\boldsymbol{p}}) + \sum_j \mathrm{pdf}_j^*(\bar{\boldsymbol{p}})}$, for any material path $\bar{\boldsymbol{p}}$ and $i = 1, 2, 3$.

6.5 Results

We now evaluate our antithetic sampling techniques introduced earlier in this chapter by comparing differentiable rendering results obtained with our approach to finite-difference (FD) references and baseline methods. We will show more inverse-rendering results in Chapter 8.

6.5.1 BSDF Antithetic Sampling

We compare derivatives estimated with and without antithetic sampling using two base differentiable rendering algorithms: unidirectional path tracing with edge sampling [45] indicated as "Edge" and our path-space methods with "PS1" indicating the unidirectional algorithm (**I.1**) and "PS2" the bidirectional one (**I.2**). When applying antithetic sampling, we use our BSDF-level variant (discussed in §6.2) with "Edge" and the path-level one (presented in §6.4.1) with "PS1" and "PS2".

In both the edge sampling and path-space framework, an image derivative consists of *boundary* and *interior* components. We only show the *interior* components for comparison since the estimation of *boundary* integrals is orthogonal to our work.

Isotropic BSDFs. In Figure 6.11, we show a few scenes with glossy objects depicted with isotropic microfacet BSDFs.

The TEAPOT scene contains a glossy teapot lit by an area light. The derivatives are computed with respect to the rotation angle of the teapot (about its vertical axis). Using differentiable path tracing (Edge) [45], our BSDF antithetic sampling offers a speedup of over 60× to produce derivative estimates with approximately the same quality. We conduct the equal-quality comparisons by: (i) generating a reference image with low noise; and (ii) computing derivative images with and without antithetic sampling progressively until the differences between the rendered results and the reference drops below a predetermined threshold. At equal time, standard BSDF sampling produces high variance in specular highlights on the teapot. When using our antithetic BSDF sampling, on the other hand, much cleaner derivative estimates can be obtained.

The rest of the examples in Figure 6.11 are rendered using the path-space method. The CORNELL-BOX scene contains a glossy sphere, and the derivatives are computed using our

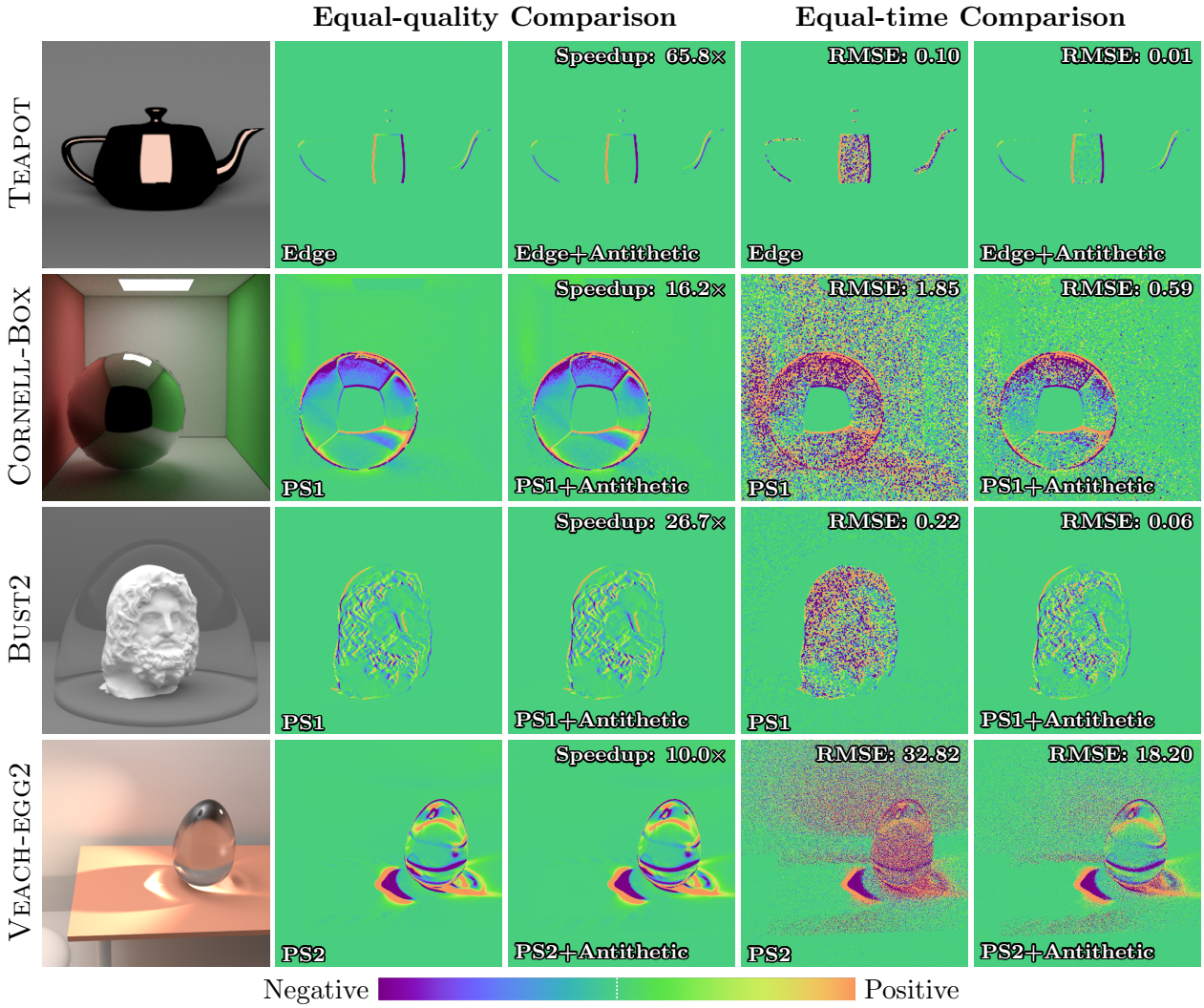


Figure 6.11: **Differentiable rendering of isotropic BSDFs:** We show (the *interior* components of) derivatives estimated with and without our antithetic sampling technique using different base differentiable rendering methods.

unidirectional algorithm (PS1) with respect to the vertical translation of the sphere. At equal quality, our path-level antithetic sampling (§6.4.1) offers a 16.2× speedup. At equal time, the estimated derivatives contain high variance without antithetic sampling. We note that even the non-glossy regions (such as the diffuse walls) suffer from high noise due to interreflections. With our technique, in contrast, the variance is greatly reduced.

The BUST2 scene consists of a diffuse bust (whose 3D model is from McGuire's Computer Graphics Archive [52]) inside a glossy glass dome, and we compute derivatives with respect to

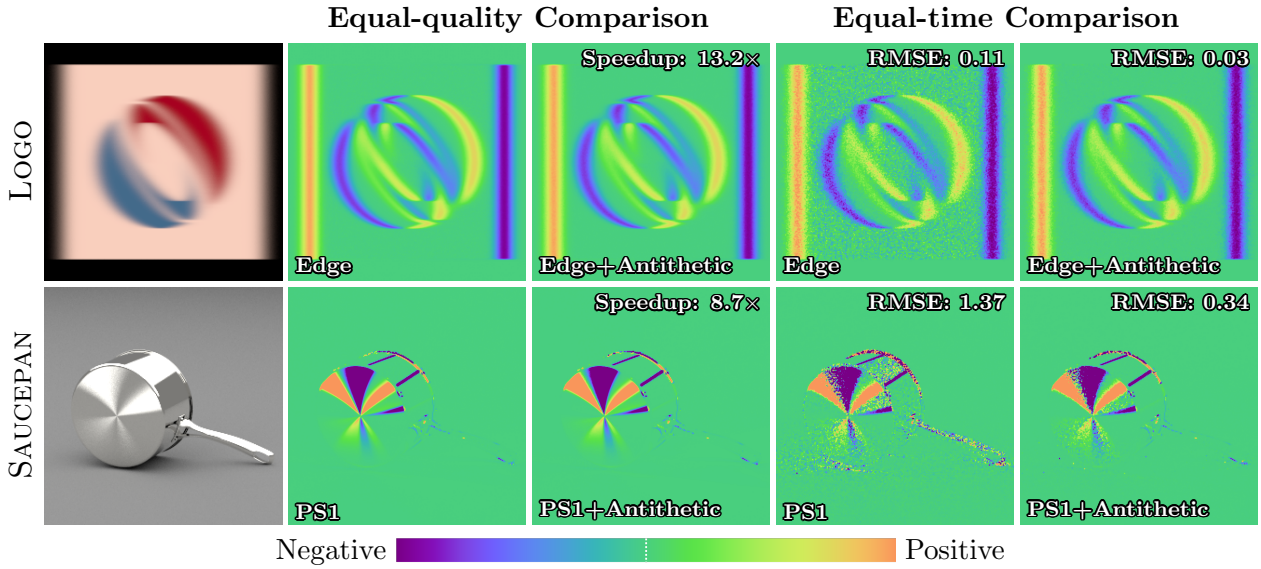


Figure 6.12: **Differentiable rendering of anisotropic BRDFs:** We show (the *interior* components of) derivatives obtained with and without our antithetic sampling technique using different base differentiable rendering methods.

the rotation of the bust about its vertical axis. Our antithetic sampling achieves a speedup of 26.7× to generate equal-quality derivative estimates and produces significantly lower variance at equal time.

Lastly, we reuse the Veach-egg2 scene introduced in §5.3 but with a different view. This scene contains a glass egg lit by a small spot light, creating caustics on the table. The derivatives are computed with respect to the vertical translation of the egg. Due to the complexity of light transport in this example, we estimate the derivatives using our bidirectional algorithm (PS2). Our path-level antithetic sampling provides a 10× speedup and, similar to the previous examples, offers much cleaner results at equal time.

Anisotropic BRDFs. Our antithetic sampling technique also applies to anisotropic BSDFs, which we demonstrate in Figure 6.12. Similar to Figure 6.11, we only show contributions of the *interior* terms.

The logo scene shows the virtual image of a SIGGRAPH logo on an anisotropic reflector

with derivatives computed with respect to the rotation angle of the logo around its horizontal axis. Our antithetic sampling technique achieves a speedup of 13.2× at equal-quality and provides considerably more accurate results in equal time.

The SAUCEPAN scene contains a glossy saucepan made of brushed metal lit by a small area light, resulting in characteristic anisotropic highlights on the bottom. When computing derivatives with respect to the vertical rotation of the saucepan, our technique offers a speedup of 8.7× at equal quality and much lower image RMSE in equal time.

6.5.2 Pixel-Filter Antithetic Sampling

In §6.3, we have demonstrated in Figures 6.3 and 6.4 the effectiveness of our pixel-filter antithetic sampling without reusing path vertices. We now evaluate the path-level extension introduced in §6.4.2 using differentiable-rendering results.

In Figure 6.13, we show results generated using adjoint particle tracing (APT) and bidirectional path tracing (BDPT). As discussed in §6.2, without reusing path vertices, pixel-filter antithetic sampling has difficulties in supporting methods like APT and BDPT that trace "shadow rays" toward the camera.

The EARTH and BUST3 examples in Figure 6.13 contain, respectively, a textured diffuse earth object and a homogeneous translucent bust. Both examples are rendered using APT. Additionally, the KITTY result involves a glossy kitty (from the Digitalized 3D Object Collection [16]) inside a Cornell box lit by an area light facing toward the ceiling, creating a challenging light-transport situation. The GLASS-BALL example contains a rough glass sphere lit by a small area light. Both examples are rendered using BDPT. For all examples, our vertex-reusing technique provides significant variance reduction.

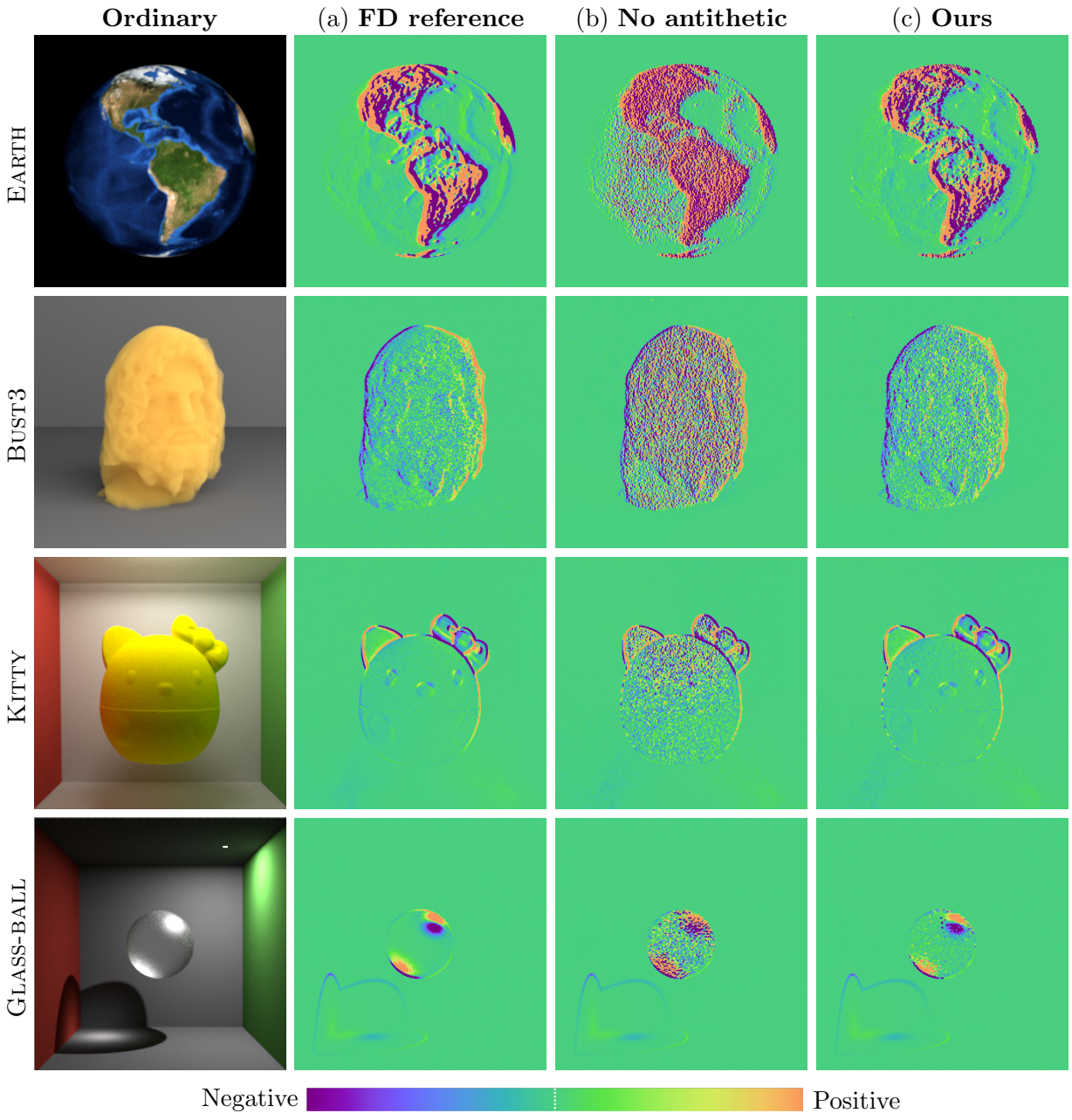


Figure 6.13: **Pixel-filter antithetic sampling with vertices reusing:** To demonstrate the effectiveness of our pixel-filter antithetic sampling (with vertex reusing) (§6.4.1), we show equal-time comparisons of derivative estimateswith respect to object translations. Results in (b) and (c) are generated without and with our antithetic sampling, respectively. The EARTH and the BUST3 examples are generated with volumetric adjoint particle tracing (APT); the KITTY and the GLASS-BALL use bidirectional path tracing (BDPT). All examples use the tent pixel reconstruction filter.

6.A Analysis of Antithetic Estimators

Let (Ω, μ) be some measure space. In what follows, we consider the problem of estimating integrals of the form

$$I = \int_\Omega f(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x}). \tag{6.21}$$

Specifically, let $\beta : \Omega \mapsto \Omega$ be a diffeomorphism satisfying that $\beta(\beta(\boldsymbol{x})) = \boldsymbol{x}$ (that is, $\beta = \beta^{-1}$) and $|\det \mathrm{D}\beta(\boldsymbol{x})| = 1$ for all $\boldsymbol{x} \in \Omega$. Then, we will show that

$$\boxed{\langle I \rangle_{\mathrm{dual}} := \frac{f(\boldsymbol{X}) + f(\beta(\boldsymbol{X}))}{p(\boldsymbol{X}) + p(\beta(\boldsymbol{X}))},} \tag{6.22}$$

is an unbiased estimator of Eq. (6.21) where $\boldsymbol{X}$ is a random variable over Ω with probability density p (such that $f(\boldsymbol{x}) \neq 0$ implies $p(\boldsymbol{x}) > 0$).

Proof of unbiasedness Since the Jacobian determinant of β is assumed to have unit absolution value, applying a change of variable from $\boldsymbol{x}$ to $\boldsymbol{y} = \beta(\boldsymbol{x})$ yields

$$\int_\Omega f(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x}) = \int_\Omega f(\boldsymbol{y}^*)\,\mathrm{d}\mu(\boldsymbol{y}), \tag{6.23}$$

$$\int_\Omega p(\boldsymbol{x})\,\frac{f(\boldsymbol{x}) + f(\boldsymbol{x}^*)}{p(\boldsymbol{x}) + p(\boldsymbol{x}^*)}\,\mathrm{d}\mu(\boldsymbol{x}) = \int_\Omega p(\boldsymbol{y}^*)\,\frac{f(\boldsymbol{y}^*) + f(\boldsymbol{y})}{p(\boldsymbol{y}^*) + p(\boldsymbol{y})}\,\mathrm{d}\mu(\boldsymbol{y}), \tag{6.24}$$

where $\boldsymbol{x}^* := \beta(\boldsymbol{x})$ and $\boldsymbol{y}^* := \beta(\boldsymbol{y})$. Then, it holds that

$$\begin{aligned}
\mathbb{E}\left[\langle I \rangle_{\mathrm{dual}}\right] &= \frac{1}{2}\left[2\int_\Omega p(\boldsymbol{x})\,\frac{f(\boldsymbol{x}) + f(\boldsymbol{x}^*)}{p(\boldsymbol{x}) + p(\boldsymbol{x}^*)}\,\mathrm{d}\mu(\boldsymbol{x})\right] \\
&= \frac{1}{2}\left[\int_\Omega p(\boldsymbol{x})\,\frac{f(\boldsymbol{x}) + f(\boldsymbol{x}^*)}{p(\boldsymbol{x}) + p(\boldsymbol{x}^*)}\,\mathrm{d}\mu(\boldsymbol{x}) + \int_\Omega\ p(\boldsymbol{y}^*)\,\frac{f(\boldsymbol{y}^*) + f(\boldsymbol{y})}{p(\boldsymbol{y}^*) + p(\boldsymbol{y})}\,\mathrm{d}\mu(\boldsymbol{y})\right] \\
&= \frac{1}{2}\int_\Omega \left[f(\boldsymbol{x}) + f(\boldsymbol{x}^*)\right]\mathrm{d}\mu(\boldsymbol{x}) \\
&= \int_\Omega f(\boldsymbol{x})\,\mathrm{d}\mu(\boldsymbol{x}) = I.
\end{aligned} \tag{6.25}$$

Variance analysis We now analyze on the variance of antithetic sampling, in comparison to the traditional Monte Carlo estimator

$$\langle I \rangle_{\text{single}} := f(\boldsymbol{X})/p(\boldsymbol{X}). \tag{6.26}$$

We have

$$\begin{aligned}
\operatorname{Var}\left[\langle I \rangle_{\text{dual}}\right] - \operatorname{Var}\left[\langle I \rangle_{\text{single}}\right] &= \int_\Omega p(\boldsymbol{x}) \left[\frac{f(\boldsymbol{x}) + f(\boldsymbol{x}^*)}{p(\boldsymbol{x}) + p(\boldsymbol{x}^*)}\right]^2 \mathrm{d}\mu(\boldsymbol{x}) - \int_\Omega p(\boldsymbol{x}) \left[\frac{f(\boldsymbol{x})}{p(\boldsymbol{x})}\right]^2 \mathrm{d}\mu(\boldsymbol{x}) \\
&\leq \int_\Omega \frac{[f(\boldsymbol{x}) + f(\boldsymbol{x}^*)]^2}{p(\boldsymbol{x})} \mathrm{d}\mu(\boldsymbol{x}) - \int_\Omega \frac{f(\boldsymbol{x})^2}{p(\boldsymbol{x})} \mathrm{d}\mu(\boldsymbol{x}) \\
&= \int_\Omega \frac{[f(\boldsymbol{x}) + f(\boldsymbol{x}^*)]^2 - f(\boldsymbol{x})^2}{p(\boldsymbol{x})} \mathrm{d}\mu(\boldsymbol{x}).
\end{aligned} \tag{6.27}$$

Thus, if $|f(\boldsymbol{x}) + f(\boldsymbol{x}^*)| \leq |f(\boldsymbol{x})|$ for all $\boldsymbol{x} \in \Omega$, we have $\operatorname{Var}[\langle I \rangle_{\text{dual}}] \leq \operatorname{Var}[\langle I \rangle_{\text{single}}]$. In other words, if we can have $f(\boldsymbol{x})$ and $f(\beta(\boldsymbol{x}))$ to cancel each other out, the antithetic-sampling estimator given by Eq. (6.22) will offer a lower variance.

6.B Problems of Attached Sampling

Assuming the pixel reconstruction filter can be perfectly importance sampled, the intensity I of a pixel can also be expressed using a primary-sample-space integral:

$$I = \int_{[0,1)^2} c\, L_{\text{i}}(\boldsymbol{x}_{\text{cam}}(\theta) \to \boldsymbol{y}^\perp(\boldsymbol{\xi}; \theta))\, \mathrm{d}\boldsymbol{\xi}, \tag{6.28}$$

where c is a normalization factor, and L_{i} denotes incident radiance that can, in turn, be estimated using path integrals (3.9). Additionally, $\boldsymbol{y}^\perp : [0,1)^2 \mapsto \mathcal{I}$ is a mapping that encodes the importance sampling (of the pixel reconstruction filter) and transforms a primary sample $\boldsymbol{\xi} \sim U[0,1)^2$ to a point on the image plane $\mathcal{I}$. We note that, when the scene

evolves with a parameter θ, the mapping $\boldsymbol{y}^\perp$ generally depends on θ and, thus, needs to be differentiated when estimating derivatives of Eq. (6.28). This formulation is also known as *attached sampling* [92] (applied to the pixel reconstruction filter $\mathcal{P}$).

Under the attached-sampling-based formulation of Eq. (6.28), for any light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-2}, \boldsymbol{y})$, the last vertex $\boldsymbol{y}$ is given by attached sampling (and ray intersection), and the remaining ones $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-2}$—which are used to estimate the incident radiance $L_\mathrm{i}(\boldsymbol{x}_\mathrm{cam} \to \boldsymbol{y}^\perp)$ and its derivative—by our path-space method. Thus, the change rates of $\boldsymbol{y}$ and $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-2}$ (with respect to θ) are calculated differently where the former is given differentiating the sampling and ray intersection process:

$$\frac{\mathrm{d}\boldsymbol{y}}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta}\mathrm{rayIntersect}(\boldsymbol{x}_\mathrm{cam}(\theta) \to \boldsymbol{y}^\perp(\boldsymbol{\xi}; \theta)), \tag{6.29}$$

and the latter by differentiating the material-form parameterization:

$$\frac{\mathrm{d}\boldsymbol{x}_n}{\mathrm{d}\theta} = \frac{\mathrm{d}}{\mathrm{d}\theta}\mathtt{X}(\boldsymbol{p}_n, \theta), \tag{6.30}$$

for all $n = 0, 1, \ldots, N-2$ with $\boldsymbol{p}_n$ being a material point independent of the scene parameter θ. The discrepancy between Eqs. (6.29) and (6.30) becomes problematic when the vertex $\boldsymbol{y}$ and its neighbor $\boldsymbol{x}_{N-2}$ are very close to each other—which can occur at corners or in participating media (as illustrated in Figure 6.14). Precisely, when the distance $\|\boldsymbol{y} - \boldsymbol{x}_{N-2}\|$ between $\boldsymbol{y}$ and $\boldsymbol{x}_{N-2}$ approaches zero while the difference $\|(\mathrm{d}\boldsymbol{y}/\mathrm{d}\theta) - (\mathrm{d}\boldsymbol{x}_{N-2}/\theta)\|$ between their derivatives does not, the term $\frac{G(\boldsymbol{y} \leftrightarrow \boldsymbol{x}_{N-2})}{\mathrm{d}\theta}/G(\boldsymbol{y} \leftrightarrow \boldsymbol{x}_{N-2})$ diverges (i.e., goes to infinity). This term is a factor of the *interior* path integral when differentiating $L_\mathrm{i}(\boldsymbol{x}_\mathrm{cam} \to \boldsymbol{y}^\perp)$ in Eq. (6.28). As demonstrated in Figure 6.15, this divergence can lead to very high variance in resulting derivative estimates. In contrast, our method uses the same material-form parameterization for all vertices and, thus, does not suffer from this problem.

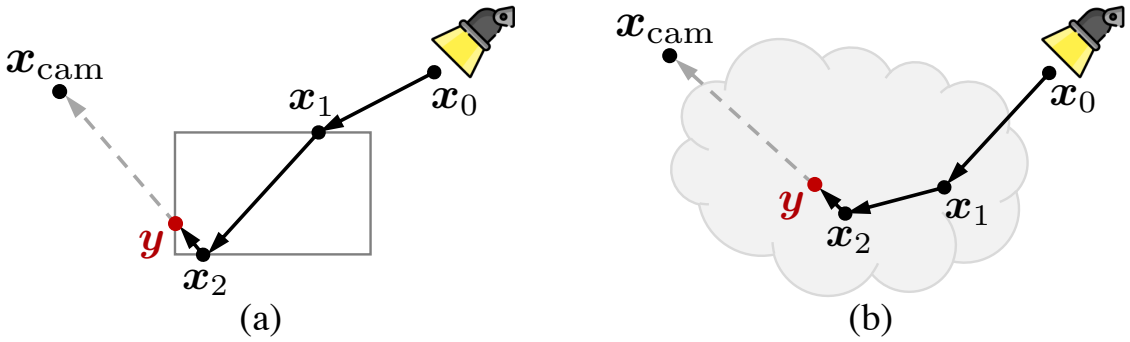


Figure 6.14: **Problem of pixel-filter attached sampling:** For any light path $\bar{\boldsymbol{x}} = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_{N-2}, \boldsymbol{y})$, when the last vertex $\boldsymbol{y}$ is drawn by applying attached sampling to the pixel reconstruction filter, the change rate $\mathrm{d}\boldsymbol{y}/\mathrm{d}\theta$ is determined by the sampling process and differentiable ray intersection—which differs from the material-form parameterization (discussed in §3.2) that give the change rates of other vertices $\boldsymbol{x}_0, \dots, \boldsymbol{x}_{N-2}$. When the distance between $\boldsymbol{y}$ and the neighboring vertex $\boldsymbol{x}_{N-2}$ (e.g., $\boldsymbol{x}_2$ in this figure) approaches to zero—which can occur at object edges/corners (a) and in the interior of participating media (b)—the change rates $\mathrm{d}\boldsymbol{y}/\mathrm{d}\theta$ and $\mathrm{d}\boldsymbol{x}_{N-2}/\mathrm{d}\theta$ remain different, leading to highly noisy derivative estimates.

6.C Derivations of Pixel-Filter Antithetic Sampling

We now derive our pixel-filter antithetic sampling technique (§6.4.1). For notation simplicity, we only consider the light transport between surfaces.

Let $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \dots, \boldsymbol{p}_N) \in \hat{\boldsymbol{\Omega}}$ be some material light path and $\bar{\mathtt{X}}(\bar{\boldsymbol{p}}, \theta) = (\boldsymbol{x}_0, \dots, \boldsymbol{x}_N) \in \boldsymbol{\Omega}(\theta)$ be the corresponding ordinary path with $\boldsymbol{x}_i = \mathtt{X}(\boldsymbol{p}_i, \theta)$ for $i = 0, 1, \dots, N$. Given $\mathcal{I} \subseteq \{1, 2, \dots, N-1\}$ consisting of vertex indices such that BSDF antithetic sampling is needed at $\boldsymbol{p}_i$ for each $i \in \mathcal{I}$, we factor out BSDF terms at these vertices in the material measurement contribution of Eq. (3.10), yielding:

$$\hat{f}(\bar{\boldsymbol{p}}) = \hat{f}_0(\bar{\boldsymbol{p}}) \prod_{i \in \mathcal{I}} f_{\mathrm{s}}[i](\bar{\boldsymbol{p}}), \tag{6.31}$$

where $f_{\mathrm{s}}[i](\bar{\boldsymbol{p}}) := f_{\mathrm{s}}(\boldsymbol{x}_{i-1} \to \boldsymbol{x}_i \to \boldsymbol{x}_{i+1})$; and $\hat{f}_0$ consists of all the other terms in $\hat{f}$ including rough BSDFs that do not need to be antithetically sampled, the geometric terms, and the Jacobian determinant in Eq. (3.10). Then, according to the product rule, differentiating

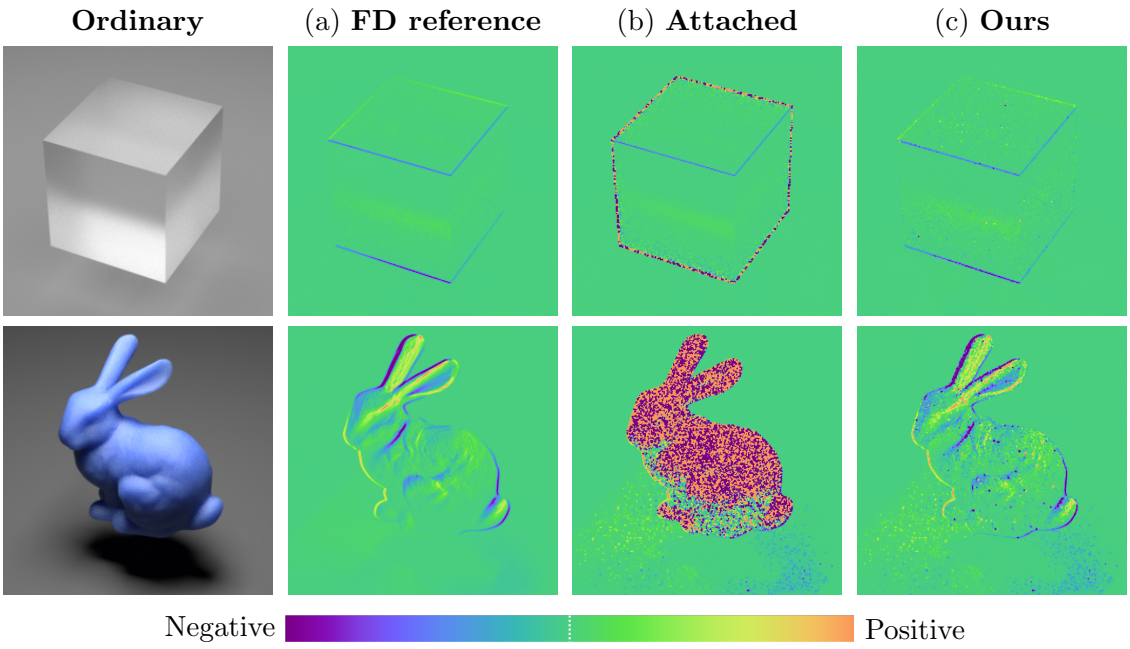


Figure 6.15: **Equal-time comparison** of derivative images estimated with the vertex $\boldsymbol{y}$ drawn using pixel-filter attached sampling (b) and our antithetic sampling (c). All results use the tent reconstruction filter (6.10) and are rendered with unidirectional path tracing.

Eq. (6.31) gives:

$$\frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} = \frac{\mathrm{d}\hat{f}_0(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \prod_{i\in\mathcal{I}} f_\mathrm{s}[i](\bar{\boldsymbol{p}}) + \hat{f}_0(\bar{\boldsymbol{p}}) \sum_{i\in\mathcal{I}} \frac{\mathrm{d}f_\mathrm{s}[i](\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \prod_{j\in\mathcal{I}\setminus\{i\}} f_\mathrm{s}[j](\bar{\boldsymbol{p}}). \tag{6.32}$$

It follows that the *interior* term of Eq. (4.14) can be rewritten as

$$\int_{\hat{\boldsymbol{\Omega}}} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\theta}\,\mathrm{d}\mu(\bar{\boldsymbol{p}}) = \int_{\hat{\boldsymbol{\Omega}}} \frac{\mathrm{d}\hat{f}_0(\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \left(\prod_{i\in\mathcal{I}} f_\mathrm{s}[i](\bar{\boldsymbol{p}})\right) \mathrm{d}\mu(\bar{\boldsymbol{p}}) + \sum_{i\in\mathcal{I}} \left[\int_{\hat{\boldsymbol{\Omega}}} \hat{f}_0(\bar{\boldsymbol{p}}) \frac{\mathrm{d}f_\mathrm{s}[i](\bar{\boldsymbol{p}})}{\mathrm{d}\theta} \left(\prod_{j\in\mathcal{I}\setminus\{i\}} f_\mathrm{s}[j](\bar{\boldsymbol{p}})\right) \mathrm{d}\mu(\bar{\boldsymbol{p}})\right]. \tag{6.33}$$

We note that the right-hand side of Eq. (6.33) involves multiple path integrals where the first one does not involve derivatives of glossy BSDFs and can be handled using an ordinary path $\bar{\boldsymbol{p}}$ generated with standard unidirectional or bidirectional method.

Each remaining path integral, on the other hand, involves exactly one derivative of the form $\mathrm{d}f_{\mathrm{s}[i]}/\mathrm{d}\theta$. We estimate this integral using $\bar{\boldsymbol{p}}$ and its *antithesis* $\bar{\boldsymbol{p}}_i^*$, as discussed in §6.4.1.

Chapter 7

Efficient Computational Differentiation

Thus far, our discussion has mostly focused on the differentiation of scalar sensor responses $I \in \mathbb{R}_{\geq 0}$ with respect to one scene parameter $\theta \in \mathbb{R}$. On the other hand, most practical problems involve vector-valued sensor responses $\boldsymbol{I} \in \mathbb{R}_{\geq 0}^{m_{\boldsymbol{I}}}$ (e.g., as an image with $m_{\boldsymbol{I}}$ pixels) and multiple scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$. When both the response $\boldsymbol{I}$ and the scene parameters $\boldsymbol{\theta}$ are high-dimensional (i.e., $m_{\boldsymbol{I}}$ and $m_{\boldsymbol{\theta}}$ are large), efficient differentiation becomes crucial for the practicality of differentiable rendering systems.

In this chapter, we devise a mathematical formulation that expresses image-loss gradients as differential path integrals comprised of an *interior* and a *boundary* components (§7.1). Based on this formulation, we propose an algorithm to efficiently compute the *interior* component by exploiting the layered structure of the computation graph (§7.2). Additionally, we discuss how the *boundary* component can be estimated in a unified fashion (§7.3).

7.1 Differential Image-Loss Path Integrals

Physics-based rendering typically involves estimating multiple, say $m_{\boldsymbol{I}}$, response values (e.g., one per pixel). To this end, we make the (material) measurement contribution function to be vector-valued, denoted as $\hat{\boldsymbol{f}}$. Then, the material-form path integral (3.9) can be rewritten in a vector-valued form as

$$\boldsymbol{I} = \int_{\hat{\boldsymbol{\Omega}}} \hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}) \, \mathrm{d}\mu(\bar{\boldsymbol{p}}), \tag{7.1}$$

where $\boldsymbol{I}$, $\hat{f}(\bar{\boldsymbol{p}})$ are $m_{\boldsymbol{I}}$-dimensional (column, i.e., $\mathbb{R}^{m_{\boldsymbol{I}} \times 1}$) vectors. Assuming we have $m_{\boldsymbol{\theta}}$ scene parameters denoted as $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times 1}$, by differentiating Eq. (7.1) with respect to $\boldsymbol{\theta}$, it is easy to verify that the vector-valued form of the differential path integrals (4.14) is

$$\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}} = \overbrace{\int_{\hat{\boldsymbol{\Omega}}} \frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \, \mathrm{d}\mu(\bar{\boldsymbol{p}})}^{\text{interior}} + \overbrace{\int_{\partial\hat{\boldsymbol{\Omega}}} \Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}) \, \boldsymbol{v}_\perp(\boldsymbol{p}_K)^\perp \, \mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}^{\text{boundary}}, \tag{7.2}$$

where $\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}$, $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \in \mathbb{R}^{m_{\boldsymbol{I}} \times m_{\boldsymbol{\theta}}}$, $\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{x}}) \in \mathbb{R}^{m_{\boldsymbol{I}} \times 1}$ and $\boldsymbol{v}_\perp(\boldsymbol{p}_K) \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times 1}$. We note that, $\boldsymbol{v}_\perp(\boldsymbol{p}_K)$ 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. (7.2) such as $\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}$ and $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}}$ 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 will discuss next.

Image-Loss Gradients as Path Integrals

As an important application of differentiable rendering, many inverse-rendering problems are formulated as finding scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ minimizing some (scalar-valued) loss $\mathcal{L}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{I}(\boldsymbol{\theta})). \tag{7.3}$$

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. (7.3) requires computing gradients of the loss $\mathcal{L}$ with respect to the scene parameters $\boldsymbol{\theta}$. Based on the chain rule, it holds that

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = (\partial_{\boldsymbol{I}}\mathcal{L})\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}, \tag{7.4}$$

where $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} \in \mathbb{R}^{1\times m_{\boldsymbol{\theta}}}$ and $\partial_{\boldsymbol{I}}\mathcal{L} := \frac{\partial\mathcal{L}}{\partial\boldsymbol{I}} \in \mathbb{R}^{1\times m_{\boldsymbol{I}}}$ are row vectors, and $\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}$ is an $(m_{\boldsymbol{I}} \times m_{\boldsymbol{\theta}})$-matrix. Lastly, substituting Eq. (7.2) into Eq. (7.4) yields,

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \overbrace{\int_{\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L})\,\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}}\,\mathrm{d}\mu(\bar{\boldsymbol{p}})}^{\text{interior}} + \overbrace{\int_{\partial\hat{\Omega}} (\partial_{\boldsymbol{I}}\mathcal{L})\,\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}})\,\boldsymbol{v}(\boldsymbol{p}_K)^\top\,\mathrm{d}\dot{\mu}(\bar{\boldsymbol{p}})}^{\text{boundary}}, \tag{7.5}$$

which we term as the **differential image-loss path integral**. Given Eq. (7.5), 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 $\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}$ (or individual elements of $\boldsymbol{I}$ in a differentiable fashion).

In what follows, we examine both the *interior* and the *boundary* components of Eq. (7.5). Additionally, we will discuss Monte Carlo estimations of these terms in §7.2 and §7.3.

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

$$\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}) = \hat{\boldsymbol{f}}_0(\bar{\boldsymbol{p}})\,\hat{f}_1(\bar{\boldsymbol{p}}), \tag{7.6}$$

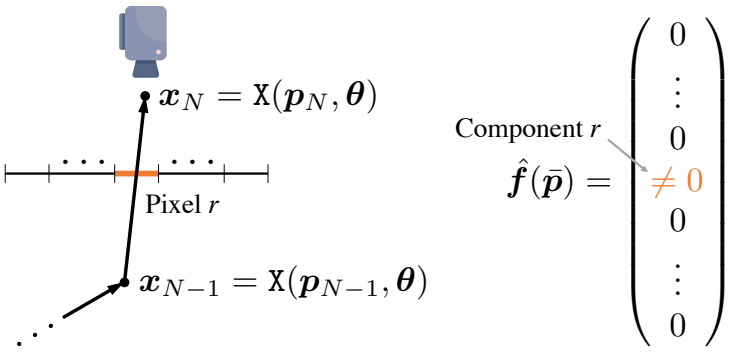


Figure 7.1: Given a material light path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_{N-1}, \boldsymbol{p}_N)$, its material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})$ is typically very **sparse** where only components corresponding to pixels that "intersect" 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.

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 responses $\boldsymbol{I}$ 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 BSDFs, geometric terms, and Jacobian determinants resulting from the material-form parameterization—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 7.1). 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. (7.5) equals

$$(\partial_{\boldsymbol{I}}\mathcal{L})\,\frac{\mathrm{d}\hat{\boldsymbol{f}}}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{p}}) = \sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}}\mathcal{L})[r]\,\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]\right), \tag{7.7}$$

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. (7.5) 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 response $\boldsymbol{I}$ are pixel intensities. It follows that the

integrand of the *boundary* integral equals

$$(\partial_{\boldsymbol{I}}\mathcal{L})\,\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}})\,\boldsymbol{v}(\boldsymbol{p}_K)^\top = \sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}})}(\partial_{\boldsymbol{I}}\mathcal{L})[r]\,(\Delta\hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}))[r]\,\boldsymbol{v}(\boldsymbol{p}_K)^\top, \tag{7.8}$$

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. (7.8) is the same as the one in Eq. (7.7), since the set of pixels to which a light transport path contribute is regardless of whether the path is ordinary or boundary.

7.2 Estimating the Interior Path Integral

Given Eq. (7.7), we can obtain the following unbiased (single-sample) Monte Carlo estimator of the *interior* component of the differential image-loss path integral (7.5):

$$\left\langle \frac{\sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}})}(\partial_{\boldsymbol{I}}\mathcal{L})[r]\,\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}}\left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]\right)}{\mathrm{pdf}_{\mathrm{i}}(\bar{\boldsymbol{p}})} \right\rangle, \tag{7.9}$$

where the material light path $\bar{\boldsymbol{p}} \in \hat{\boldsymbol{\Omega}}$, as discussed in §5.1, is sampled in a non-differentiable fashion using standard techniques (such as unidirectional path tracing); $\mathrm{pdf}_{\mathrm{i}}(\bar{\boldsymbol{p}})$ denotes the probability density for sampling $\bar{\boldsymbol{p}}$. Additionally, antithetic sampling techniques, as introduced in Chapter 6, can be applied to better handle glossy BSDFs and pixel reconstruction filters. For expositional simplicity, we omit the details of these advanced path-sampling techniques in the remainder of our exposition.

In the following, we present a general method (that is not specific to any path sampling technique) for efficient evaluation of Eq. (7.9) in §7.2.1, and discuss how this method can be further optimized in unidirectional (§7.2.2) and bidirectional path tracing (§7.2.3) settings.

7.2.1 Differentiating Measurement Contributions

Once a material light path $\bar{\boldsymbol{p}}$ is drawn, the estimation of the *interior* component of Eq. (7.5) 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. (7.9) 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 $\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \boldsymbol{\theta})$ for $0 \leq n \leq N$). The material measurement contribution defined in Eq. (3.10) can be rewritten as

$$\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r] = \left[\prod_{n=0}^{N} \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})\right] \left[\prod_{n=0}^{N-1} G(\boldsymbol{x}_n \leftrightarrow \boldsymbol{x}_{n+1})\right], \tag{7.10}$$

where G is the generalized geometric term defined in Eq. (2.28). Additionally,

$$\hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1}) := f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})\, J(\boldsymbol{p}_n), \tag{7.11}$$

where J defined in Eq. (3.11) is the Jacobian determinant resulting from the material-form parameterization (i.e., the change of variable from $\boldsymbol{x}_n$ to $\boldsymbol{p}_n$); and $f_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ defined in (2.27) captures the contribution of the vertex $\boldsymbol{x}_n$. To be specific,

- 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 vari-

Algorithm 4: Efficient differentiation of material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$ in Eq. (7.7) with respect to scene parameters $\boldsymbol{\theta}$

`ComputerMeasurementContribution(`$\boldsymbol{\theta}$, $\bar{\boldsymbol{p}}$, r`)`

Input: Scene parameters $\boldsymbol{\theta}$, a material path $\bar{\boldsymbol{p}} = (\boldsymbol{p}_0, \ldots, \boldsymbol{p}_N)$, and a pixel index r

Output: $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$ and its gradient ${\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]}/{\mathrm{d}\boldsymbol{\theta}}$

begin

`/* Forward pass (layer 1) */`

$\boldsymbol{x}_n = \mathtt{X}(\boldsymbol{p}_n, \boldsymbol{\theta})$ for each $0 \leq n \leq N$;

`/* Forward pass (layer 2) */`

$g_n = \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ for each $0 \leq n \leq N$;

$g_n = G(\boldsymbol{x}_{n-N-1} \leftrightarrow \boldsymbol{x}_{n-N})$ for each $N < n \leq 2N$;

`/* Forward pass (layer 3) */`

$\hat{f} = \prod_{n=0}^{2N} g_n$;

`/* Backward pass (layer 2): compute` $\mathrm{d}g_n := {\mathrm{d}\hat{f}}/{\mathrm{d}g_n}$ `*/`

$\mathrm{d}g_n = \frac{\hat{f}}{g_n}$ for each $0 \leq n \leq 2N$; // $\frac{\mathrm{d}\hat{f}}{\mathrm{d}g_n} = \prod_{n' \neq n} g_{n'} = \frac{\hat{f}}{g_n}$

`/* Backward pass (layer 1): compute` $\mathrm{d}\boldsymbol{x}_n := {\mathrm{d}\hat{f}}/{\mathrm{d}\boldsymbol{x}_n}$ `*/`

foreach $0 \leq n \leq N$ **do**

  $(\mathrm{d}\boldsymbol{x}_{n-1}, \mathrm{d}\boldsymbol{x}_n, \mathrm{d}\boldsymbol{x}_{n+1}) \mathrel{+}= \mathrm{d}g_n \frac{\partial \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_{n-1}, \boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$;

end

foreach $0 \leq n < N$ **do**

  $(\mathrm{d}\boldsymbol{x}_n, \mathrm{d}\boldsymbol{x}_{n+1}) \mathrel{+}= \mathrm{d}g_{N+1+n} \frac{\partial G(\boldsymbol{x}_n \leftrightarrow \boldsymbol{x}_{n+1})}{\partial(\boldsymbol{x}_n, \boldsymbol{x}_{n+1})}$;

end

`/* Backward pass (layer 0) compute` $\mathrm{d}\boldsymbol{\theta} := {\mathrm{d}\hat{f}}/{\mathrm{d}\boldsymbol{\theta}}$ `*/`

$\mathrm{d}\boldsymbol{\theta} = \sum_{n=0}^{N} \mathrm{d}\boldsymbol{x}_n \frac{\partial \mathtt{X}(\boldsymbol{p}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$;

return $(\hat{f}, \mathrm{d}\boldsymbol{\theta})$;

end

able).

Evaluating Eq. (7.9) requires computing the gradient of Eq. (7.10) with respect to the scene parameters $\boldsymbol{\theta}$. Since Eq. (7.10) 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. (7.10) will involve a great number of computations that require a large computation graph to represent. As observed in prior work [61], this can be problematic for storage (e.g., precluding GPU implementation) and performance.

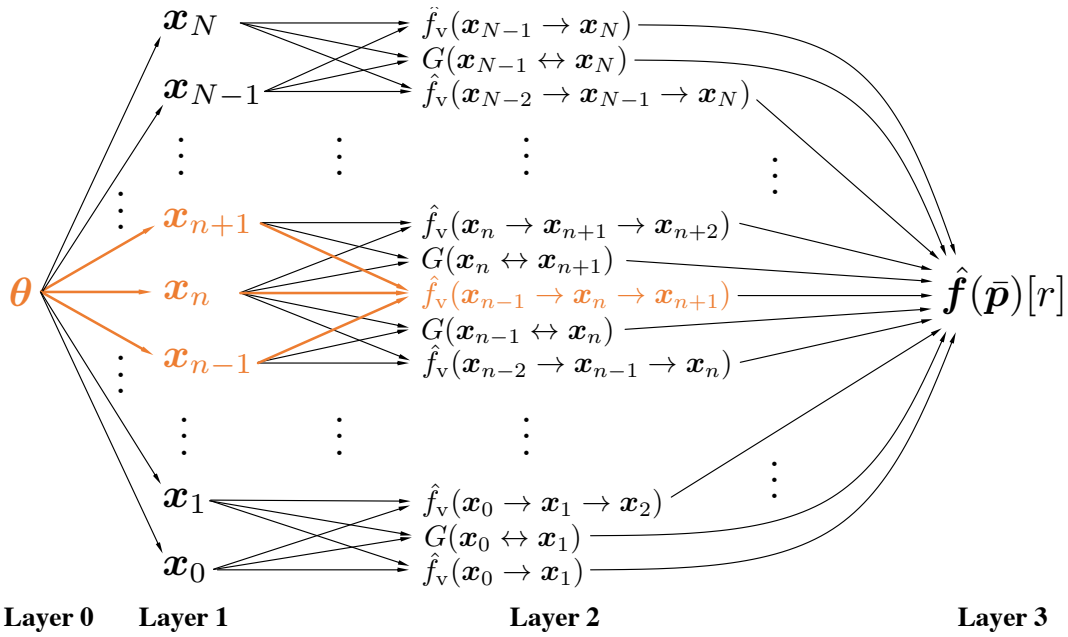


Figure 7.2: A layered **computation graph** for evaluating the material measurement contribution $\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]$. All terms on which $\hat{f}_\mathrm{v}(\boldsymbol{x}_{n-1} \to \boldsymbol{x}_n \to \boldsymbol{x}_{n+1})$ depends are highlighted in orange. We omitted the vertices $\boldsymbol{p}_n$ of the material path $\bar{\boldsymbol{p}}$ as they are independent of the scene parameters $\boldsymbol{\theta}$.

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. (7.10) can be evaluated in a largely *independent* fashion—even if the parameters $\boldsymbol{\theta}$ control scene geometry. This is thanks to the material-form parameterization: the gradient $\frac{\mathrm{d}\boldsymbol{x}_n}{\mathrm{d}\boldsymbol{\theta}}$ of each path vertex $\boldsymbol{x}_n$ can be computed independently by differentiating the mapping $\mathtt{X}(\cdot, \boldsymbol{\theta})$. That is,

$$\frac{\mathrm{d}\boldsymbol{x}_n}{\mathrm{d}\boldsymbol{\theta}} = \frac{\partial \mathtt{X}(\boldsymbol{p}_n, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \qquad \text{(for all } 0 \leq n \leq N\text{)}. \tag{7.12}$$

Figure 7.2 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 $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}})[r]}{\mathrm{d}\boldsymbol{\theta}}$ by traversing the computation graph in a layer-by-layer fashion.

As shown in Algorithm 4, 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 $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}[r])}{\mathrm{d}\boldsymbol{\theta}}$. 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 $\frac{\partial \mathtt{X}(\boldsymbol{p}_n,\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ from Lines 9, 12 and 14, respectively, can be computed efficiently using standard reverse-mode automatic differentiation.

In what follows, we discuss how Algorithm 4 can be further optimized for unidirectional and bidirectional path tracing – two widely adopted path sampling methods.

7.2.2 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 $0 < n \leq N$, connecting $\boldsymbol{p}_n^{\mathrm{D}}$ and $\boldsymbol{p}_n^{\mathrm{S}}$ produces a full light transport path

$$\bar{\boldsymbol{p}}_n = (\boldsymbol{p}_n^{\mathrm{S}}, \boldsymbol{p}_n^{\mathrm{D}}, \boldsymbol{p}_{n-1}^{\mathrm{D}}, \ldots, \boldsymbol{p}_0^{\mathrm{D}}), \tag{7.13}$$

as illustrated in Figure 7.3.[1] Then, the *interior* component of Eq. (7.5) can be estimated by summing Eq. (7.9) over all $\bar{\boldsymbol{p}}_n$:

$$\left\langle \sum_{n=1}^{N} \frac{\sum_{r\in\mathrm{nz}(\bar{\boldsymbol{p}}_n)} (\partial_{\boldsymbol{I}}\mathcal{L})[r] \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_n)[r] \right)}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)} \right\rangle. \tag{7.14}$$

Although this expression can be evaluated by applying Algorithm 4 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 $0 < n \leq N$, we rearrange the

[1] 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.

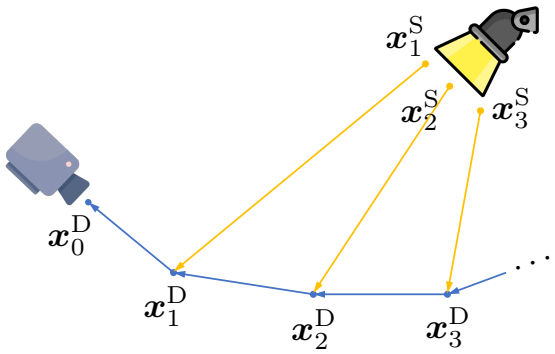


Figure 7.3: With next-event estimation (NEE), a unidirectional path tracer effectively constructs a set of light paths that share one detector subpath $(\boldsymbol{x}_0^{\mathrm{D}}, \boldsymbol{x}_1^{\mathrm{D}}, \ldots)$ shown in blue. The arrows in this figure illustrates the flow of light. We present a technique to efficiently compute and differentiate the measurement contribution of all paths by factoring out common terms.

terms of Eq. (7.14) and obtain:

$$\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}}^{\mathrm{D}})} (\partial_{\boldsymbol{I}} \mathcal{L})[r] \frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}\boldsymbol{\theta}} \quad \text{where} \quad h_{\mathrm{NEE}} := \sum_{n=1}^{N} \frac{\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_n)[r]}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)}, \tag{7.15}$$

with $\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)$ treated as "detached" (i.e., independent of $\boldsymbol{\theta}$). To efficiently compute Eq. (7.15), similar to how unidirectional path tracing is implemented for forward rendering, we factor out the common terms in the inner summation of Eq. (7.15). Let

$$h_n^{\mathrm{D}} := \hat{f}_{\mathrm{v}}(\boldsymbol{x}_{n+1}^{\mathrm{D}} \to \boldsymbol{x}_n^{\mathrm{D}} \to \boldsymbol{x}_{n-1}^{\mathrm{D}})\, G(\boldsymbol{x}_{n+1}^{\mathrm{D}} \leftrightarrow \boldsymbol{x}_n^{\mathrm{D}}), \tag{7.16}$$

$$h_n^{\mathrm{S}} := \frac{\hat{f}_{\mathrm{v}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_n^{\mathrm{D}} \to \boldsymbol{x}_{n-1}^{\mathrm{D}})\, G(\boldsymbol{x}_0^{\mathrm{S}} \leftrightarrow \boldsymbol{x}_n^{\mathrm{D}})\, \hat{L}_{\mathrm{e}}(\boldsymbol{x}_0^{\mathrm{S}} \to \boldsymbol{x}_0^{\mathrm{D}})}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)}, \tag{7.17}$$

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 $0 < n \leq N$,

$$\frac{\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_n)[r]}{\mathrm{pdf}_{\mathrm{NEE}}(\bar{\boldsymbol{p}}_n)} = \left(\prod_{n'=0}^{n} h_{n'}^{\mathrm{D}} \right) h_n^{\mathrm{S}}. \tag{7.18}$$

It follows that

$$h_{\mathrm{NEE}} = h_0^{\mathrm{D}} \left(h_1^{\mathrm{S}} + h_1^{\mathrm{D}} \left(h_2^{\mathrm{S}} + h_2^{\mathrm{D}} \left(h_3^{\mathrm{S}} + h_3^{\mathrm{D}}(\ldots) \right) \right) \right), \tag{7.19}$$

which can be differentiated in a layered fashion using a process similar to Algorithm 4. Specifically, in the forward pass, we first obtain the path vertices $\boldsymbol{x}_n^{\mathrm{D}} = \mathtt{X}(\boldsymbol{p}_n^{\mathrm{D}}, \boldsymbol{\theta})$ and $\boldsymbol{x}_n^{\mathrm{S}} = \mathtt{X}(\boldsymbol{p}_n^{\mathrm{S}}, \boldsymbol{\theta})$ 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 $\mathrm{d}h_n^{\mathrm{D}} := \frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}h_n^{\mathrm{D}}}$ and $\mathrm{d}h_n^{\mathrm{S}} := \frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}h_n^{\mathrm{S}}}$ (layer 2) by differentiating Eq. (7.19). Then, we evaluate $\frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}\boldsymbol{x}_n^{\mathrm{D}}}$ and $\frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}\boldsymbol{x}_n^{\mathrm{S}}}$ (layer 1) followed with the gradient $\frac{\mathrm{d}h_{\mathrm{NEE}}}{\mathrm{d}\boldsymbol{\theta}}$ (layer 0) that can be used to estimate the *interior* integral via Eq. (7.15).

7.2.3 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. (7.5) can be estimated using

$$\left\langle \sum_{s,t} \sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}}_{s,t})} w_{s,t}(\bar{\boldsymbol{p}}_{s,t}) \frac{(\partial_{\boldsymbol{I}} \mathcal{L})[r] \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} \left(\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_{s,t})[r] \right)}{\mathrm{pdf}_{s,t}(\bar{\boldsymbol{p}}_{s,t})} \right\rangle, \tag{7.20}$$

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. (7.20) 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 4. Lastly, we reuse these terms to evaluate the gradient $\frac{\mathrm{d}\hat{\boldsymbol{f}}(\bar{\boldsymbol{p}}_{s,t})[r]}{\mathrm{d}\boldsymbol{\theta}}$ for all s and t while avoiding duplicate computation/differentiation.

7.2.4 Relation with Prior Works

Our technique presented above is closely related to some recent works [61, 84], which formulate image-loss gradients as solutions of an adjoint transport problem. We will show that some key results in these work can be considered a specific realization of Eq. (7.5). 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 $\boldsymbol{W}_{\mathrm{e}}(\bar{\boldsymbol{x}}) \in \mathbb{R}^{m_{\boldsymbol{I}} \times 1}$ 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. (7.5) simplifies to

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \int_{\boldsymbol{\Omega}} \underbrace{(\partial_{\boldsymbol{I}}\mathcal{L})\, \boldsymbol{W}_{\mathrm{e}}(\bar{\boldsymbol{x}})}_{=:\, \boldsymbol{A}_{\mathrm{e}}(\bar{\boldsymbol{x}})} \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}})\, \mathrm{d}\mu(\bar{\boldsymbol{x}}). \tag{7.21}$$

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. (7.21) as

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}} = \int_{\mathcal{M}^2} \boldsymbol{A}_{\mathrm{e}}(\boldsymbol{x}_{N-1} \to \boldsymbol{x}_N) \left[\int_{\boldsymbol{\Omega}} \frac{\mathrm{d}f_1}{\mathrm{d}\boldsymbol{\theta}}(\bar{\boldsymbol{x}})\, \mathrm{d}\mu(\bar{\boldsymbol{x}}_-) \right] \mathrm{d}A(\boldsymbol{x}_{N-1})\, \mathrm{d}A(\boldsymbol{x}_N), \tag{7.22}$$

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. (7.22) is equivalent to a key result in §3.3 of the radiative backpropagation work [61].

Moreover, a key idea in path replay backpropagation [84] is to reuse light transport paths when recursively expanding 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,

$$\int_{\hat{\boldsymbol{\Omega}}} \frac{\mathrm{d}\hat{f}(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \,\mathrm{d}\mu(\bar{\boldsymbol{p}}) = \int_{\hat{\boldsymbol{\Omega}}} \left(\sum_n \frac{\mathrm{d}g_n(\bar{\boldsymbol{p}})}{\mathrm{d}\boldsymbol{\theta}} \prod_{n' \neq n} g_{n'}(\bar{\boldsymbol{p}}) \right) \mathrm{d}\mu(\bar{\boldsymbol{p}}). \tag{7.23}$$

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 4).

7.3 Estimating the Boundary Path Integral

Eq. (7.8) induces an unbiased (single-sample) Monte Carlo estimator of the *boundary* component of the differential image-loss path integral (7.5) as

$$\left\langle \frac{\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}} \mathcal{L})[r] \, (\Delta \hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}))[r] \, \boldsymbol{v}(\boldsymbol{p}_K)^\top}{\mathrm{pdf}_{\mathrm{b}}(\bar{\boldsymbol{p}})} \right\rangle, \tag{7.24}$$

where the boundary light path $\bar{\boldsymbol{p}} \in \partial\hat{\boldsymbol{\Omega}}$ can be sampled in a multi-directional fashion (starting with the boundary segment) as discussed in §5.2; and $\mathrm{pdf}_{\mathrm{b}}(\bar{\boldsymbol{p}})$ denotes the probability density for sampling $\bar{\boldsymbol{p}}$.

With the boundary path $\bar{\boldsymbol{p}}$ sampled, evaluating the numerator of Eq. (7.24) is, in fact, relatively inexpensive. This is because, with $\partial_{\boldsymbol{I}} \mathcal{L}$ 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 (§7.2) that requires differentiating the full material measurement contribution. Specifically, it holds that

$$\boldsymbol{v}(\boldsymbol{p}_K) = \left(\frac{\mathrm{d}\boldsymbol{p}_K}{\mathrm{d}\boldsymbol{\theta}}\right)^\top \boldsymbol{n}(\boldsymbol{p}_K), \tag{7.25}$$

where $\boldsymbol{n}(\boldsymbol{p}_K)$ is a three-dimensional (column) vector representing the unit normal of the visibility boundary at $\boldsymbol{p}_K$. Further, the calculation of the spatial point $\boldsymbol{p}_K$—which generally depends on the parameters $\boldsymbol{\theta}$—has been presented in §5.2.5 via, for example, Eqs. (5.28) and (5.32). To obtain $\boldsymbol{v}(\boldsymbol{p}_K)$ using reverse-mode automatic differentiation, we let

$$V(\boldsymbol{p}_K) := \boldsymbol{p}_K^\top \boldsymbol{n}(\boldsymbol{p}_K). \tag{7.26}$$

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

$$\boldsymbol{v}(\boldsymbol{p}_K) = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\theta}} V(\boldsymbol{p}_K). \tag{7.27}$$

It follows that the scalar-valued expression

$$\boldsymbol{p}_K^\top \left(\boldsymbol{n}(\boldsymbol{p}_K) \frac{\sum_{r \in \mathrm{nz}(\bar{\boldsymbol{p}})} (\partial_{\boldsymbol{I}} \mathcal{L})[r] \, (\Delta \hat{\boldsymbol{f}}_K(\bar{\boldsymbol{p}}))[r]}{\mathrm{pdf}_\mathrm{b}(\bar{\boldsymbol{p}})}\right), \tag{7.28}$$

with all the terms expect the first (i.e., $\boldsymbol{p}_K^\top$) fixed, is an "anti-gradient" of Eq. (7.24). Thus, by applying reverse-mode automatic differentiation (i.e., backward) to the result of this expression, we can accumulate the contribution given by Eq. (7.24) to the final gradient $\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}$.

Chapter 8

Application: Physics-Based Inverse Rendering

As a main application of differentiable rendering, *inverse rendering* (aka. *analysis by synthesis*) is concerned with recovering a set of scene parameters $\boldsymbol{\theta} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ (such as material or geometric properties of objects in the scene) from one or multiple input images $\tilde{\boldsymbol{I}} \in \mathbb{R}^{m_{\boldsymbol{I}}}$. This process is normally be formulated as the optimization of some predetermined (scalar-valued) loss $\mathcal{L}_{\text{tot}}$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}_{\text{tot}}(\boldsymbol{I}(\boldsymbol{\theta}), \boldsymbol{\theta}; \tilde{\boldsymbol{I}}) \tag{8.1}$$

where $\boldsymbol{I}(\boldsymbol{\theta}) \in \mathbb{R}^{m_{\boldsymbol{I}}}$ denotes images generated with parameters $\boldsymbol{\theta}$ (provided by *forward rendering*), and each component of $\tilde{\boldsymbol{I}}$ and $\boldsymbol{I}(\boldsymbol{\theta})$ stores the intensity of an image pixel.

Loss function. The loss $\mathcal{L}_{\text{tot}}$ in Eq. (8.1) is a crucial ingredient for solving inverse-rendering problems and typically formulated as the sum of an *image loss* $\mathcal{L}$ and a *regularization loss* $\mathcal{L}_{\text{reg}}$

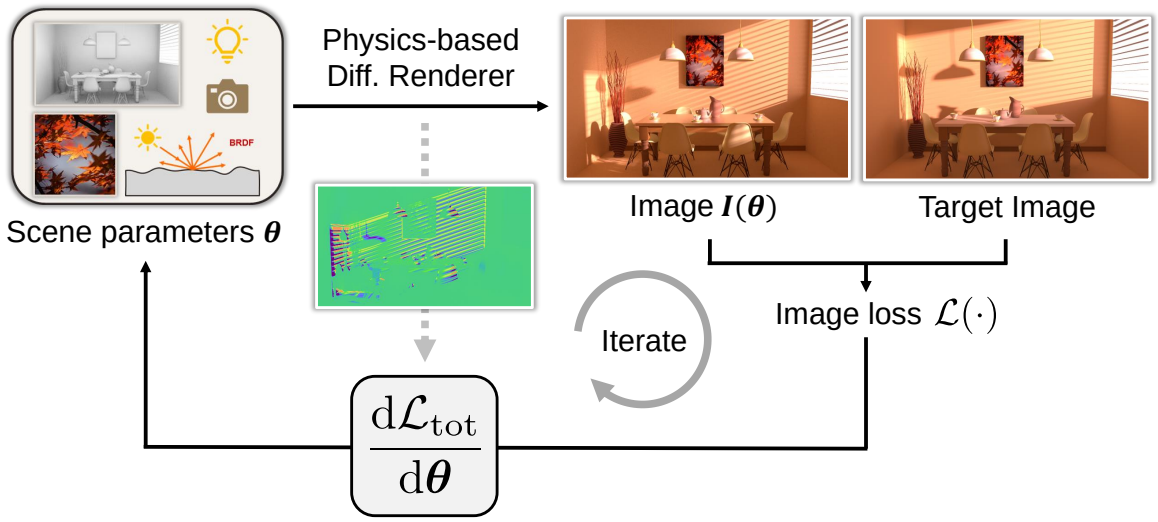


Figure 8.1: The typical optimization pipeline for gradient-based inverse rendering applications. In this iterative process, the derivatives generated by the differentiable renderer is the key for updating the scene parameters $\boldsymbol{\theta}$.

weighted with $\lambda \in \mathbb{R}_{>0}$:

$$\mathcal{L}_{\mathrm{tot}}(\boldsymbol{I}(\boldsymbol{\theta}), \boldsymbol{\theta}; \tilde{\boldsymbol{I}}) := \mathcal{L}(\boldsymbol{I}(\boldsymbol{\theta}), \tilde{\boldsymbol{I}}) + \lambda\, \mathcal{L}_{\mathrm{reg}}(\boldsymbol{\theta}). \tag{8.2}$$

In practice, the image loss $\mathcal{L}$ can be simple L_1 or L_2 differences (between $\boldsymbol{I}(\boldsymbol{\theta})$ and $\tilde{\boldsymbol{I}}$), or more sophisticated neural metrics [74, 35, 56, 72]. The regularization loss $\mathcal{L}_{\mathrm{reg}}$ can encode simple smoothness conditions (e.g., using image or mesh Laplacian) or complex data-driven priors.

Solving inverse rendering problems. The inverse-rendering optimization in Eq. (8.1) is typically solved using gradient-based methods. For example, when using (stochastic) gradient descent, one starts with some initial guess $\boldsymbol{\theta}^{(0)}$ that is then updated iteratively for $t = 0, 1, \ldots$ via:

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma^{(t)} \left(\frac{\mathrm{d}\mathcal{L}_{\mathrm{tot}}}{\mathrm{d}\boldsymbol{\theta}}\bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} \right), \tag{8.3}$$

where $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{m_{\boldsymbol{\theta}}}$ indicates the parameter values at iteration t, and $\gamma^{(t)} \in \mathbb{R}$ denotes the *learning rate* (or step size) for that iteration. Further, more advanced optimization methods like Adam [42] replace the scalar step sizes $\gamma^{(t)}$ with *preconditioning matrices* $\boldsymbol{M}^{(t)} \in \mathbb{R}^{m_{\boldsymbol{\theta}} \times m_{\boldsymbol{\theta}}}$ computed based on the parameter values $\boldsymbol{\theta}^{(t')}$ for $t' \leq t$.

Applying the iterative update outlined in Eq. (8.3) requires computing the gradient $\frac{\mathrm{d}\mathcal{L}_{\text{tot}}}{\mathrm{d}\boldsymbol{\theta}}$ at $\boldsymbol{\theta} = \boldsymbol{\theta}^{(t)}$:

$$\left.\frac{\mathrm{d}\mathcal{L}_{\text{tot}}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} = \left.\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} + \left.\frac{\mathrm{d}\mathcal{L}_{\text{reg}}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}, \tag{8.4}$$

where, according to the chain rule, it holds that

$$\left.\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}} = \left(\left.\frac{\partial\mathcal{L}}{\partial\boldsymbol{I}}\right|_{\boldsymbol{I}=\boldsymbol{I}(\boldsymbol{\theta}^{(t)})}\right)\left(\left.\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}\right). \tag{8.5}$$

In Eqs. (8.4) and (8.5), $\left.\frac{\mathrm{d}\mathcal{L}_{\text{reg}}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$ and $\left.\frac{\partial\mathcal{L}}{\partial\boldsymbol{I}}\right|_{\boldsymbol{I}=\boldsymbol{I}(\boldsymbol{\theta}^{(t)})}$ are obtained, respectively, using differentiable evaluations of the regularization and image losses $\mathcal{L}_{\text{reg}}$ and $\mathcal{L}$. The image gradient $\left.\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$, on the other hand, requires differentiable rendering—the main focus of this dissertation. We show an overview of the inverse-rendering optimization process in Figure 8.1.

Moreover, as discussed in Chapter 7, the image-loss gradient $\left.\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$ in Eq. (8.4) can be estimated efficiently using differential image-loss path integrals (7.5) by taking as input $\left.\frac{\partial\mathcal{L}}{\partial\boldsymbol{I}}\right|_{\boldsymbol{I}=\boldsymbol{I}(\boldsymbol{\theta}^{(t)})}$. This avoid storing the full image gradient $\left.\frac{\mathrm{d}\boldsymbol{I}}{\mathrm{d}\boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}^{(t)}}$ (or the corresponding global computation graph).

In what follows, to further demonstrate the practical usefulness of our path-space differentiable rendering techniques introduced in the earlier chapters, we provide several synthetic inverse-rendering examples in §8.1 and §8.2 using gradients estimated with these methods.

8.1 Inverse Rendering Comparisons

We now demonstrate the effectiveness of our techniques by comparing inverse-rendering results produced by our methods and a variety of baselines.

All our results are generated using an educational CPU-based implementation that has been released with our previous publications [94, 96, 93].

Configurations. For all comparisons in this section, we set the image loss $\mathcal{L}$ as the root-mean-square error (RMSE) between one target image and the corresponding rendering and apply no additional regularization (i.e., $\mathcal{L}_{\text{reg}} \equiv 0$). Additionally, we use the Adam method [42] to solve the optimizations. To ensure fairness, we use identical initial states $\boldsymbol{\theta}^{(0)}$ and learning rates for each comparison.

8.1.1 Comparisons with Non-Path-Space Methods

As demonstrated in §5.3.2, our Monte Carlo estimators (**I.1**, **I.2**) based on the formulation of differential path integrals scale well to scenes with complex geometries. Additionally, our bidirectional estimator (**I.2**) is capable of efficiently handling complex light-transport effects like caustics. In the following, we compare inverse-rendering results with gradients estimated with our methods and several previous (non-path-space) methods.

Complex geometry. We first demonstrate the effectiveness of our technique using examples with complex geometries and compare inverse-rendering results generated by Monte Carlo edge sampling [45, 95], biased reparameterization [49], and our methods.

Figure 8.2 shows an example that reuses the BRANCHES2 scene described in §5.3. We use two inverse-rendering setting with identical initial configurations but different targets where

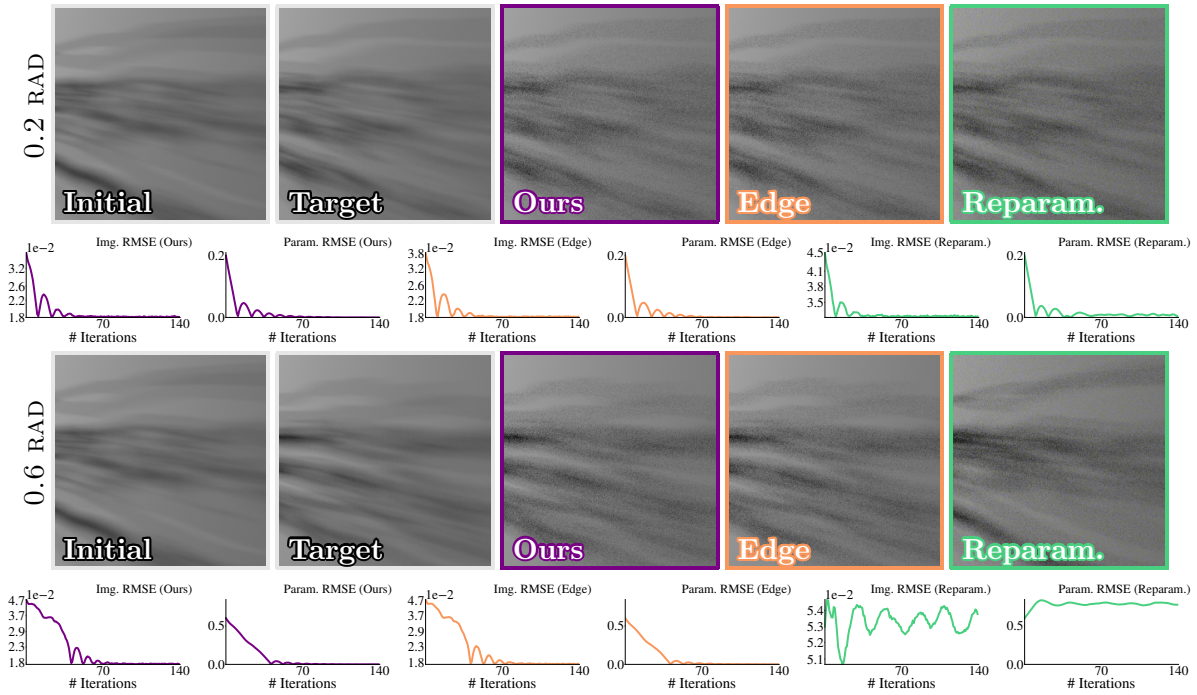


Figure 8.2: **Inverse rendering comparisons** using the BRANCHES2 scene with the object's rotation angle being optimized. All methods are configured to have *equal sample* count per pixel. We show rendered images produced by each method after the last iteration with ours marked in purple, edge sampling in orange, and reparameterization in green. The image and parameter RMSE plots are color-coded the same way, and the latter is not used for optimization.

the object is rotated, respectively, by 0.2 and 0.6 radian (from the initial). Under the first setting, all methods including the biased reparameterization method, manage to converge to the correct answer. Under the second setting, on the other hand, the reparameterization approach fails to converge properly due to its high bias. Under both settings, our method runs significantly faster than edge sampling while producing much cleaner derivatives.

In Figure 8.3, we show inverse-rendering processes of the PUFFER-BALL scene with the light sources sizes being optimized. Details of this scene can be found in §5.3. Due to the very high face count, edge sampling produces too much noise for the optimization to converge properly. Our technique again produces clean and unbiased derivative estimates, allowing the optimization to converge easily.

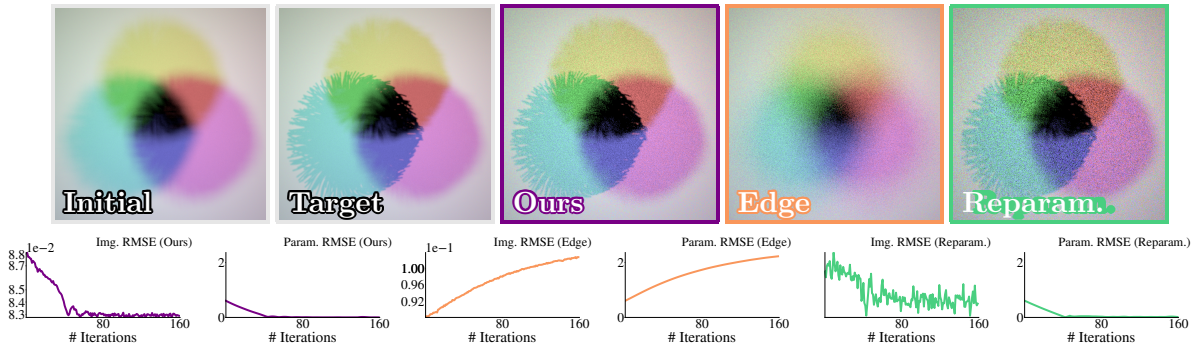


Figure 8.3: **Inverse rendering comparisons** using the PUFFER-BALL scene with the light source sizes being optimized. All methods are configured to use equal sample count per pixel, and the visualization scheme follows that of Figure 8.2.

The experiments are conducted on a workstation equipped with an octa-core Intel i7-7820X CPU and an Nvidia Titan RTX graphics card. The performance statistics for therse two *equal-sample* comparisons are included in Table 8.1.

Complex light-transport effects. To further demonstrate the advantage of our bidirectional estimator (**I.2**), we use the VEACH-EGG2 scene, whose derivative comparisons are provided in Figure 5.7, with a new inverse-rendering setup where the position of the spot light and the refractive index of the glass egg are optimized jointly. We compare the performance of our unidirectional and bidirectional methods as well as edge sampling [45, 95]. We adjust the sample counts so that each iteration takes roughly *equal time* for all methods. We do not include the reparameterization method [49] for this comparison as its implementation does not support derivatives with respect to refractive indices. As shown in Figure 8.4, gradients

Table 8.1: Performance statistics for the inverse-rendering comparisons in Figures 8.2, 8.3. The "*time*" numbers indicate average computation time (in seconds) per iteration, including the overhead (shown in parentheses) for precomputing importance-sampling grid (discussed in §5.2.4).

Scene	**# param.**	**# iter.**	**Time** (Ours)	**Time** (Edge)	**Time** (Reparam.)
BRANCHES2	1	140	0.5 (0.1)	5.7	0.3
PUFFER BALL	3	160	4.5 (2.0)	28.6	1.5

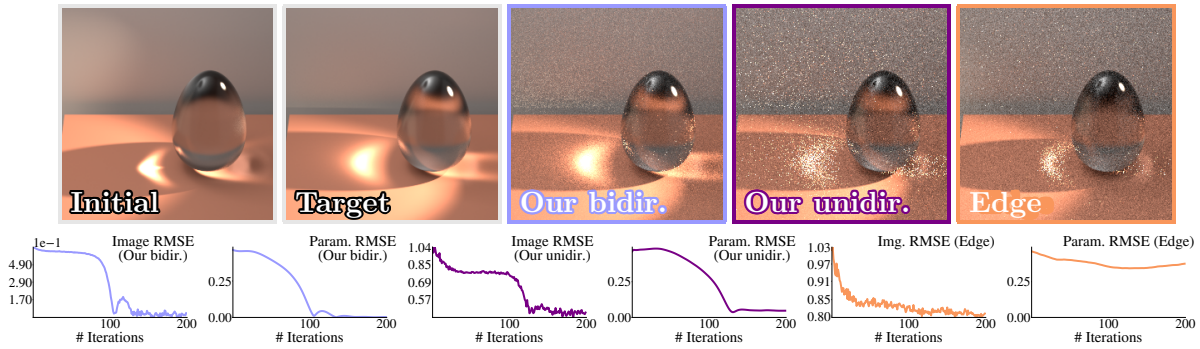


Figure 8.4: **Inverse rendering comparison** using the VEACH-EGG2 scene with the spot light's location and the glass egg's refractive index optimized jointly. The optimizations are configured so that each iteration takes *equal time* for all methods, and the visualization scheme follows that of Figure 8.2.

estimated with edge sampling are too noisy for the optimization to converge properly. Those produced by our unidirectional algorithm (**I.1**) have higher quality but are still noisy, preventing the optimization from finding to the exact solution. The bidirectional variant (**I.2**), on the other hand, produce significantly cleaner gradient estimates that allow the optimization to converge smoothly to the global optimum.

Volumetric light transport. We now provide a set of comparisons where volumetric light transport is involved in Figure 8.5. We compare the inverse-rendering performance of our method and DTRT to demonstrate the usefulness of our low-variance derivative estimates. We adjust sample counts so that each optimization iteration takes approximately *equal time*.

For the BRANCHES and BUST scenes in this figure, we use the exact scene and parameter configurations given in §5.3. For the BUMPY-SPHERE scene, we replace the point light with a small area light since DTRT only supports the latter.

For the BRANCHES scene, gradient-based optimizations driven by both methods converge correctly. Our method offers faster convergences thanks to its cleaner derivative estimates. For the BUST and BUMPY-SPHERE scenes, optimizations using gradients estimated by DTRT fail to converge due to very high variance in the estimated gradients. Our method, on the

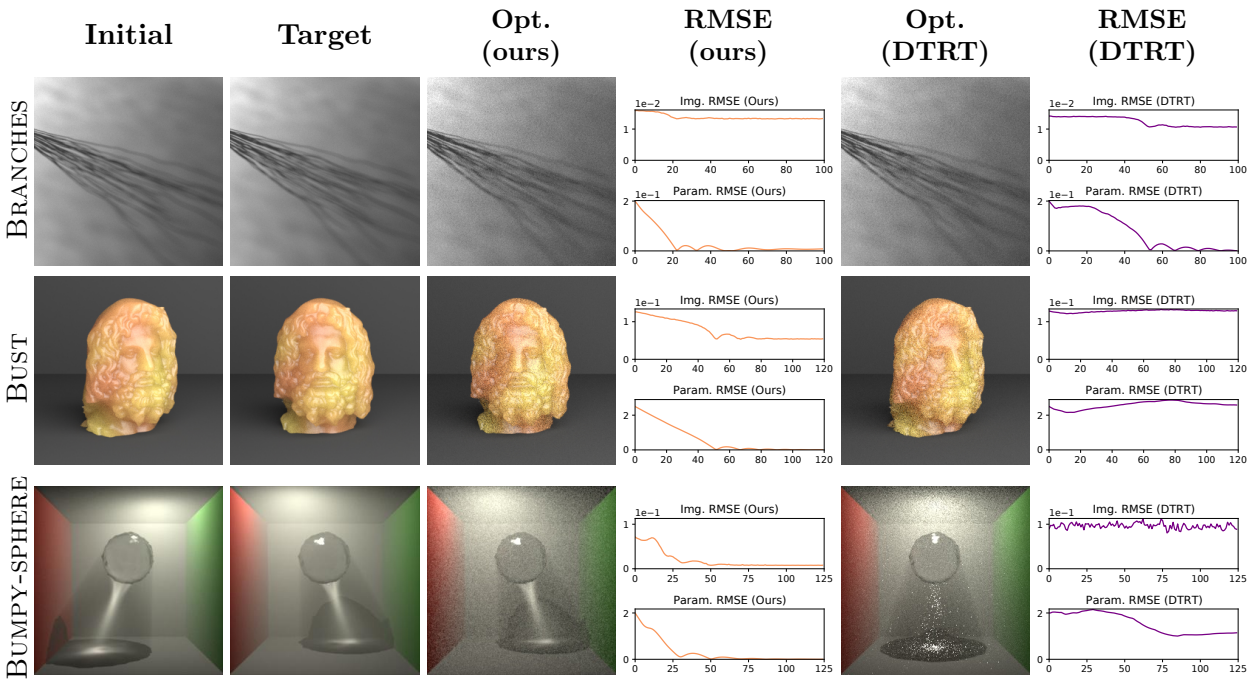


Figure 8.5: **Inverse-rendering comparisons** using gradients estimated with our technique and DTRT [95]. The sample count is adjusted so that both methods take approximately equal time per iteration.

other hand, allows smooth convergence to the correct results.

8.1.2 Antithetic Sampling for Inverse Rendering

Now, to highlight the importance of low-variance derivative estimates attributed from antithetic sampling discussed in Chapter 6, we compare the inverse rendering performance using derivatives estimated with and without this technique.

BSDF antithetic sampling. We now show inverse-rendering results with and without BSDF antithetic sampling (§6.2) using our unidirectional and bidirectional algorithms (**I.1**, **I.2**) labeled as "PS1" and "PS2", respectively, as the base methods.

Figure 8.6 uses a GLASS-BUNNY scene where a glass bunny model is rotated around the verti-

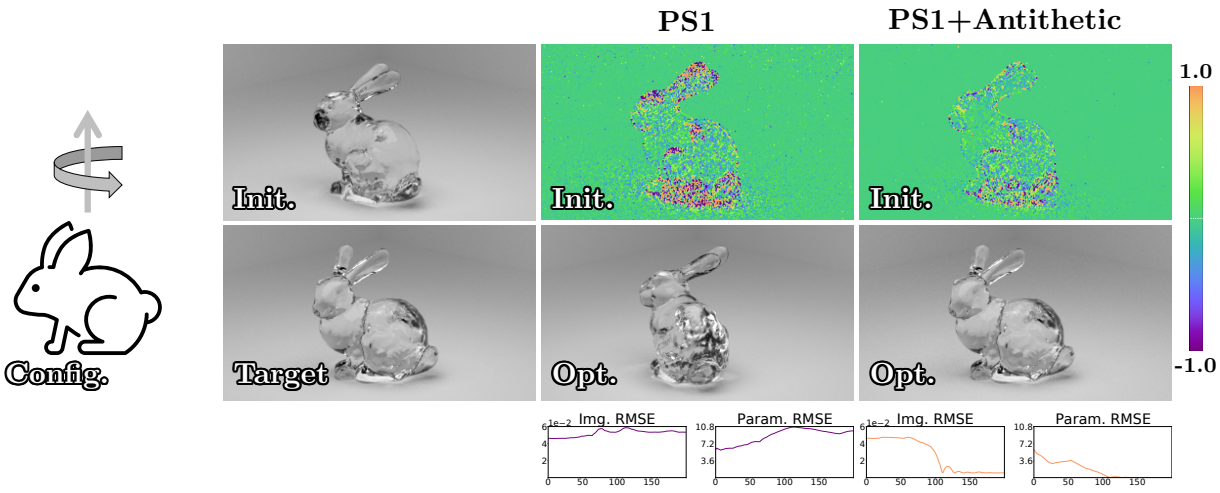


Figure 8.6: **Inverse-rendering comparison** using GLASS-BUNNY scene: We search for the rotation angle of the glass bunny to match the target image by minimizing the image RMSE (the plotted parameter RMSE information is used for evaluation only). The visualized derivatives involve both *interior* and *boundary* contributions with the latter estimated using the base method for both results.

cal axis (as illustrated in "Config."). Using our unidirectional estimator (**I.1**), the derivative images (including both *interior* and *boundary* contributions) corresponding to the initial configuration are shown in the top row. These images are generated in equal time, and the one using antithetic sampling (i.e., PS1+Antithetic) enjoys much lower noise. This reduced variance makes a significant difference in inverse rendering performance by allowing the inverse-rendering optimization to converge nicely. Without antithetic sampling, on the other hand, the optimization fails to converge.

In Figure 8.7, we show a MUG scene consisting of a small area light inside a near-specular glass mug, creating complex caustics on the surface below, as illustrated in "Config.". Given a target image with the desired caustics pattern, we solve for the position (depicted with three parameters) of the area light. We use the bidirectional path-space algorithm as the base method for this example. Without antithetic sampling, even with our bidirectional estimator (**I.2**), the derivative image remains very noisy, causing the optimization to have difficulties in converging. With antithetic sampling, on the other hand, the derivative estimates become

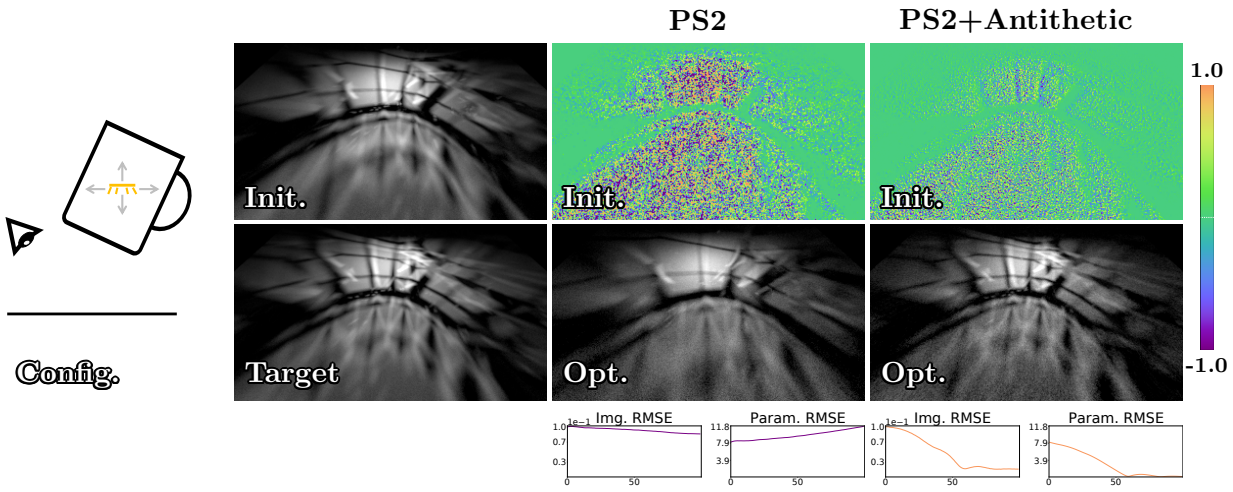


Figure 8.7: **Inverse-rendering comparison** using MUG scene: We search for the 3D location of the area light inside the glass mug to match the caustics patterns on the surface below. Similar to Figure 8.6, the visualized derivatives involve both *interior* and *boundary* contributions.

significantly cleaner, leading to much easier convergence.

Lastly, we show in Figure 8.8 an EINSTEIN scene that contains an area light with spatially varying emission displaying a distorted photo of Einstein. The emitted light is then reflected by a glossy surface before reaching the camera. Given a target reflection that is non-distorted, we solve for the shape of this surface (parameterized using 100 variables). Without antithetic sampling, our unidirectional estimator (**I.1**) fails to converge within 300 iterations. On the contrary, with antithetic sampling, the optimization successfully recovers the target geometry (as illustrated using the height maps).

Pixel-filter antithetic sampling. To demonstrate the practical effectiveness of our pixel-filter antithetic sampling with vertices reusing (§6.4.2), we compare inverse-rendering results with and without this technique.

In Figure 8.9, the VOL-BUNNY example includes a translucent bunny under area lighting. The KITTY2 result uses a configuration similar to that in Figure 6.13 with a Cornell box con-

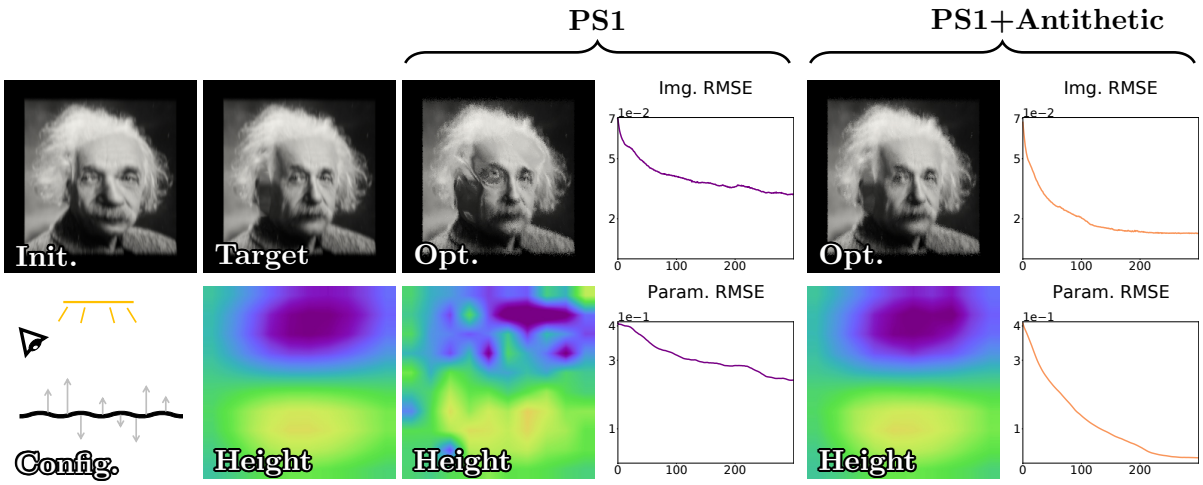


Figure 8.8: **Inverse-rendering comparison** using EINSTEIN scene: We search for the shape of the glossy reflector (parameterized using 100 variables) so that the reflection matches the target. We visualize the target and optimized reflector geometries as height maps on the bottom row.

taining a kitty and an area light facing the ceiling. We use our unidirectional and bidirectional estimators (**I.1**, **I.2**) for the two examples, respectively. For both examples, we optimize the object shapes by minimizing the image loss (using 20 target images) and use Nicolet et al.'s method [58] to update mesh vertex positions (provided the estimated gradients). The reduced variance offered by our antithetic sampling technique has allowed both optimizations to converge more quickly, resulting in reconstructed geometries with lower error (measured by Chamfer distances [6]).

8.2 Additional Inverse-Rendering Results

We now show additional inverse-rendering results generated using our new CPU-based implementation that utilize the efficient differentiation algorithms presented in Chapter 7 and the Enzyme automatic differentiation framework [55].

We show examples using gradients estimated with our unidirectional algorithm (**I.1**) in Fig-

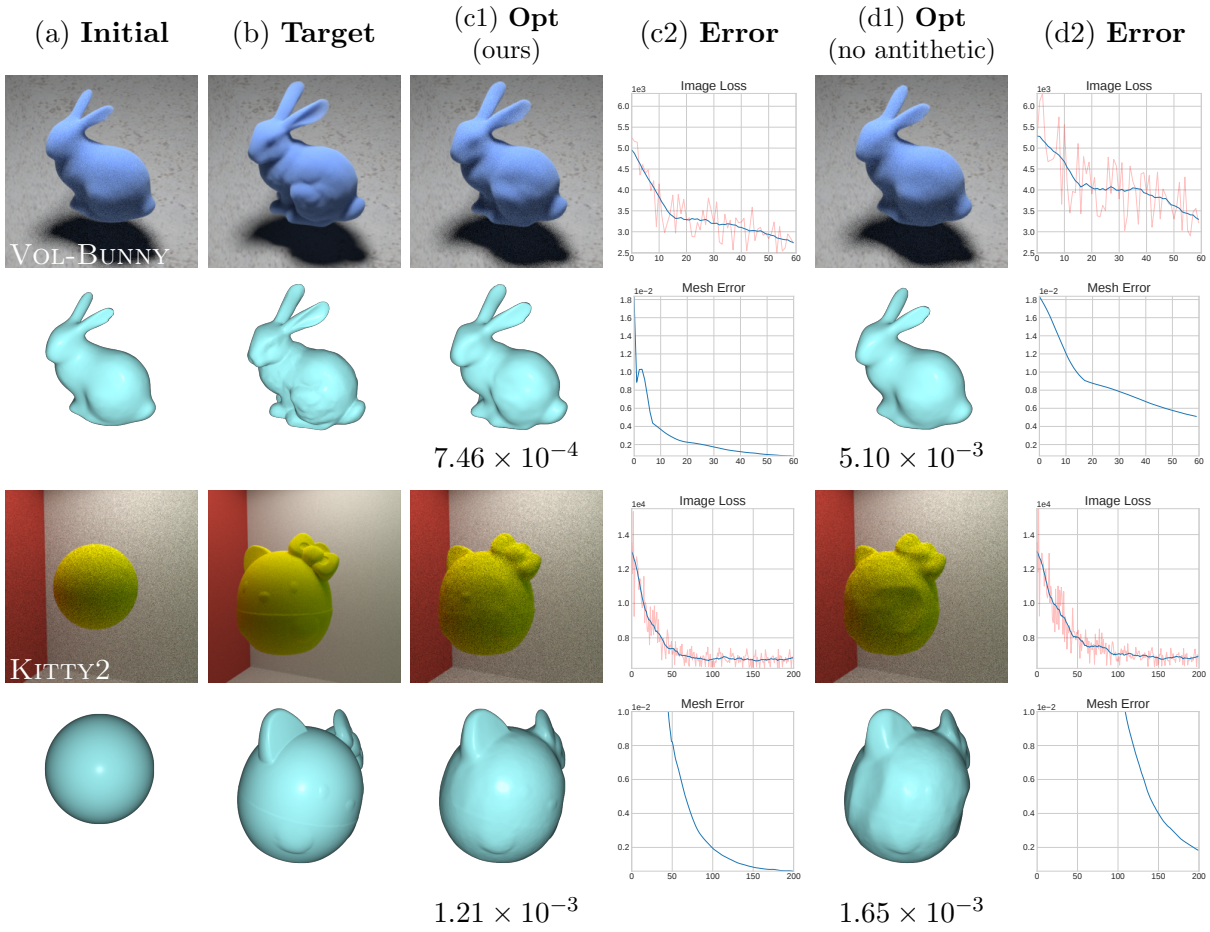


Figure 8.9: We show inverse-rendering comparisons of derivatives estimated with (c) and without (d) our **pixel-filter antithetic sampling with vertices reusing** (presented in §6.4.2). The mesh error numbers shown under visualized geometries in (c1) and (d1) as well as plotted in (c2) and (d2) indicate the Chamfer distances [6] 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).

ures 8.10 and 8.11 as well as bidirectional estimator (**I.2**) in Figure 8.12. Additionally, Table 8.2 summarizes optimization configurations and performance statistics.

In Figure 8.10, the WHITE-BUNNY scene contains a diffuse bunny inside a box. The GLOSSY-KITTY 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.

Table 8.2: 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 [58]). 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
WHITE-BUNNY	40	30,000	2	1,000	8.40s	0.38s
GLOSSY-KITTY	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
GLOSSY-BUNNY2	50	150,000	4	600	34.52s	1.75s
CAUSTICS	1	651	1	300	6.46s	0.03s

Additionally, Figure 8.11 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 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 8.10, 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 8.12 illustrates results leveraging our bidirectional estimator (**I.2**). The GLOSSY-KITTY2 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 optimize 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. This time, with a single target image, we optimize the shape of the air-water interface *without every directly viewing the surface.*

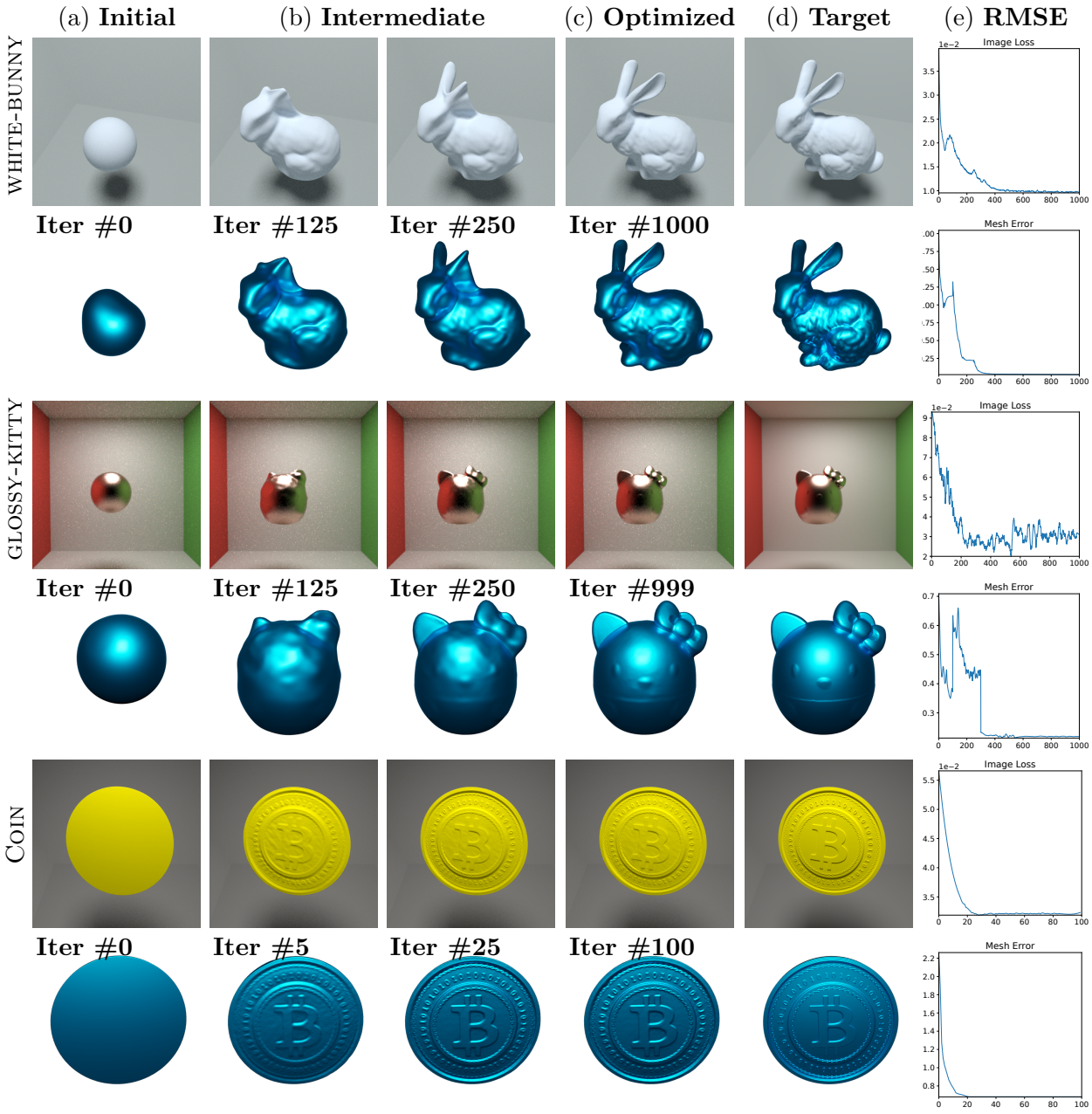


Figure 8.10: **Inverse-rendering results** obtained with gradients estimated using our unidirectional estimator (**I.1**). The mesh error plotted in column (e) captures the Chamfer distance [6] between the reconstructed and groundtruth geometries (normalized so that the GT has a unit bounding box). We use this information only for evaluation (and not for optimization).

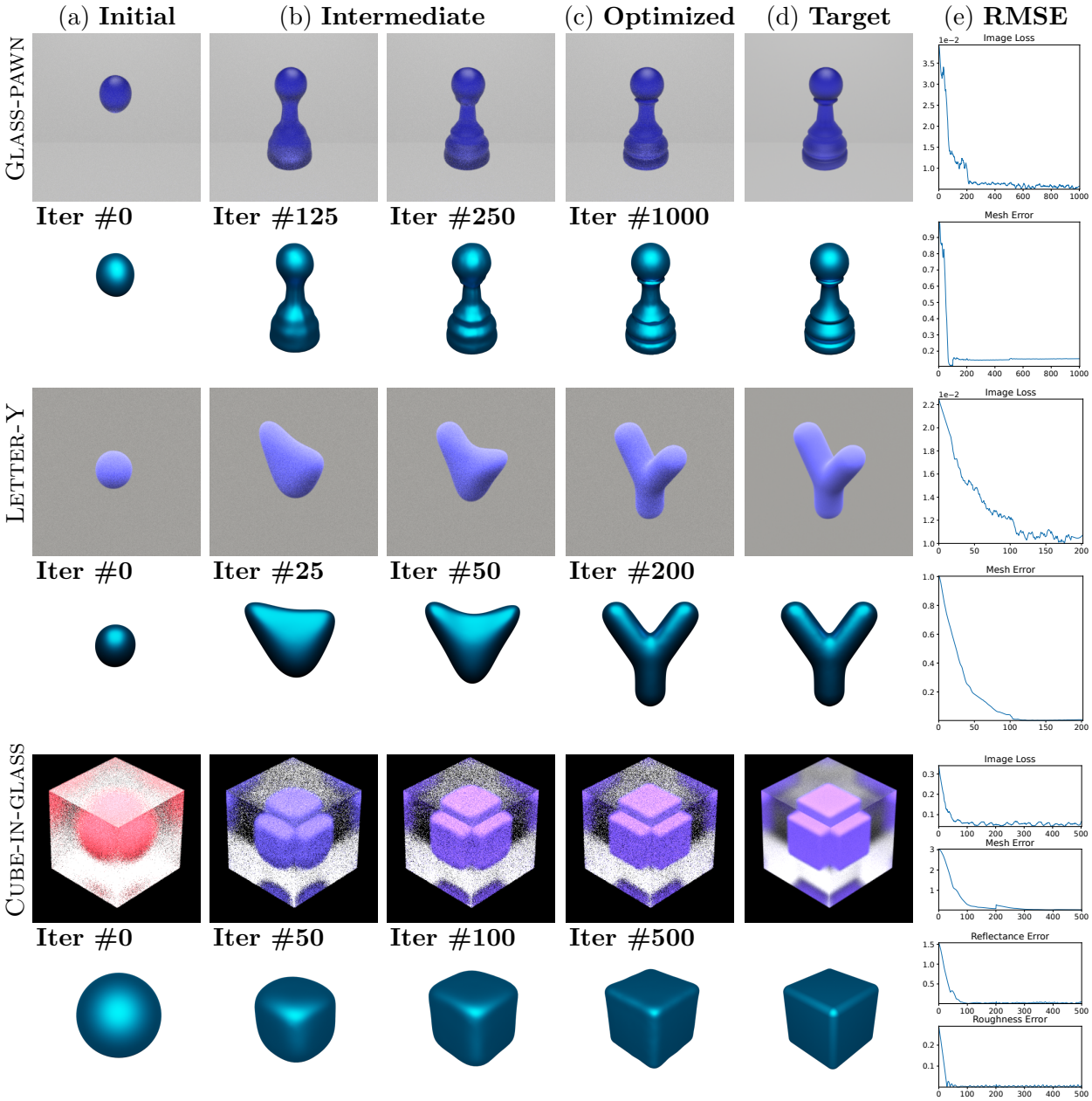


Figure 8.11: **Inverse-rendering results** obtained with gradients estimated using our unidirectional estimator (**I.1**). All examples in this figure involve non-opaque (i.e., transparent or translucent) objects that cannot be easily handled by simple rasterization-based or direct-illumination-only differentiable renderers. The mesh, reflectance, and roughness errors are used for evaluation only (and not for optimization).

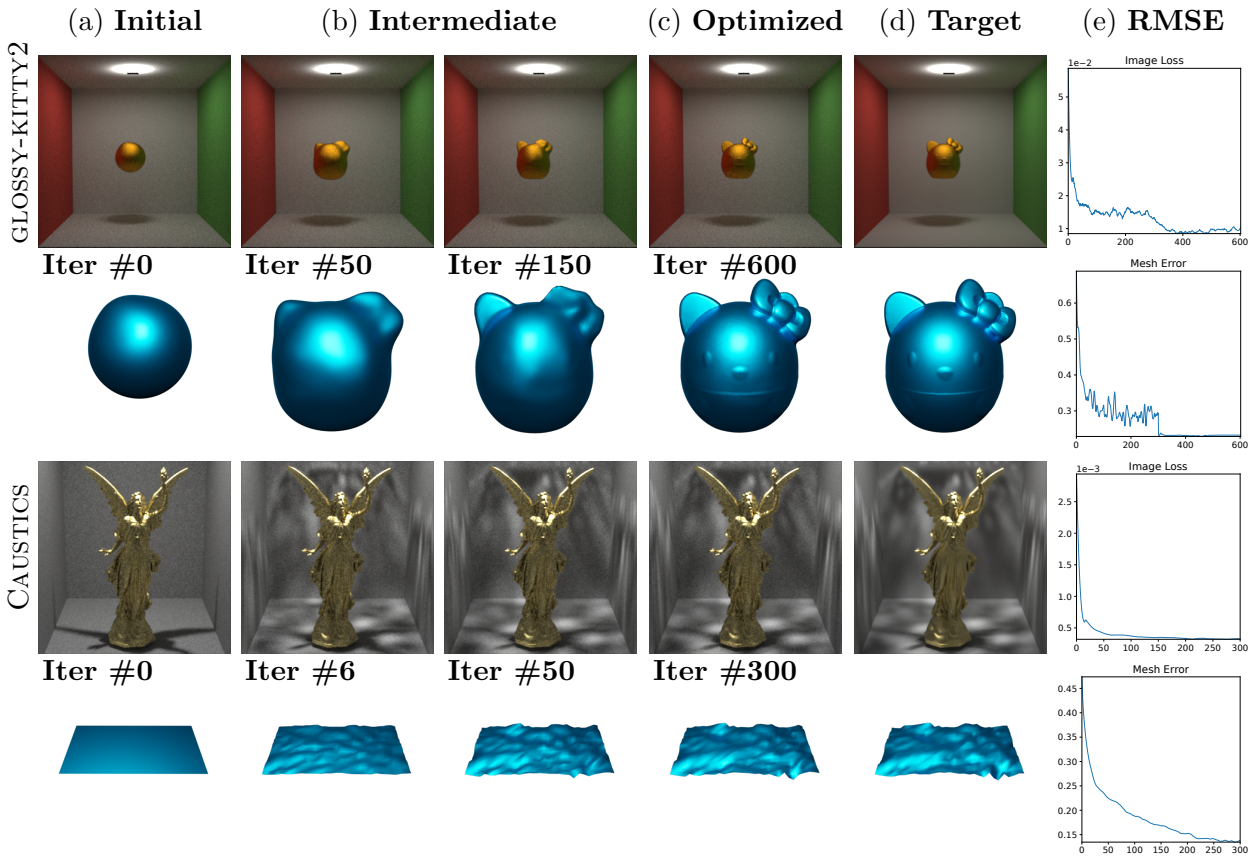


Figure 8.12: **Inverse-rendering results** obtained with gradients estimated using our bidirectional estimator (**I.2**). The mesh error is used for evaluation only (and not for optimization).

Chapter 9

Conclusion

Physics-based differentiable rendering focuses on differentiating photorealistic forward renderings with respect to arbitrary scene parameters. Being crucial for not only solving inverse-rendering problems but also integrating physics-based forward rendering into probabilistic-inference and machine-learning pipelines, differentiable-rendering techniques have applications in a wide array of areas such as computer graphics, vision, computational imaging, and computational fabrication.

This dissertation establishes a new theoretical framework for physics-based differentiable rendering and introduces a family of new path-space algorithms based on this framework. Our main contributions are summarized in the following.

Differential path integral. We introduced the theoretical framework of differential path integrals (Chapter 4) that offers the generality of differentiating both interfacial and volumetric light transport with respect to arbitrary scene parameters. To devise the differential path integrals, we rewrote forward-rendering path integrals [81, 65] in the material form (Chapter 3), where the domain of integration is independent of the scene parameters.

Path-space differentiable rendering algorithms. Based on the new framework, we developed new Monte Carlo methods that provide unbiased and consistent estimations of the *interior* and *boundary* components of differential path integrals (Chapter 5). Specifically, for efficient estimation of the *boundary* component, we introduced a multi-directional method (Algorithm 1) that does not require explicitly searching for object silhouettes. In practice, our techniques are capable of efficiently handling complex scene geometries and light-transport effects such as soft shadows, interreflection, and caustics.

Antithetic sampling. To further improve the efficiency and robustness when estimating the *interior* components of differential path integrals, we introduced antithetic sampling to path-space differentiable rendering (Chapter 6). Specifically, we developed new antithetic sampling algorithms for efficient differentiation of glossy BSDFs and pixel reconstruction filters. In addition, we demonstrated how these algorithms to be applied for full light transport paths (as opposed to individual vertices).

Efficient differentiation. Lastly, for efficiently handling large numbers (e.g., 10^6–10^9) of scene parameters, we presented the formulation of differentiable image-loss path integrals (Chapter 7) that directly expresses image-loss gradients as *differential path integrals*. Based on this formulation, we developed algorithms that efficiently compute the *interior* components by exploiting the layered structure of computation graphs. In addition, we demonstrated how the *boundary* integrals can be handled in a unified fashion.

9.1 Future Research Topics

General light-transport formulations. Our theory and algorithms presented in the earlier chapters have assumed steady-state and unpolarized light transport. Thus, extending

our techniques to more general (e.g., polarized) light-transport scenarios is an interesting topic for future research. Notably, Wu et al. [89] have recently introduced the formulation of *time-gated differential path integrals* based on our theory to differentiate time-of-flight renderings.

Advanced differentiable rendering systems. To integrate differentiable rendering in inverse-rendering or machine-learning pipelines, efficient systems are as important, if not more so, as sophisticated theories. Thankfully, significant progresses have been made recently in the development of efficient differentiable renderers such as Redner [45] and Mitsuba 2 [62] as well as automatic differentiation frameworks such as Enoki [31], Dr. Jit [33], and Enzyme [55].

In this regard, one important direction for future research and development is to build new differentiable rendering systems based on our path-space algorithms introduced in the earlier chapters. To ensure practical usefulness, these systems need to be capable of efficiently handling complex virtual scenes with large numbers of parameters. Ideally, these systems should also support models, such as BSDFs, expressed using neural networks. One example in this direction is the recently released PSDR-CUDA [91] renderer that implements some of our path-space algorithms on the GPU using a wavefront design.

Physics-based inverse rendering. As discussed in Chapter 8, a major application of differentiable rendering is to solve inverse-rendering problems. Although we only showed synthetic results in this chapter, our technique has been used by several recent works for reconstructing 3D models of real-world objects.

Specifically, Luan et al. [50] have developed an approach to jointly reconstruct the shape and reflectance of a real-world object. Cai et al. [8] have recently introduced an inverse-rendering pipeline that utilizes both implicit (e.g., SDF) and explicit (e.g., meshes) representations

of object geometries. Additionally, Deng et al. [15] have extended our technique for 3D reconstruction of translucent objects.

Physics-based learning. Lastly, with efficient differentiable rendering systems, integrating physics-based forward rendering into machine-learning pipelines is worth exploring. We believe that strategically combining data-driven methods and inverse-rendering (aka. analysis-by-synthesis) optimizations has the potential to yield new *physics-based learning* techniques that enjoy a new level of robustness and generalizability.

Bibliography

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[2] J. Arvo. The Irradiance Jacobian for partially occluded polyhedral sources. In *SIGGRAPH '94*, pages 343–350, 1994.

[3] D. Azinovic, T.-M. Li, A. Kaplanyan, and M. Niessner. Inverse path tracing for joint material and lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[4] S. Bangaru, T.-M. Li, and F. Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020.

[5] S. P. Bangaru, J. Michel, K. Mu, G. Bernstein, T.-M. Li, and J. Ragan-Kelley. Systematically differentiating parametric discontinuities. *ACM Trans. Graph.*, 40(4):107:1–107:18, 2021.

[6] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. Technical report, SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1977.

[7] J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82*, pages 21–29, 1982.

[8] G. Cai, K. Yan, Z. Dong, I. Gkioulekas, and S. Zhao. Physics-based inverse rendering using combined implicit and explicit geometries. *Computer Graphics Forum*, 41(4), 2022.

[9] P. Cermelli, E. Fried, and M. E. Gurtin. Transport relations for surface integrals arising in the formulation of balance laws for evolving fluid interfaces. *Journal of Fluid Mechanics*, 544:339–351, 2005.

[10] S. Chandrasekhar. *Radiative Transfer*. Dover Books on Intermediate and Advanced Mathematics. Dover Publications, 1960.

[11] C. Che, F. Luan, S. Zhao, K. Bala, and I. Gkioulekas. Inverse transport networks. *arXiv preprint arXiv:1809.10820*, 2018.

[12] J. Chen, B. Wang, and J.-H. Yong. Improved stochastic progressive photon mapping with metropolis sampling. In *Computer Graphics Forum*, volume 30, pages 1205–1213. Wiley Online Library, 2011.

[13] M. Chen and J. Arvo. Theory and application of specular path perturbation. *ACM Trans. Graph.*, 19(4):246–278, 2000.

[14] M. F. Cohen, J. R. Wallace, and P. Hanrahan. *Radiosity and realistic image synthesis*. Morgan Kaufmann, 1993.

[15] X. Deng, F. Luan, B. Walter, K. Bala, and S. Marschner. Reconstructing translucent objects using differentiable rendering. In *Proceedings of SIGGRAPH 2022*, 2022.

[16] Y. Dong. Digitalized 3D Object Collection, 2022. http://www.shading.me.

[17] H. Flanders. Differentiation under the integral sign. *The American Mathematical Monthly*, 80(6):615–627, 1973.

[18] I. Georgiev, J. Křivánek, T. Davidovič, and P. Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, 2012.

[19] J. Geweke. Antithetic acceleration of Monte Carlo integration in Bayesian inference. *Journal of Econometrics*, 38(1-2):73–89, 1988.

[20] I. Gkioulekas, A. Levin, and T. Zickler. An evaluation of computational imaging techniques for heterogeneous inverse scattering. In *European Conference on Computer Vision*, pages 685–701. Springer, 2016.

[21] I. Gkioulekas, S. Zhao, K. Bala, T. Zickler, and A. Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6):162:1–162:13, 2013.

[22] A. Griewank and A. Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*, volume 105. Siam, 2008.

[23] M. E. Gurtin. *An introduction to continuum mechanics*. Academic press, 1981.

[24] T. Hachisuka and H. W. Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Trans. Graph.*, 30(5):114–1, 2011.

[25] J. M. Hammersley and J. Mauldon. General principles of antithetic variates. In *Mathematical proceedings of the Cambridge philosophical society*, volume 52, pages 476–481. Cambridge University Press, 1956.

[26] J. Hasselgren, J. Munkberg, J. Lehtinen, M. Aittala, and S. Laine. Appearance-driven automatic 3D model simplification. *arXiv preprint arXiv:2104.03989*, 2021.

[27] M. Hašan and R. Ramamoorthi. Interactive albedo editing in path-traced volumetric materials. *ACM Trans. Graph.*, 32(2):11:1–11:11, 2013.

[28] E. Heitz and E. d'Eon. Importance sampling microfacet-based BSDFs using the distribution of visible normals. In *Computer Graphics Forum*, volume 33, pages 103–112. Wiley Online Library, 2014.

[29] H. Igehy. Tracing ray differentials. In *SIGGRAPH '99*, pages 179–186, 1999.

[30] A. Ishimaru. *Wave propagation and scattering in random media*. Academic press New York, 1978.

[31] W. Jakob. Enoki: structured vectorization and differentiation on modern processor architectures, 2019. https://github.com/mitsuba-renderer/enoki.

[32] W. Jakob and S. Marschner. Manifold exploration: A Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. *ACM Trans. Graph.*, 31(4), 2012.

[33] W. Jakob, S. Speierer, N. Roussel, and D. Vicini. Dr.Jit: A just-in-time compiler for differentiable rendering. *CoRR*, abs/2202.01284, 2022.

[34] W. A. Jakob. *Light transport on path-space manifolds*. Cornell University, 2013.

[35] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[36] J. T. Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.

[37] J. T. Kajiya and B. P. Von Herzen. Ray tracing volume densities. *SIGGRAPH Comput. Graph.*, 18(3):165–174, 1984.

[38] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3907–3916, 2018.

[39] C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, volume 21, pages 531–540. Wiley Online Library, 2002.

[40] M. Kettunen, M. Manzi, M. Aittala, J. Lehtinen, F. Durand, and M. Zwicker. Gradient-domain path tracing. *ACM Trans. Graph.*, 34(4), 2015.

[41] P. Khungurn, D. Schroeder, S. Zhao, K. Bala, and S. Marschner. Matching real fabrics with micro-appearance models. *ACM Trans. Graph.*, 35(1):1:1–1:26, 2015.

[42] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[43] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. Mansinghka. Picture: a probabilistic programming language for scene perception. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[44] S. Laine, J. Hellsten, T. Karras, Y. Seol, J. Lehtinen, and T. Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[45] T.-M. Li, M. Aittala, F. Durand, and J. Lehtinen. Differentiable Monte Carlo ray tracing through edge sampling. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[46] T.-M. Li, J. Lehtinen, R. Ramamoorthi, W. Jakob, and F. Durand. Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Trans. Graph.*, 34(6):209:1–209:13, 2015.

[47] Z. Li, Z. Xu, R. Ramamoorthi, K. Sunkavalli, and M. Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Trans. Graph.*, 37(6):269:1–269:11, 2018.

[48] M. M. Loper and M. J. Black. OpenDR: an approximate differentiable renderer. In *European Conference on Computer Vision*, pages 154–169. Springer, 2014.

[49] G. Loubet, N. Holzschuch, and W. Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Transactions on Graphics (TOG)*, 38(6):1–14, 2019.

[50] F. Luan, S. Zhao, K. Bala, and Z. Dong. Unified shape and SVBRDF recovery using differentiable Monte Carlo rendering. *Computer Graphics Forum*, 40(4):101–113, 2021.

[51] J. L. McClelland, D. E. Rumelhart, P. R. Group, et al. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2:216–271, 1986.

[52] M. McGuire. Computer Graphics Archive, July 2017. https://casual-effects.com/data.

[53] A. Meka, M. Maximov, M. Zollhoefer, A. Chatterjee, H.-P. Seidel, C. Richardt, and C. Theobalt. Lime: Live intrinsic material estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6315–6324, 2018.

[54] M. I. Mishchenko, L. D. Travis, and A. A. Lacis. *Multiple scattering of light by particles: radiative transfer and coherent backscattering*. Cambridge University Press, 2006.

[55] W. Moses and V. Churavy. Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12472–12485. Curran Associates, Inc., 2020.

[56] A. Mosinska, P. Marquez-Neila, M. Koziński, and P. Fua. Beyond the pixel-wise loss for topology-aware delineation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3136–3145, 2018.

[57] J. Munkberg, J. Hasselgren, T. Shen, J. Gao, W. Chen, A. Evans, T. Müller, and S. Fidler. Extracting triangular 3d models, materials, and lighting from images. *arXiv preprint arXiv:2111.12503*, 2021.

[58] B. Nicolet, A. Jacobson, and W. Jakob. Large steps in inverse rendering of geometry. *ACM Trans. Graph.*, 40(6):248:1–248:13, 2021.

[59] M. Nimier-David, Z. Dong, W. Jakob, and A. Kaplanyan. Material and lighting reconstruction for complex indoor scenes with texture-space differentiable rendering. 2021.

[60] M. Nimier-David, T. Müller, A. Keller, and W. Jakob. Unbiased inverse volume rendering with differential trackers. *ACM Trans. Graph.*, 41(4):44:1–44:20, July 2022.

[61] M. Nimier-David, S. Speierer, B. Ruiz, and W. Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering. *ACM Trans. Graph.*, 39(4), 2020.

[62] M. Nimier-David, D. Vicini, T. Zeltner, and W. Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics (TOG)*, 38(6):1–17, 2019.

[63] A. C. Öztireli. Integration with stochastic point processes. *ACM Trans. Graph.*, 35(5):160:1–160:16, 2016.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32:8026–8037, 2019.

[65] M. Pauly, T. Kollig, and A. Keller. Metropolis light transport for participating media. In *Eurographics Workshop on Rendering Techniques*, pages 11–22. Springer, 2000.

[66] F. Petersen, A. H. Bermano, O. Deussen, and D. Cohen-Or. Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv preprint arXiv:1903.11149*, 2019.

[67] G. Pomraning. The equations of radiation hydrodynamics. *International Series of Monographs in Natural Philosophy, Oxford: Pergamon Press, 1973*, 1973.

[68] R. Ramamoorthi, D. Mahajan, and P. Belhumeur. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph.*, 26(1):2:1–2:21, 2007.

[69] H. Ren, S. Zhao, and S. Ermon. Adaptive antithetic sampling for variance reduction. In *International Conference on Machine Learning*, pages 5420–5428. PMLR, 2019.

[70] O. Reynolds. *Papers on mechanical and physical subjects: the sub-mechanics of the universe*, volume 3. The University Press, 1903.

[71] H. E. Rushmeier and K. E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. In *SIGGRAPH '87*, pages 293–302, 1987.

[72] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[73] S. Sengupta, A. Kanazawa, C. D. Castillo, and D. W. Jacobs. SfSNet: Learning shape, reflectance and illuminance of faces in the wild'. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6296–6305, 2018.

[74] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[75] G. Singh, K. Subr, D. Coeurjolly, V. Ostromoukhov, and W. Jarosz. Fourier analysis of correlated Monte Carlo importance sampling. *Computer Graphics Forum*, 39(1):7–19, 2020.

[76] G. Singh, C. Öztireli, A. G. Ahmed, D. Coeurjolly, K. Subr, O. Deussen, V. Ostromoukhov, R. Ramamoorthi, and W. Jarosz. Analysis of sample correlations for Monte Carlo rendering. *Computer Graphics Forum*, 38(2):473–491, 2019.

[77] K. Subr, D. Nowrouzezahrai, W. Jarosz, J. Kautz, and K. Mitchell. Error analysis of estimators that use combinations of stochastic sampling strategies for direct illumination. *Computer Graphics Forum*, 33(4):93–102, 2014.

[78] D. Sumin, T. Rittig, V. Babaei, T. Nindel, A. Wilkie, P. Didyk, B. Bickel, J. Křivánek, K. Myszkowski, and T. Weyrich. Geometry-aware scattering compensation for 3D printing. *ACM Trans. Graph.*, 38(4):111:1–111:14, 2019.

[79] C.-Y. Tsai, A. C. Sankaranarayanan, and I. Gkioulekas. Beyond volumetric albedo—a surface optimization framework for non-line-of-sight imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[80] H. C. van de Hulst et al. Light scattering by small particles. 1957.

[81] E. Veach. *Robust Monte Carlo methods for light transport simulation*, volume 1610. Stanford University PhD thesis, 1997.

[82] E. Veach and L. Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995.

[83] Z. Velinov, M. Papas, D. Bradley, P. Gotardo, P. Mirdehghan, S. Marschner, J. Novák, and T. Beeler. Appearance capture and modeling of human teeth. *ACM Trans. Graph.*, 37(6):207:1–207:13, 2018.

[84] D. Vicini, S. Speierer, and W. Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *ACM Trans. Graph.*, 40(4), 2021.

[85] B. Walter, S. R. Marschner, H. Li, and K. E. Torrance. Microfacet models for refraction through rough surfaces. *Rendering techniques*, 2007:18th, 2007.

[86] R. E. Wengert. A simple automatic derivative evaluation program. *Communications of the ACM*, 7(8):463–464, 1964.

[87] E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In *Proc. Conf. Applications of Computing Methods to Reactor Problems*, volume 557, 1965.

[88] J. Wu, J. B. Tenenbaum, and P. Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 699–707, 2017.

[89] L. Wu, G. Cai, R. Ramamoorthi, and S. Zhao. Differentiable time-gated rendering. *ACM Transactions on Graphics (TOG)*, 40(6):1–16, 2021.

[90] M. Wu, N. Goodman, and S. Ermon. Differentiable antithetic sampling for variance reduction in stochastic variational inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2877–2886. PMLR, 2019.

[91] K. Yan and S. Zhao. PSDR-CUDA: Path-space differentiable renderer, 2021. https://github.com/uci-rendering/psdr-cuda/.

[92] T. Zeltner, S. Speierer, I. Georgiev, and W. Jakob. Monte Carlo estimators for differential light transport. *ACM Trans. Graph.*, 40(4):78:1–78:16, 2021.

[93] C. Zhang, Z. Dong, M. Doggett, and S. Zhao. Antithetic sampling for Monte Carlo differentiable rendering. *ACM Trans. Graph.*, 40(4):77:1–77:12, 2021.

[94] C. Zhang, B. Miller, K. Yan, I. Gkioulekas, and S. Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020.

[95] C. Zhang, L. Wu, C. Zheng, I. Gkioulekas, R. Ramamoorthi, and S. Zhao. A differential theory of radiative transfer. *ACM Trans. Graph.*, 38(6):227:1–227:16, 2019.

[96] C. Zhang, Z. Yu, and S. Zhao. Path-space differentiable rendering of participating media. *ACM Trans. Graph.*, 40(4):76:1–76:15, 2021.

[97] S. Zhao, L. Wu, F. Durand, and R. Ramamoorthi. Downsampling scattering parameters for rendering anisotropic media. *ACM Trans. Graph.*, 35(6):166:1–166:11, 2016.

[98] C. Zheng and D. L. James. Energy-based self-collision culling for arbitrary mesh deformations. *ACM Trans. Graph.*, 31(4):98:1–98:12, 2012.