EPFL

Thèse n° 8188

Light Path Gradients for Forward and Inverse Rendering

Présentée le 10 décembre 2021

Faculté informatique et communications
Laboratoire d'informatique graphique réaliste
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Tizian Lucien ZELTNER

École
polytechnique
fédérale
de Lausanne

2021

Abstract

Physically based rendering is a process for photorealistic digital image synthesis and one of the core problems in computer graphics. It involves simulating the *light transport*, i.e. the emission, propagation, and scattering of light through a virtual scene that is defined by a detailed description of object geometry and appearance. Research over the last decades has led to sophisticated rendering techniques and recently, the inversion of this process, i.e. recovering scene parameters from image observations, has also received significant attention. In this thesis, we investigate methods in both physically based *forward* and *inverse* rendering that exploit *light path gradients*.

The first part is concerned with scattering from *specular surfaces*, which produces complex optical effects that are frequently encountered in realistic scenes: intricate caustics due to focused reflection, multiple refractions, and high-frequency glints from specular microstructure. Yet, despite their importance and considerable research to this end, sampling of light paths that cause these effects remains a formidable challenge of forward rendering.

We propose a surprisingly simple and general path sampling strategy that targets the examples above. Valid light path configurations need to fulfill the physical laws of reflection and refraction, and we find these using a numerical root-finding process that is driven by geometric light path gradients. In contrast to prior work, our method supports high-frequency normal- or displacement-mapped geometry, samples specular-diffuse-specular (SDS) paths, and is compatible with standard Monte Carlo methods including unidirectional path tracing. We demonstrate our method on a range of challenging scenes and evaluate it against state-of-the-art methods for rendering caustics and glints.

In the second part, we consider *differentiable rendering algorithms.* These propagate derivatives through the full light transport simulation to solve inverse rendering problems via gradient-based optimization. Recent progress has led to methods that can simultaneously compute derivatives with respect to millions of scene parameters. At the same time, elementary properties of these methods remain poorly understood.

Current algorithms for differentiable rendering are constructed by mechanically differentiating a given primal algorithm. As differentiation fundamentally changes the underlying problem, this is often suboptimal and instead, primal and differential algorithms should be decoupled so that the latter can suitably adapt. This is surprisingly complex.

Even the most basic Monte Carlo path tracer already involves several design choices concerning the techniques for sampling materials and emitters, and their combination, e.g. via multiple importance sampling (MIS). Differentiation causes a veritable explosion of this decision tree: should we differentiate only the estimator, or also the sampling technique? Should MIS be applied before or after differentiation? Are specialized derivative sampling strategies of any use? How should visibility-related discontinuities be handled when millions of parameters are differentiated simultaneously? We provide a taxonomy and analysis of different estimators for differential light transport to provide intuition about these and related questions.

Acknowledgements

I am extremely grateful to my advisor Wenzel Jakob for his constant guidance and support over the last five years. I feel fortunate that he gave me this unique opportunity despite my little research experience at the time. Learning from him and working on a variety of different projects together was truly enjoyable and I could not have asked for a better advisor.

I would also like to extend my deepest gratitude to my thesis jury Mark Pauly, Wojciech Jarosz, Tzu-Mao Li, and Sabine Süsstrunk for their time and the interesting discussion during the private defense.

It was a great pleasure to work with the other members of the *Realistic Graphics Lab*, in particular: Merlin Nimier-David, Delio Vicini, Guillaume Loubet, Sébastien Speierer, Baptiste Nicolet, Miguel Crespo, Dongqing Wang, and Ziyi Zhang. The numerous conversations with them greatly benefited all of my research projects and I will miss working in this fun environment very much. Thanks should also go to Pauline Raffestin for keeping everything running smoothly in the background.

I am also grateful to Seung-Hwan Baek, Min H. Kim, Xin Tong, Hyun Jin Ku, Inseung Hwang, Nicolas Holzschuch, and especially Iliyan Georgiev for collaborating on some of the publications I was a part of. Thanks also to Sai Praveen Bangaru, Lingqi Yan, and Miloš Hašan for helpful discussions and providing code of prior work.

During my second year I was lucky to intern in the *Hyperion Renderer* team at *Walt Disney Animation Studios* in Los Angeles. Matt Chiang, Yining Karl Li, Daniel Teece, David Adler, Peter Kutz, Ralf Habel, and Brent Burley were very welcoming and I fondly look back at my time there.

I want to additionally thank Héloïse De Dinechin for pointing out a crucial bug in the reference implementation of one of our papers, Olesya Jakob for designing some of the scenes in our teaser figures, and Milan Bombsch for proofreading large parts of this thesis and bringing up many helpful comments.

I really appreciate the friendship of Michalina Pacholska who started her PhD at the same time as I. We shared many of the same ups and down and she always listened to all my complaints and offered useful advice, especially when research was not going very smoothly.

I thank my girlfriend Silja for all her love and emotional support over these years, and for being so immensely patient with me regarding long working hours and the many stressful SIGGRAPH deadlines.

Finally, I would like to give special thanks to my parents, Beatrix and Raymond, for their continued support and encouragement throughout my life. The completion of this thesis would not have been possible without them.

The research in this thesis was generously supported by a grant from *Autodesk*.

Figure 1.1 shows a modified scene licensed from *Evermotion's Archinteriors vol. 43*.
The shoes in Figures 3.1, 3.23 and 3.30 are modeled by Olesya Jakob.
The kettle model in Figure 3.30 is by *Blendswap* user *PrinterKiller*.
The scene from Figures 4.1, 4.20 and 4.21 includes a silver tray modeled by Olesya Jakob, and various other objects licensed from *Turbosquid*: two different pitchers by *1d_inc*, cups by *Tornado Studio*, a frame by *aelxr*, and a lamp by *rabser*.
The chess pieces in Figures 4.2 to 4.5 and 4.24 is by *Free3D* user *printable_models*.
The PBRT book model in Figure 4.24 is adapted from a scene by Yining Karl Li.
The chair and textures in Figure 4.23 are from Benedikt Bitterli's rendering resources, in particular the *The Wooden Staircase* scene created by *Blendswap* user *Wig42*.
The background of Figure 4.25 is licensed from *cgtrader* user *random-scans*.
Many of the scenes additionally use textures from *CC0 Textures*, *cgbookcase*, *Share Textures*, and *Texture Haven*. We use environment maps courtesy Paul Debevec and *HDRI Haven*.

To Beatrix and Raymond

Table of Contents

List of Figures

List of Algorithms

1 | Introduction

Digital image synthesis in the form of *physically based rendering* is a classical problem in computer graphics and beyond. The goal is to faithfully simulate the underlying physical process of light emission, scattering through a scene, and measurement by a sensor. Progress in this field over the last decades has lead to tremendous advances in both accuracy and efficiency of these simulations. They are capable of producing images with an extremely high degree of realism and they have been widely adopted in areas like visual effects for both live-action or animated film production, product design, or architecture visualization.

Figure 1.1 shows a high level overview of the rendering process which turns a description of a *virtual scene* into a rendered image. Such a description takes the role of a digital blueprint and includes at least the following elements:

- A geometric representation of all objects in the scene, usually in the form of polyhedral meshes (e.g. made from triangles or quads).
- Information about the surface material properties of objects, which dictates how light should interact with them.
- A number of light sources that emit light into the scene.
- A virtual camera (or other measurement sensor), which specifies the framing of the final image.

Scene creation is an important, and often labor-intensive, step in the production of photorealistic renderings. It requires modeling of intricate geometric detail as well as faithful recreation of real-world materials. To this end, image textures are often used to encode spatially varying material or emission properties. Common examples are the diffuse color or the surface roughness of objects. Depending on the desired use case, textures can either be created by the hand of an artist, or captured from the real world. They are also commonly used to add fine-scale details to the geometry via normal or displacement maps that modify the orientation of surface shading normals or the position of mesh vertices.

The digital image is then created by computing the intensity of light arriving at each individual pixel, which turns out to be an *infinite-dimensional integration problem* over the space of all possible light paths connecting the emitters with the camera. Rendering algorithms that solve this mostly sample such paths stochastically using *Monte Carlo*

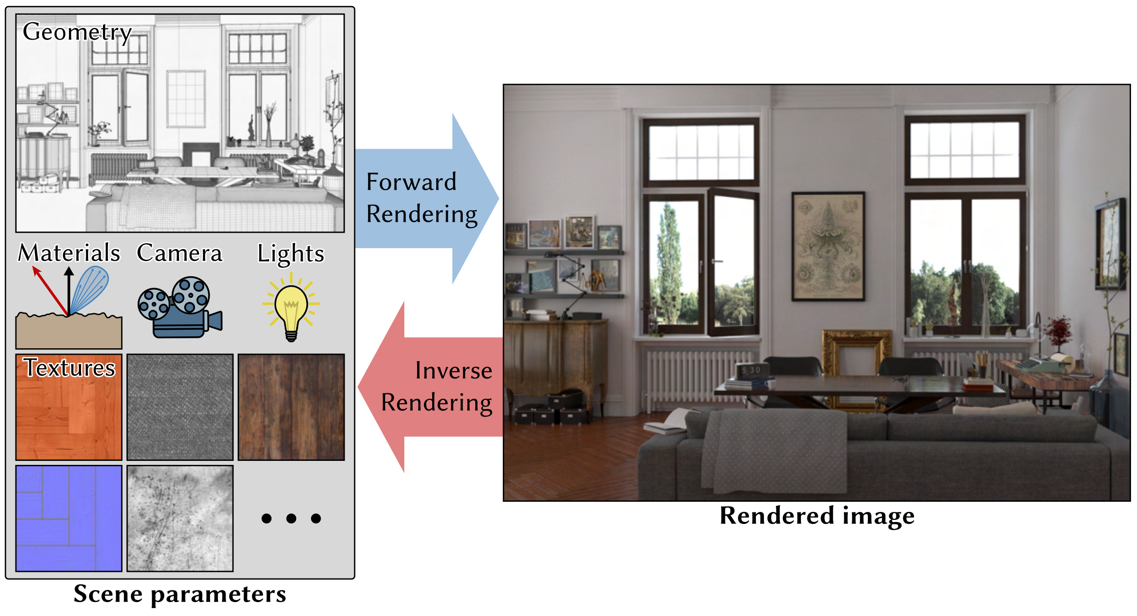


Figure 1.1: Physically based *forward* rendering generates images based on a detailed description of a virtual scene, including a geometric representation of objects and information about their material properties. The *inverse* rendering process turns this problem on its head and instead attempts to *recover* scene parameters from given images. In this case, the input does not necessarily need to be a rendered image but can instead be a photograph or other type of measurement.

integration methods. Light paths are constructed from multiple straight lines connecting points in the scene that were found via *ray tracing*, an operation that determines intersection locations between lines and the scene geometry.

The comprehensive nature of such a simulation is what ultimately allows us to recreate the rich visual complexity found in the real world, such as diffuse or glossy interreflections between objects in the scene. Realistic light sources cast soft shadows from objects, the time dimension of a scene can be taken into account to capture motion blur, and virtual cameras even simulate scattering through complex lens systems resulting in effects like depth of field. All this can be summarized under the general term of *global illumination*—a form of radiative coupling between all objects in the scene—and this happens naturally due to how the simulation imitates the true physical process.

This is generally no simple task: generating a single image with such a method can take minutes or even hours on current hardware. While this can be acceptable in some cases (e.g. the visual effects industry), it is impractical for real-time applications (e.g. video games) where individual animation frames need to be rendered within milliseconds. The currently predominant solution to this is to use *rasterization* methods in favor of ray tracing. These have the advantage that objects are shaded and drawn to the final image in isolation of each other, which enables high performance due to hardware support in traditional GPU architectures. Compared to ray tracing, this unfortunately cannot account for global illumination between objects as easily, without using additional approximations or workarounds.

The recent addition of hardware accelerated ray tracing on modern GPUs is closing this gap more and more and, especially during the last few years, ray tracing has also become increasingly feasible and popular in real-time settings. In practice, rendering at acceptable quality levels is often still problematic due to the tight computational budget for each frame, so physically based rendering methods are usually combined with spatial and temporal post-processing (e.g. denoising) in order to create high fidelity outputs in real-time.

Light transport simulation is not only useful for the creation of appealing looking images. Many scientific fields are studying the propagation of light or other electromagnetic radiation as a mean to understand the world surrounding us. Knowledge about accurate forward simulation turns out to be also tremendously helpful in designing approaches to invert this process (Figure 1.1).

Scene recovery and understanding from captured image data is of course a core problem in the field of *computer vision*. Some important applications include 3D shape reconstruction and pose estimation, sample analysis via various forms of biomedical imaging, or even earth and atmospheric observations using satellite data.

Unfortunately, inverting the full image formation process, including global illumination, is a remarkably challenging task, mostly due to a large number of possible ambiguities that can arise. Consider for instance a picture of a red object: from a single image, it can be unclear whether the object itself has a red color (due to its material properties) or if it is in fact a white object that is illuminated by a red light source. Many specialized reconstruction techniques therefore operate under additional assumptions and while they can be highly effective within their design scope, they can easily fail when these assumptions are violated. For example, the interior of an object can be reconstructed by taking a number of measurements with a computed tomography (CT) scanner and subsequently inverting the process of X-ray absorption. This normally assumes absorptive materials and tends to produce severe artifacts when the specimen contains metal fragments that are highly reflective to X-rays.

Rather than addressing specific flaws of such techniques, our goal of *physically based inverse rendering* is to study a universal mathematical framework that has the potential to improve the quality of solutions in these challenging inversion tasks in the future. We also note that global illumination can even provide valuable additional cues that we can leverage when unraveling the radiative scattering inside a scene.

1.1 Topics discussed in this thesis

The contributions of this thesis are split into two parts that focus on separate topics in forward and inverse rendering. In both cases, *light path gradients* will provide vital information to tackle the respective problems.

Part 1: Light transport involving specular materials. In the first part, we consider the problem of *specular light paths*: a category of paths connecting camera and emitters where light interacts at least with one *specular material*, such as a mirror or glass object. Even though today's state-of-the-art rendering techniques are generally considered to be robust in the presence of many different global illumination effects, these specular paths remain challenging—or in some configurations even impossible—to render with many popular algorithms. We focus on the two optical phenomena of *caustics* and *glinty microstructures* that are particularly eye-catching effects that arise due to such paths.

Caustics. Curved specular surfaces bend and focus incident light rays and cast intricate illumination patterns called caustics onto other (non-specular) surfaces in a scene. This indirect light effect is commonly observed when highly concentrated illumination, e.g. from the sun, strikes materials such as metals, glass, or water. This is the same principle that lets a camera lens systems focus light onto the image sensor.

Glinty microstructures. A related concept is the glinty appearance of certain material types. This mostly involves a single reflection from a light source to the observer via, e.g., a metal surface. This can create small, but visually striking, highlights due to surface detail and imperfections, such as tiny bumps or scratches. Many car paints also include small metallic flakes that cause a glittery appearance, and manufactured objects can have glinty appearance due to the underlying fabrication process, e.g. in the case of brushed aluminum.

Note that the surface details should be small *relative to the scale of the final image*. For example, the same challenges appear when observing highlights from the sun's reflection in a wavy water surface from a distance.

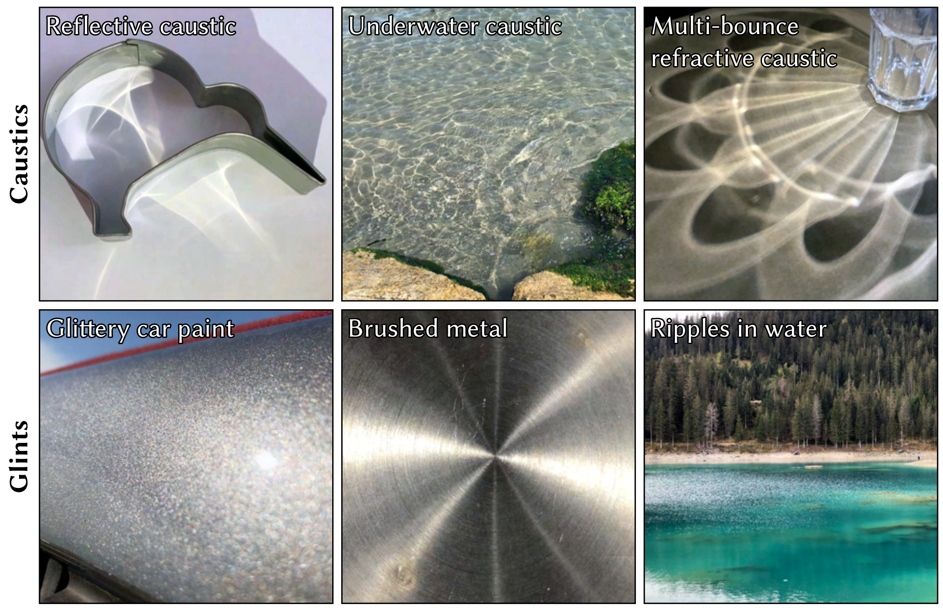


Figure 1.2: Examples of common caustics and glints. Note that these are all photographs.

See Figure 1.2 for some example photographs that illustrate representative examples of both caustics and glints from everyday life.

What makes rendering these two effects so challenging in practice? Light that interacts with specular materials has to follow the laws of reflection and refraction. This however means that only a small subset of all possible light paths connecting the lights and the sensor is "valid". Sampling the right configurations in this high-dimensional space becomes very unreliable in that case, which often translates to dramatic increases in rendering times compared to simpler scenarios without specular materials.

We will describe a general path sampling strategy that exploits this concentrated nature of specular paths and can generate them more systematically. In particular, we look at the problem from the perspective of *numerical root-finding* where valid paths represent roots of a high-dimensional function. We then compute light path gradients that give us information about how the geometric configuration of previously invalid paths should be modified. This is, in principle, nothing else than the well known Newton–Raphson method—but interesting challenges arise when combining such an approach with standard rendering techniques.

Part 2: Differentiable light transport. In the second part, we investigate physically based *differentiable* rendering and outline how it can be used as a tool for solving inverse rendering problems. Due to the previously discussed complexities, a direct inversion of light transport is almost always impossible and instead we resort to nonlinear optimization techniques.

Concretely, when recovering a set of scene parameters, we start with an *initial guess* for all values that is subsequently refined via an iterative optimization process. At each iteration, we can create a rendered image of the current state, which is then evaluated by an *objective* or *loss function* to quantify "how close" we are to a desired goal, e.g. by comparing against a target image. However, this does not tell us in what *direction* we should move next to improve the loss. Also note that the space of scene parameters, in which we search for a solution, is often huge—it is not uncommon to optimize for millions of scene parameters at once (e.g. textured material properties or vertex positions of a triangle mesh), so blindly moving around in this space is clearly ineffective.

A way out of this sounds rather straightforward at first: instead of only generating a rendering at each step of the process, we additionally want to compute the *scene parameter gradients.* This is known to be the direction of steepest ascent inside the loss landscape which allows us to take an informed step that will (at least locally) improve the objective. In other words, we rely on standard nonlinear optimization, which is relatively well understood today.

But this also means that we need to turn the complete image formation and rendering process into a differentiable operation. Note that this is much more complex compared to the gradients described in the context of specular paths above, which only involved the local light path geometry. It also goes beyond simply implementing the complete renderer on top of existing frameworks for automatic differentiation of numerical code, as this often performs suboptimally—or can even produce incorrect gradients when done naïvely.

In this thesis, we revisit large parts of the well established theory of physically based rendering in this new context of differentiation. We outline which modifications to existing forward rendering algorithms are necessary to solve the differentiated form of light transport. This also includes discussions of entirely new challenges that appear in such a setting, e.g. the problem of discontinuous visibility changes or silhouette edges that initially appear to be non-differentiable.

1.2 Organization of the thesis

The remainder of this thesis is split into four chapters. Chapter 2 will cover the necessary foundations behind physically based (forward) rendering to provide enough context for the two subsequent topics. Chapter 3 then covers the case of specular light paths and Chapter 4 summarizes the theory and implementation of differentiable rendering techniques for solving inverse problems. We also put particular emphasis on discussing prior work that is directly related to our new contributions. Finally, we conclude in Chapter 5.

1.3 List of all publications

A substantial part of this thesis is based on the following two publications [1, 2]:

Chapter 3:
Specular manifold sampling for rendering high-frequency caustics and glints
Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob
ACM Trans. Graph. (Proc. SIGGRAPH), Vol. 39, No. 4, pp 149:1–149:15, Jul. 2020

Chapter 4:
Monte Carlo estimators for differential light transport
Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob
ACM Trans. Graph. (Proc. SIGGRAPH), Vol. 40, No. 4, pp 78:1–78:16, Jul. 2021

The author has also contributed to these additional articles that are not described in this thesis [3, 4, 5, 6]:

The layer laboratory: a calculus for additive and subtractive composition of anisotropic surface reflectance
Tizian Zeltner and Wenzel Jakob
ACM Trans. Graph. (Proc. SIGGRAPH), Vol. 37, No. 4, pp 74:1–74:14, Aug. 2018

> We present a computational framework for modeling arbitrarily layered material structures that, compared to prior work, supports general anisotropic reflectance. This builds on a sparse representation in frequency-space that enables additive, as well as subtractive composition of layers. The accuracy and scope of our model is shown on different examples and validated against measurements of real-world materials.

Mitsuba 2: a retargetable forward and inverse renderer
Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob
ACM Trans. Graph. (Proc. SIGGRAPH Asia), Vol. 38, No. 6, pp 203:1–203:17, Nov. 2019

> This system paper describes the implementation and design of the *Mitsuba 2* renderer that is intrinsically retargetable to various use cases. This includes, e.g., changing the representation of color to spectral or polarized quantities, vectorization that operates on large bundles of light paths simultaneously, compilation into kernels that run on the GPU, or even differentiable rendering. We demonstrate this on several applications in both forward and inverse rendering.

Image-based acquisition and modeling of polarimetric reflectance
Seung-Hwan Baek, Tizian Zeltner, Hyun Jin Ku, Inseung Hwang, Xin Tong, Wenzel Jakob, and Min H. Kim
ACM Trans. Graph. (Proc. SIGGRAPH), Vol. 39, No. 4, pp 139:1–139:14, Jul. 2020

> While polarization is usually a subtle effect to a human observer, it can easily be perceived by an optical sensor (e.g. using polarizing optical elements). Realistic modeling of the interaction between polarized light and general materials is therefore useful for solving inverse rendering problems. This project involved the acquisition of the first polarimetric material dataset that captures the full angular domain at multiple wavelengths. The measured data reveals interesting relationships between polarization and appearance, and is readily usable in polarized physically based rendering systems.

Slope-space integrals for specular next event estimation
Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob
ACM Trans. Graph. (Proc. SIGGRAPH Asia), Vol. 39, No. 6, pp 239:1–239:13, Dec. 2020

> This article also investigates the problem of caustic and glint rendering, albeit with a very different approach compared to the previously mentioned method [1]. Instead of using numerical root-finding, this closely investigates the case of a single reflective or refractive triangle, and finds analytic expressions for the resulting radiance even in presence of glossy microfacet materials. This is then combined with an algorithm that can efficiently support meshes with high geometric complexity.

2 | Background

This section reviews background material and central references that relate to the topics of Chapters 3 and 4. Outstanding starting points for further reading are the seminal thesis *Robust Monte Carlo Methods for Light Transport Simulation* by Eric Veach [7] or the excellent book *Physically Based Rendering: From Theory to Implementation* by Pharr et al. [8].

We first begin by outlining the basics of light and how it is measured (Section 2.1), how it iteracts with typical materials (Section 2.2), and ultimately how to describe the scattering, propagation, and measurement processes using light transport theory (Section 2.3).

We then take a step back and review essential probability theory and especially Monte Carlo methods (Section 2.4) that are behind many state-of-the-art rendering algorithms (Section 2.5). Finally, we present a short overview of how gradients of computer programs can be evaluated efficiently (Section 2.6), which will be crucial for the two applications in forward and inverse rendering later.

2.1 Light and radiometry

Before moving on to the topic of light transport we will briefly cover the fundamental principles of light itself and how it is measured.

Humans are fundamentally visual creatures and therefore, scientists and philosophers have speculated for over two millennia about the nature of light and the process of vision. The *extramission theory* of light was first postulated by Greek philosopher Empedocles (494 BC–434 BC) who believed that the eyes emit a form of “fire” which lets us see objects after being combined with rays arriving from the sun. The theory was also famously supported by Plato (428 BC–348 BC) and Euclid (about 325 BC–265 BC) and was influential for many centuries until the Arab scholar Alhazen (965–1040) conclusively determined it to be wrong.

In a similar vein, opinions were split about whether light travels instantaneously or with a fixed speed: Descartes (1596–1650), for instance, believed strongly in an infinite speed of light and it was Rømer (1644–1710) who first demonstrated that its speed must be finite by observing irregularities in the eclipses of a Jupiter moon, which he attributed to the change in time that light requires to arrive at the earth due to the planets orbiting the sun.

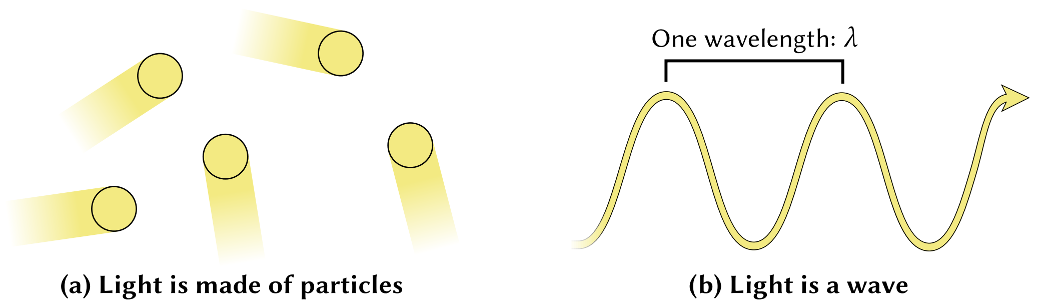


Figure 2.1: Two opposite views on the nature of light.

This also strongly supported Huygens's (1629–1695) theory that light was a wave propagating through the *ether*, similar to how sound waves propagate through air. A strong opponent of this was Newton (1643–1727) who believed in the *corpuscular theory* where light is made from particles instead, see Figure 2.1.

Throughout the 18th and 19th century, many experiments of diffraction, interference, and polarization by Young (1773–1829), Malus (1775–1812), Arago (1786–1853), and Fresnel (1788–1827) further established the wave theory, and it was Maxwell (1831–1879) who finally explained light inside the unified theory of *electromagnetism*. This means, light is a *transverse wave* and part of the larger electromagnetic spectrum (Figure 2.2): it consists of both an electric and a magnetic field that oscillate in union and reinforce each other. This is also what allows the wave to travel without the presence of a medium.

At this point, the understanding of light seemed complete. But the 20th century brought a revival of the particle theory after Planck (1858–1947) and Einstein (1879–1955) observed that light seems to be *quantized* into small packages that are now known as *photons*. Today, light is understood to be *both* a particle and a wave, and both theories are needed to fully explain its behavior.

Luckily, it is not necessary to simulate the full complexity of wave optics if our goal is the creation of photorealistic images. Instead, we can work in the framework of *geometric optics* where the effects of polarization, interference, and diffraction can be ignored. In such a setting, it is sufficient to use *ray tracing*, i.e. intersection of straight lines with the scene geometry, in order to construct light paths. Further, we assume no wavelength-dependent effects like dispersion, fluorescence, or phosphorescence in this thesis. Note however that all these topics have actually been explored in the context of computer graphics too. For instance, there are rendering systems that additionally keep track of polarization or fluorescence throughout the simulation [4, 9] and it is increasingly more

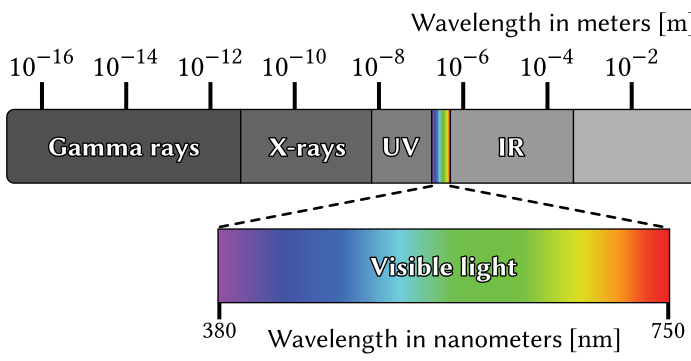


Figure 2.2: Visible light represents a small band of the full electromagnetic spectrum, between roughly 380 and 750 nanometers.

common to locally account for wave-optical effects in derivations of material models [10, 11, 12]. Recently, Steinberg and Yan [13] also made advances towards a generalized light transport framework that can incorporate interference effects on a global scale.

Radiometry is concerned with the measurement of electromagnetic radiation. A good introduction to the field is McCluney's reference text [14] but many sources in computer graphics also cover the essentials [7, 15] which we will outline here. As radiometry measures positional and/or directional densities of energy we will first clarify these domains.

2.1.1 Positional and directional domains

For the purpose of simulating light scattering between objects in a (virtual) scene, we consider positions $\mathbf{x} \in \mathcal{M}$ on a two dimensional sub manifold of surfaces embedded in 3D space ($\mathcal{M} \subseteq \mathbb{R}^3$).

The associated *area measure* $\mathrm{d}A(\mathbf{x})$ is used to integrate functions over surface areas:

$$\int_{\mathcal{M}} f(\mathbf{x})\,\mathrm{d}A(\mathbf{x}). \tag{2.1}$$

Light is traveling between positions along straight lines parameterized as normalized directions $\boldsymbol{\omega} \in \mathcal{S}^2$ on the 3D unit sphere. They are also commonly expressed via spherical coordinates *zenith* $\theta \in [0, \pi)$ and *azimuth* $\phi \in [0, 2\pi)$ as

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix} \tag{2.2}$$

computed via

$$\theta = \arccos(\omega_z), \quad \phi = \operatorname{arctan2}(\omega_y, \omega_x). \tag{2.3}$$

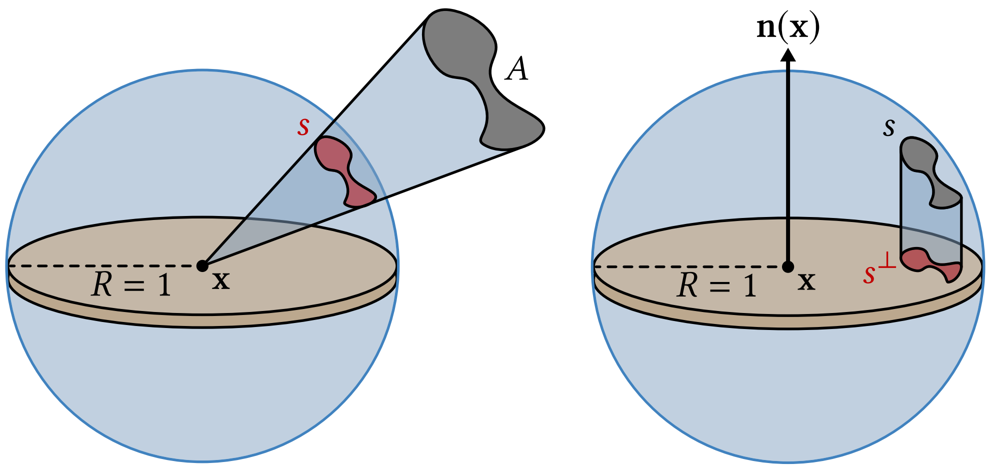


Figure 2.3: **Left**: Illustration of the solid angle s subtended by some area A. **Right**: The corresponding projected solid angle $s^{\perp}$.

The area on the unit sphere $\mathcal{S}^2$ subtended by an area A is known as a *solid angle*. This is the extension of the well known planar angle (on the 2D unit circle) to 3D space, see Figure 2.3 (left). Solid angles are measured in the dimensionless unit *steradian* [sr].

Integration over the spherical directions $\mathcal{S}^2$,

$$\int_{\mathcal{S}^2} f(\boldsymbol{\omega})\, \mathrm{d}\Omega(\boldsymbol{\omega}) := \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi)\; \sin\theta\, \mathrm{d}\theta\, \mathrm{d}\phi, \tag{2.4}$$

uses the *solid angle measure* $\mathrm{d}\Omega(\boldsymbol{\omega}) = \sin\theta\, \mathrm{d}\theta\, \mathrm{d}\phi$, where $\sin\theta$ is a Jacobian determinant term that accounts for compression along the poles of the unit sphere.

Another useful related measure is *projected solid angle* $\mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega}) = \mathrm{d}\Omega(\boldsymbol{\omega})\; |\cos\theta|$ shown in Figure 2.3 (right). Here, the solid angle is additionally projected onto the disk and θ is the angle between the surface normal $\mathbf{n}(\mathbf{x})$ at position $\mathbf{x}$ and the direction $\boldsymbol{\omega}$. Because both vectors are normalized, the cosine can be computed conveniently as the dot product $\cos\theta = \boldsymbol{\omega} \cdot \mathbf{n}(\mathbf{x})$.

2.1.2 Radiometric quantities

The following paragraphs define the relevant radiometric quantities for rendering, which are also illustrated in Figure 2.4.

Flux. The *Planck-Einstein* relation tells us that the energy Q (measured in *joule* [J]) carried by a single photon is inversely proportional to its wavelength λ:

$$Q = \frac{h\,c}{\lambda} \qquad [\mathrm{J}] \tag{2.5}$$

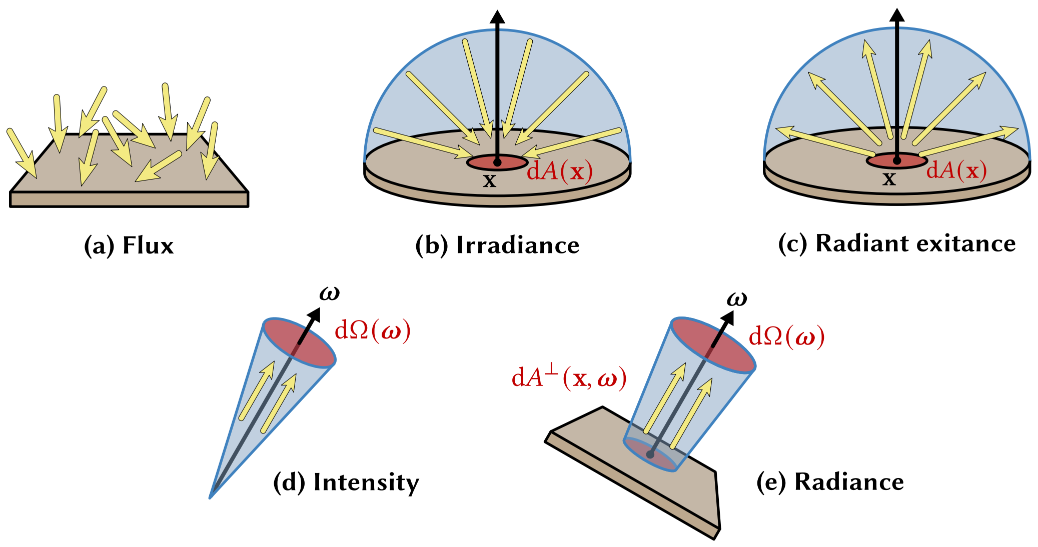


Figure 2.4: Illustration of the radiometric quantities defined in this section.

with the Planck constant $h \approx 6.63 \times 10^{-34}\,\mathrm{J \cdot s}$ and the speed of light $c \approx 3.00 \times 10^{8}\,\mathrm{m \cdot s^{-1}}$ measured in vacuum.

The amount of energy flowing through a finite surface A per unit time is known as *(radiant) flux*, see Figure 2.4 (a). Its unit is *watts* [W] or joules per second.

$$\Phi(A) = \frac{\mathrm{d}Q}{\mathrm{d}t} \qquad [\mathrm{W} = \mathrm{J} \cdot \mathrm{s}^{-1}] \tag{2.6}$$

Irradiance. The area density of flux arriving at an infinitesimal area around $\mathbf{x}$ is known as *irradiance*, see Figure 2.4 (b).

$$E(\mathbf{x}) = \frac{\mathrm{d}\Phi(A)}{\mathrm{d}A(\mathbf{x})} \qquad [\mathrm{W} \cdot \mathrm{m}^{-2}] \tag{2.7}$$

Radiant exitance. Analogously, the flux *leaving* a surface per unit area (due to emission or scattering) is *radiant exitance*, see Figure 2.4 (c).

$$M(\mathbf{x}) = \frac{\mathrm{d}\Phi(A)}{\mathrm{d}A(\mathbf{x})} \qquad [\mathrm{W} \cdot \mathrm{m}^{-2}] \tag{2.8}$$

In computer graphics, $M(\mathbf{x})$ is sometimes also referred to as *radiosity* $B(\mathbf{x})$.

Intensity. One can alternatively measure the directional density of flux that travels through an infinitesimal solid angle around $\boldsymbol{\omega}$. This is called *(radiant) intensity*, see Figure 2.4 (d).

$$I(\boldsymbol{\omega}) = \frac{\mathrm{d}\Phi(A)}{\mathrm{d}\Omega(\boldsymbol{\omega})} \qquad [\mathrm{W} \cdot \mathrm{sr}^{-1}] \tag{2.9}$$

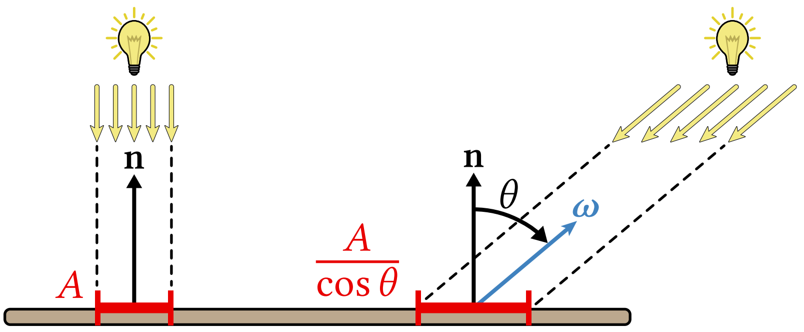


Figure 2.5: Illustration of Lambert's cosine law. **Left**: Light strikes a surface at perpendicular incidence and hits a patch of area A. **Right**: When arriving at an angle θ, the surface area of the patch is stretched.

Radiance. By taking the limit for both direction and area (perpendicular to $\boldsymbol{\omega}$) we arrive finally at the definition of *radiance*, see Figure 2.4 (e). It can be interpreted as intensity per perpendicular unit area or alternatively flux per unit solid angle and perpendicular area:

$$\begin{aligned} L(\mathbf{x},\boldsymbol{\omega}) &= \frac{\mathrm{d}I(\boldsymbol{\omega})}{\mathrm{d}A^{\perp}(\mathbf{x},\boldsymbol{\omega})} \\ &= \frac{\mathrm{d}^2\Phi(A)}{\mathrm{d}\Omega(\boldsymbol{\omega})\,\mathrm{d}A^{\perp}(\mathbf{x},\boldsymbol{\omega})} \\ &= \frac{\mathrm{d}^2\Phi(A)}{\mathrm{d}\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega})\,\mathrm{d}A(\mathbf{x})} \\ &= \frac{\mathrm{d}^2\Phi(A)}{\mathrm{d}\Omega(\boldsymbol{\omega})\,\mathrm{d}A(\mathbf{x})\,|\cos\theta|} \qquad [\mathrm{W}\cdot\mathrm{sr}^{-1}\cdot\mathrm{m}^{-2}] \end{aligned} \tag{2.10}$$

This also introduces a $\cos\theta$ factor that can either be interpreted as a projected area measure (perpendicularly to $\boldsymbol{\omega}$, i.e. $\mathrm{d}A^{\perp}(\mathbf{x},\boldsymbol{\omega}) = \mathrm{d}A(\mathbf{x})\,|\cos\theta|$), or as the projected solid angle measure $\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega})$ defined earlier.

This additional factor is also related to the well-known *Lambert cosine law* (Figure 2.5) that explains how light arriving at an angle θ is spread out, which scales the incident radiance by a factor $\cos\theta$. This is also known as the *foreshortening term*.

Radiance is the most frequently used quantity in computer graphics for two reasons:

1. All previously defined quantities can be derived in terms of radiance by integrating over surface areas or directions.

2. Radiance traveling along a direction through empty space[1] remains constant. This

[1]In this thesis we restrict ourselves to pure surface scattering. We do not cover the more general radiative transfer setting [16] where light might be scattered or absorbed while traversing *participating media* that occupy the space between surfaces.

means we can easily relate *incident radiance* L_i at a position $\mathbf{x}$ with *outgoing radiance* L_o at some other position $\mathbf{y}$ along $\boldsymbol{\omega}$ (or vice versa).

$$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = L_\mathrm{o}(\mathbf{y}, -\boldsymbol{\omega}) \tag{2.11}$$

Spectral quantities. The quantities above are usually accounting for the total energy integrated over all wavelengths. Note, however, that there exist *spectral* variants of them where an additional limit over λ is taken. For example, *spectral radiance* is defined as

$$L(\mathbf{x}, \boldsymbol{\omega}, \lambda) = \frac{\mathrm{d}L(\mathbf{x}, \boldsymbol{\omega})}{\mathrm{d}\lambda} \tag{2.12}$$

and its integrated version can be recovered as $L(\mathbf{x}, \boldsymbol{\omega}) = \int_0^\infty L(\mathbf{x}, \boldsymbol{\omega}, \lambda)\, \mathrm{d}\lambda$.

In practice, this distinction is not relevant for the covered topics as we do not simulate effects like fluorescence or dispersion that have a dependence on wavelength. In other words, we can simply simulate each wavelength of light separately.

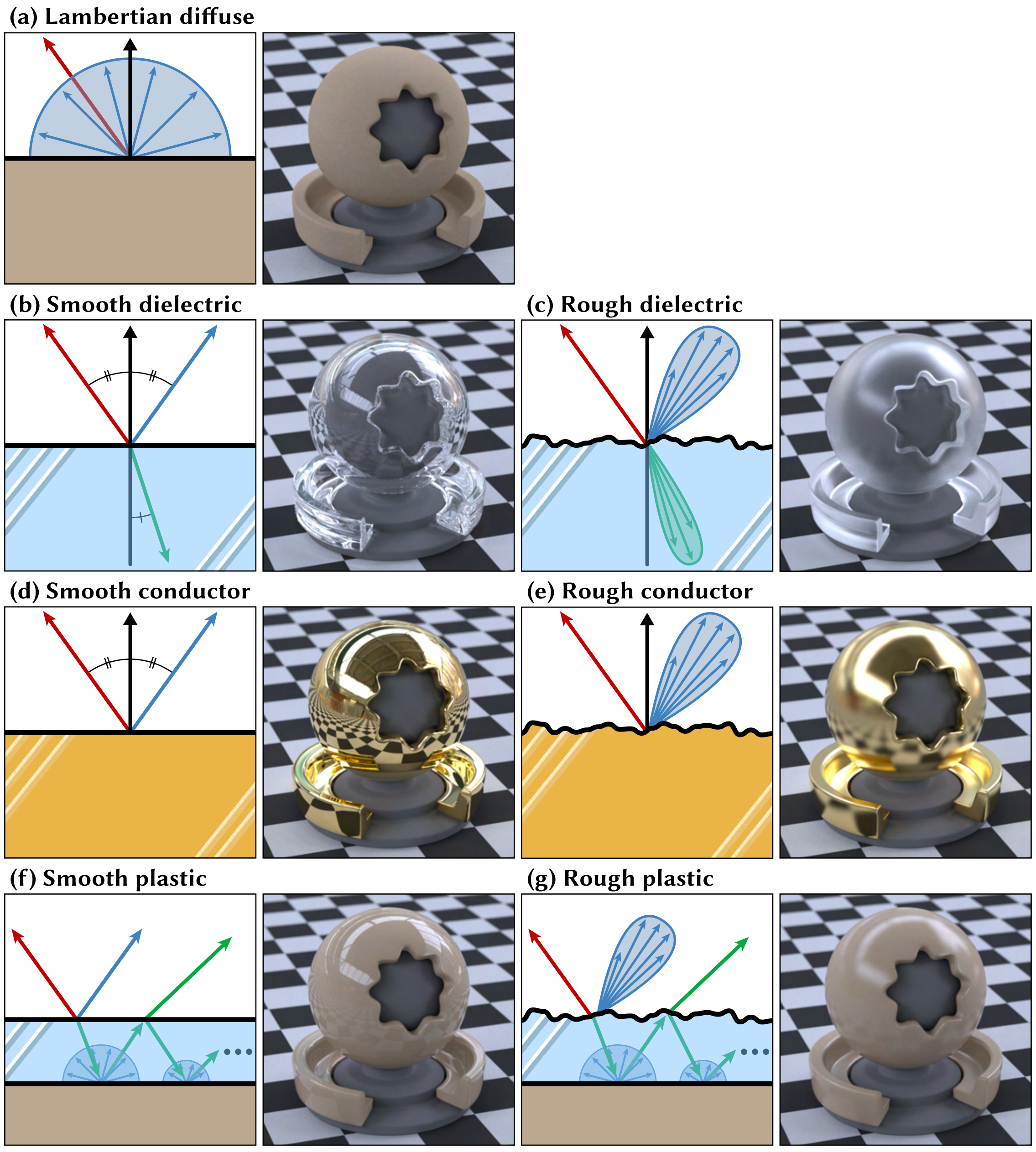
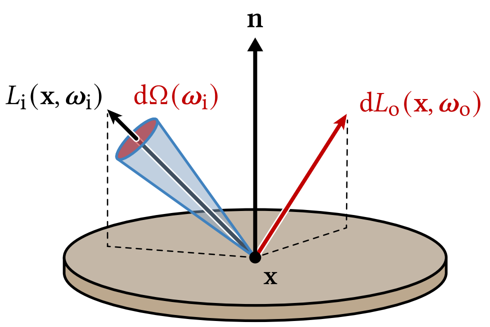


Figure 2.6: Overview of common material models found in a rendering system. **(a)** The diffuse model scatters light into all directions. **(b, d)** Specular surfaces behave based on the laws of reflection and refraction. **(c, e)** Rough materials cause glossy reflections and refractions due to a bumpy microstructure. **(f, g)** Layered combinations of a diffuse substrate with either a smooth or rough dielectric coating. This also causes multiple-scattering between the base layer and the coating.



$\mathbf{n}$
$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})$
$\mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i})$
$\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})$
$\mathbf{x}$

Figure 2.7: Geometric configuration used in the definition of the BSDF.

2.2 Material models

The creation of photorealistic images also entails simulating a diverse set of materials. Their appearance is ultimately a result of light interacting with surfaces, either in the form of reflection or transmission. Whereas some materials act like mirrors with sharp reflections, others are matte and scatter light more uniformly.

In rendering systems, this complexity is encapsulated by various *material models.*

2.2.1 Bidirectional scattering distribution function (BSDF)

The *bidirectional scattering distribution function (BSDF)* is the mathematical description of light interaction with a given material and quantifies how much of it scatters from a direction $\boldsymbol{\omega}_\mathrm{i}$ to another direction $\boldsymbol{\omega}_\mathrm{o}$ at surface position $\mathbf{x}$. Its exact definition is the ratio of differential outgoing radiance to differential incident irradiance

$$f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{\mathrm{d}E_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})} = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\,|\cos\theta_\mathrm{i}|\,\mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i})} \quad [\mathrm{sr}^{-1}]. \tag{2.13}$$

We use the arrow notation "$\to$" to indicate the direction of light flow. The corresponding geometric configuration is illustrated in Figure 2.7.

Physically based BSDFs require a few important properties to hold:

Positivity. Scattering cannot lead to a case where "negative light" is reflected.

$$f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) \geq 0 \tag{2.14}$$

Energy conservation. A surface cannot reflect more light than it receives.

$$\int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\,|\cos\theta_\mathrm{i}|\,\mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i}) \leq 1 \tag{2.15}$$

Reciprocity. The flow of light is symmetric between emission and observer. This means the BSDF should be invariant to interchanging its $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$ arguments. This is also known as *Helmholtz reciprocity* [17].

$$f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i}) \tag{2.16}$$

Sometimes it is more appropriate to use the expressions *BRDF*[2] f_r and *BTDF*[3] f_t to make the explicit distinction between light reflected to the upper, or transmitted to the lower hemisphere respectively, where $f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) + f_t(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})$.

In the following sections we will cover the most common classes of BSDFs that are also illustrated in Figure 2.6.

2.2.2 Lambertian diffuse

The most simple BSDF is one that scatters light uniformly into the upper hemisphere, as illustrated in Figure 2.6 (a). Hence, there is only a constant BRDF component without transmission ($f_t = 0$):

$$f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = \frac{\rho}{\pi}. \tag{2.17}$$

This is referred to as a *Lambertian* or simply *diffuse* BRDF, where ρ is the diffuse albedo that controls how much of the incident light is reflected: a value $\rho = 1$ means all light is reflected whereas $\rho = 0$ describes a purely absorptive material. Usually, materials have varying reflectance at different wavelengths which ultimately results in their perceived color. The scale factor $1/\pi$ comes from the energy conservation condition above and makes sure no extra energy is created in the process. It can be verified by integrating the (cosine weighted) BRDF over the upper hemisphere $\mathcal{H}^2 \subset \mathcal{S}^2$:

$$\int_{\mathcal{H}^2} f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) \, |\cos\theta_\mathrm{i}| \, \mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i}) = \frac{\rho}{\pi} \int_{\mathcal{H}^2} |\cos\theta_\mathrm{i}| \, \mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i}) = \rho. \tag{2.18}$$

Note that this BSDF is mostly of theoretical interest and real-world materials with a matte appearance behave quite differently at grazing and retro-reflective angle configurations as they are the result of scattering on micro geometry or *subsurface scattering* inside the material. In practice, more plausible models [18, 19, 20, 21] are available.

[2]Bidirectional *reflectance* distribution function.

[3]Bidirectional *transmittance* distribution function.

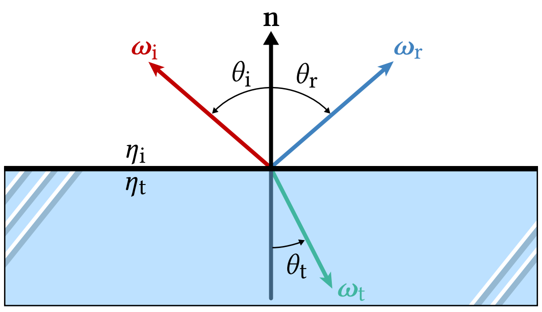


Figure 2.8: Geometric configuration of light arriving at a dielectric interface between IORs η_i and η_t. The incident direction $\boldsymbol{\omega}_i$ is split into two new directions $\boldsymbol{\omega}_r$ and $\boldsymbol{\omega}_t$ corresponding to reflection and refraction.

2.2.3 Smooth dielectric

The interaction of electromagnetic radiation with matter differs significantly between conducting (e.g. metals) and non-conducting (*dielectric*) materials. We will first cover the dielectric case before discussing conductors in Section 2.2.4.

Scattering from specular (perfectly smooth) surfaces forms the foundation of many more general material models today and therefore it is advantageous to first look at this basic case.

In general, dielectrics cause light to both reflect and refract[4], as shown in Figure 2.6 (b) and Figure 2.8.

Specular scattering. The scattering direction of light is determined via the laws of specular reflection and refraction that are analytic solutions to Maxwell's equations. The latter of the two is also known as the *Sahl-Snell law*.

$$\textbf{Reflection}: \quad \theta_i = \theta_r \tag{2.19}$$

$$\textbf{Refraction}: \quad \eta_i \sin\theta_i = \eta_o \sin\theta_o \tag{2.20}$$

These relate an incident angle θ_i with its reflected (θ_r) and refracted (θ_t) counterparts based on the *indices of refraction* (IORs) η_i and η_t on both sides of the scattering interface. The associated geometric configuration is illustrated in Figure 2.8.

These IORs describe how fast light propagates inside different media. Vacuum has an IOR of 1.0 (by definition) whereas most typical solids have an IOR around 1.5. Some

[4]In practice, not all dielectrics necessarily have a noticeable transmission component as their interiors can be filled with absorptive particles or the dielectric is applied as a coating on some substrate, see Section 2.2.6. In such cases it is usually sufficient to only model their reflection component.

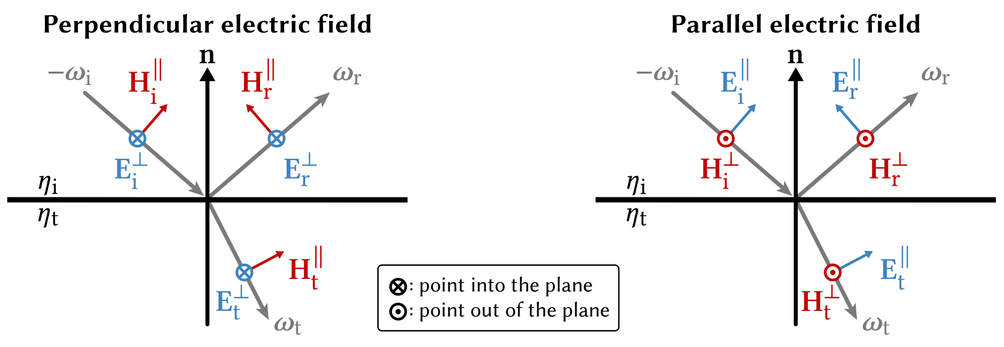


Figure 2.9: The electromagnetic fields used for the derivation of the Fresnel equations where the electric fields **E** are either perpendicular (**left**) or parallel (**right**) to the plane of incidence. The corresponding magnetic fields **H** are orthogonal to **E** and the light propagation direction.

materials (e.g. diamond, $\eta \approx 2.4$) can be notably denser, whereas most liquids (e.g. water, $\eta \approx 1.33$) are less dense[5].

Equations (2.19) and (2.20) can also be written in vector form which leads to a more straightforward implementation without trigonometric functions:

$$\mathrm{S}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta) = \begin{cases} \mathrm{S}^\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}), & \text{if reflection,} \\ \mathrm{S}^\mathrm{t}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta), & \text{if transmission,} \end{cases} \quad \text{where} \tag{2.21}$$

$$\mathrm{S}^\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}) = 2\,(\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n})\,\mathbf{n} - \boldsymbol{\omega}_\mathrm{i}, \tag{2.22}$$

$$\mathrm{S}^\mathrm{t}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta) = -\eta\,\left(\boldsymbol{\omega}_\mathrm{i} - (\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n})\,\mathbf{n}\right) - \mathbf{n}\sqrt{1 - \eta^2(1 - (\boldsymbol{\omega}_\mathrm{i} \cdot \mathbf{n})^2)}. \tag{2.23}$$

Here we additionally switched to a *relative* index of refraction $\eta = \eta_\mathrm{i}/\eta_\mathrm{t}$ for convenience.

Fresnel equations. Having defined the reflected and refracted directions, it is not yet clear how the incident radiance is split up between the two. This is quantified by the famous *Fresnel equations* [22]. Even though we had previously stated that we would ignore effects of polarization in this thesis (Section 2.1), this part requires some awareness of the underlying wave nature of light.

In particular, the equations are derived separately for two different kinds of *linearly polarized* light where the electric field **E** is aligned either perpendicularly ("⊥") or parallel ("||") to the plane of incidence[6] (defined by $\boldsymbol{\omega}_\mathrm{i}$ and $\mathbf{n}$). The situation is shown in Figure 2.9.

[5]Technically, the IOR is also a function of the light's frequency which causes light to "split" into a continuum of directions based on its wavelength. This is called *dispersion* and is most commonly known from light interacting with a prism or with water droplets in the air which will cause a rainbow. In most scenarios, the effect of dispersion is limited and therefore usually ignored in computer graphics. This is equivalent to associating a constant IOR with each dielectric material.

[6]There is an arbitrary decision here with respect to which vectors the electromagnetic fields should be

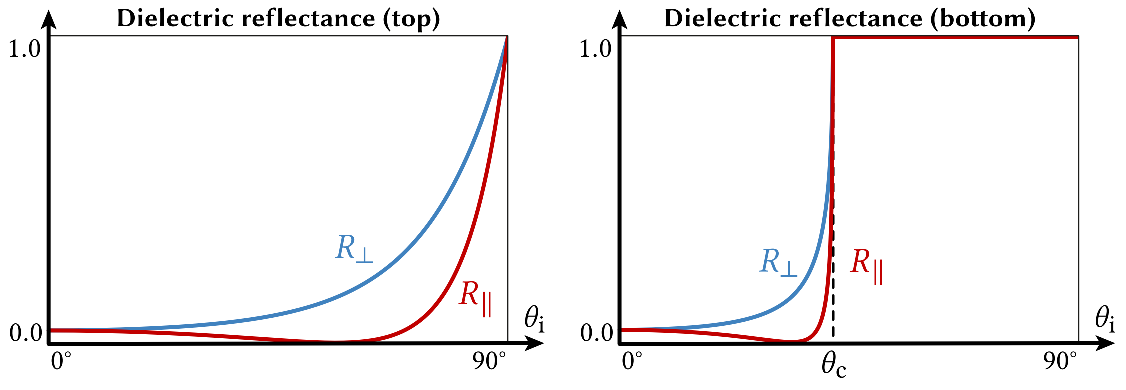


Figure 2.10: The Fresnel equations predict how much light is reflected at a dielectric interface from a particular incident angle θ_i in the case of perpendicular or parallel waves. **Left**: Light going from air to glass ($\eta_i \approx 1.0, \eta_t \approx 1.5$). **Right**: The opposite configuration where light leaves the glass. Total internal reflection takes place after the critical angle θ_c.

The Fresnel equations then relate the incident and reflected electric fields via

$$r_\perp(\theta_i, \eta_i, \eta_t) = \frac{\mathbf{E}_r^\perp}{\mathbf{E}_i^\perp} = \frac{\eta_i \cos\theta_i - \eta_t \cos\theta_t}{\eta_i \cos\theta_i + \eta_t \cos\theta_t} = -\frac{\sin(\theta_i - \theta_t)}{\sin(\theta_i + \theta_t)} \tag{2.24}$$

$$r_{||}(\theta_i, \eta_i, \eta_t) = \frac{\mathbf{E}_r^{||}}{\mathbf{E}_i^{||}} = \frac{\eta_t \cos\theta_i - \eta_i \cos\theta_t}{\eta_t \cos\theta_i + \eta_i \cos\theta_t} = +\frac{\tan(\theta_i - \theta_t)}{\tan(\theta_i + \theta_t)} \tag{2.25}$$

where $\cos\theta_t$ can be computed from the law of refraction as $\cos\theta_t = \sqrt{1 - (\eta_i/\eta_o)^2 \sin^2\theta_i}$. Note that, due to the expression in the square root, the two resulting *reflection amplitude coefficients* $r_\perp, r_{||}$ can be complex-valued and also encode the potential *phase shift* of the polarized wave.

Finally, the *reflectance* (i.e. how much of the radiance is reflected at a particular incident angle) is given by the squared magnitudes of the coefficients:

$$R_\perp(\theta_i, \eta_i, \eta_t) = |r_\perp(\theta_i, \eta_i, \eta_t)|^2 \tag{2.26}$$

$$R_{||}(\theta_i, \eta_i, \eta_t) = \left|r_{||}(\theta_i, \eta_i, \eta_t)\right|^2 . \tag{2.27}$$

Figure 2.10 shows typical curves for both $R_\perp$ and $R_{||}$ plotted over all incident angles. Note that for light scattering at a less dense medium (e.g. when scattering from glass to air), these clearly show the *critical angle* $\theta_c = \arcsin(\eta_t/\eta_i)$ after which no more light is transmitted. This phenomenon is known as *total internal reflection.*

Because we ultimately do not care about the actual polarization state here, we can now average the reflectances for the two field orientations, which gives us the final

measured exactly. This leads to different conventions for the positive/negative signs of the resulting Fresnel equations below. See Hecht [23] for a complete discussion of their derivation.

Fresnel term known in computer graphics

$$F(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}) = \frac{1}{2}\Big(R_\perp(\theta_\mathrm{i}, \eta_\mathrm{i}, \eta_\mathrm{t}) + R_\parallel(\theta_\mathrm{i}, \eta_\mathrm{i}, \eta_\mathrm{t})\Big), \tag{2.28}$$

where the explicit IOR arguments are usually omitted for brevity and the angle θ_i is the angle between incident direction $\boldsymbol{\omega}_\mathrm{i}$ and the normal direction $\mathbf{n}$. Note that this gives us at the same time the expression for *transmittance* as $1 - F$ due to energy conservation.

BSDF definition. With all these details out of the way, we can now write down the definitions of the BRDF and BTDF components of the smooth dielectric BSDF

$$f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = F(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}) \frac{\delta_\Omega(\boldsymbol{\omega}_\mathrm{o}, \mathrm{S}^\mathrm{r}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}))}{|\cos\theta_\mathrm{i}|} \tag{2.29}$$

$$f_t(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = (1 - F(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n})) \frac{\delta_\Omega(\boldsymbol{\omega}_\mathrm{o}, \mathrm{S}^\mathrm{t}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \frac{\eta_\mathrm{i}}{\eta_\mathrm{o}}))}{|\cos\theta_\mathrm{i}|} \frac{\eta_\mathrm{o}^2}{\eta_\mathrm{i}^2}, \tag{2.30}$$

where the $\cos\theta_\mathrm{i}$ terms cancel out the same foreshortening term in the BSDF definition (2.13) as it is already accounted for by the Fresnel equations.

Because light can only scatter into two fixed directions, these contain a *Dirac delta function* $\delta_\Omega(\boldsymbol{\omega}, \boldsymbol{\omega}')$ in the solid angle measure. Despite its name, this is not a function in the conventional meaning but is only meant to evaluate to a value when being integrated. In the case of functions g over the spherical domain,

$$\int_{S^2} g(\boldsymbol{\omega}')\ \delta_\Omega(\boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\Omega(\boldsymbol{\omega}') = g(\boldsymbol{\omega}), \tag{2.31}$$

where the delta function vanishes everywhere except at $\boldsymbol{\omega} = \boldsymbol{\omega}'$.

One last detail is the $\eta_\mathrm{o}^2/\eta_\mathrm{i}^2$ term that appears in f_t which describes the fact that radiance is compressed into a smaller solid angle when refracted into a medium. Interestingly, refraction does not obey the reciprocity condition from Equation (2.16). To properly restore the symmetry, Veach [7] pointed out that these scale factors need to be included as well:

$$\frac{f_t(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})}{\eta_\mathrm{o}^2} = \frac{f_t(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})}{\eta_\mathrm{i}^2}. \tag{2.32}$$

2.2.4 Smooth conductor

Similarly to the dielectric case, we can also accurately describe *conductor* (i.e. metal) materials, e.g. as shown in Figure 2.6 (d). While the law of reflection (2.19) still holds, their reflectance properties are quite different due to electrons that move freely on the atomic

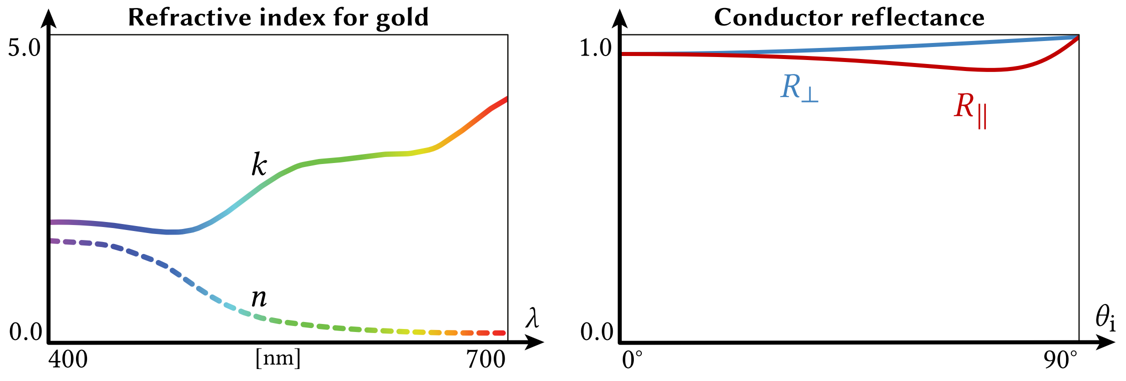


Figure 2.11: **Left**: The two components of the complex refractive index $\eta = n - ik$ for gold that shows a strong wavelength dependence. **Right**: The corresponding reflectance curves predicted by the Fresnel equations at 633nm.

structure and interact with the incident electromagnetic radiation. It is remarkable however that the Fresnel equations (2.24) and (2.25) are still sufficient to describe this process by characterizing conductors with a complex-valued index of refraction $\eta = n - ik$. The smooth conductor BRDF therefore has the same form as Equation (2.29) and we assume no transmission[7] into the metal ($f_t = 0$).

The use of complex numbers for IORs might seem odd at first but this is merely a mathematical tool to describe the absorption inside of the metal. It comes from the expression of a plane wave propagating through a medium

$$W(t) = \exp(-i\,t\,\alpha\,\eta) \tag{2.33}$$

with increasing time t and a constant α related to the wave frequency. When introducing some imaginary component $k > 0$, this now leads to an exponential decay of the wave amplitude as it travels deeper into the medium:

$$W_{\text{absorbing}}(t) = \exp(-i\,t\,\alpha\,(n - ik)) \tag{2.34}$$

$$= \underbrace{\exp(-i\,t\,\alpha\,n)}_{\text{same as in real case}} \cdot \underbrace{\exp(-t\,\alpha\,k)}_{\text{exponential decay}}. \tag{2.35}$$

Compared to dielectrics, the IORs of conductors have a strong wavelength dependence which ultimately is the cause of their colored appearance. Figure 2.11 shows the IOR of gold plotted over the visible spectrum and the resulting Fresnel reflectance curves for one particular wavelength.

[7]This is a safe assumption because light usually travels only very short distances ($< 1\mu$m) into metals before being fully absorbed. Additionally, θ_t, as computed from the law of refraction, is now a complex-valued expression and loses its previous interpretation as the refracted angle. The correct expression can be found in the optics literature [24, 25] but is not commonly used in rendering.

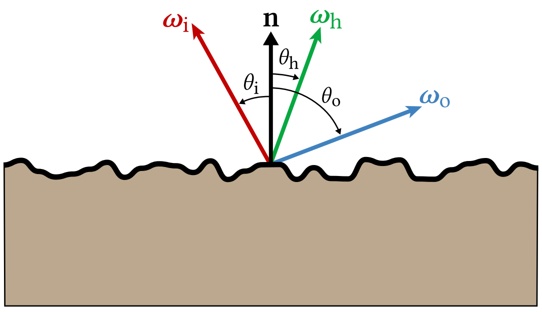


Figure 2.12: Vectors and corresponding angles involved in BSDF models based on microfacet theory. Light interacts with a rough material (that has an overall *macroscopic* surface normal $\mathbf{n}$) along incident and outgoing directions $\omega_{\mathrm{i}}, \omega_{\mathrm{o}}$. The half-vector ω_{h} is the orientation of a microfacet that would lead to perfectly specular reflection from ω_{i} to ω_{o}.

2.2.5 Microfacet models

The previous sections covered very idealized material classes that lie on extreme ends of a continuum: light is either reflected or refracted into a discrete set of directions, or scattered uniformly into the hemisphere. In reality however, most materials fall somewhere in between these two cases and have a glossy appearance, for example as shown in Figure 2.6 (c, e).

Early models for this in computer graphics relied on ad-hoc approaches [26] while the seminal work of Beckmann and Spizzichino [27] and Torrance and Sparrow [28] investigated accurate scattering of light on rough surfaces by studying reflections caused by many randomly oriented mirror-like *facets*. These can be interpreted as tiny bumps or details in the actual surface and could, in theory, also be modeled as part of the scene geometry, see Figure 2.12. When facets are sufficiently small (i.e. the surface appears smooth and no details can be perceived by the eye), we can instead turn to statistical averages to model their aggregated scattering behavior in a much more efficient way[8]. This model was introduced to graphics in form of the famous Cook-Torrance BRDF [29]

$$f_r(\mathbf{x}, \omega_{\mathrm{i}} \to \omega_{\mathrm{o}}) = \frac{F(\omega_{\mathrm{i}}, \omega_{\mathrm{h}})\, D(\omega_{\mathrm{h}})\, G(\omega_{\mathrm{i}}, \omega_{\mathrm{o}}, \omega_{\mathrm{h}})}{4\,|\cos\theta_{\mathrm{i}}| \cdot |\cos\theta_{\mathrm{o}}|} \tag{2.36}$$

that is now understood in the larger framework of *microfacet theory* and has proven successful due to its realistic appearance and its ability to match real world measurements surprisingly well.

The expression includes various terms that we will discuss shortly: F is the Fresnel reflectance, D is the *normal distribution function (NDF)* that characterizes the rough surface, and G is a *shadowing-masking* term.

[8]We will come back to the other case with a visible (specular) microgeometry later in Section 3.1.5.

The analogous refraction component (BTDF) to Equation (2.36) was later derived by Stam [30] and Walter et al. [31]:

$$f_t(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}}) = \frac{|\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{h}}| \cdot |\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{\omega}_{\mathrm{h}}|}{|\cos\theta_{\mathrm{i}}| \cdot |\cos\theta_{\mathrm{o}}|} \frac{\eta_{\mathrm{o}}^2 (1 - F(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{h}})) D(\boldsymbol{\omega}_{\mathrm{h}}) G(\boldsymbol{\omega}_{\mathrm{i}}, \boldsymbol{\omega}_{\mathrm{o}}, \boldsymbol{\omega}_{\mathrm{h}})}{(\eta_{\mathrm{i}}(\boldsymbol{\omega}_{\mathrm{i}} \cdot \boldsymbol{\omega}_{\mathrm{h}}) + \eta_{\mathrm{o}}(\boldsymbol{\omega}_{\mathrm{o}} \cdot \boldsymbol{\omega}_{\mathrm{h}}))^2}. \tag{2.37}$$

Today, almost all physically based BSDFs use microfacet models in some form, see Section 2.2.6.

In both components, an important quantity is the half-vector direction

$$\boldsymbol{\omega}_{\mathrm{h}} := \frac{\eta_{\mathrm{i}}\boldsymbol{\omega}_{\mathrm{i}} + \eta_{\mathrm{o}}\boldsymbol{\omega}_{\mathrm{o}}}{||\eta_{\mathrm{i}}\boldsymbol{\omega}_{\mathrm{i}} + \eta_{\mathrm{o}}\boldsymbol{\omega}_{\mathrm{o}}||}, \tag{2.38}$$

which is written here in its generalized form [31] including refractive indices in order to be compatible with refraction and reflection (where $\eta_{\mathrm{i}} = \eta_{\mathrm{o}}$). It can be interpreted as the orientation of a microfacet (i.e. a *microfacet normal*) that causes reflection (or refraction) along the two directions $\boldsymbol{\omega}_{\mathrm{i}}$ and $\boldsymbol{\omega}_{\mathrm{o}}$ for which the BSDF is evaluated. The reflection case is illustrated in Figure 2.12.

Fresnel reflectance F. This term was previously discussed in Section 2.2.3. Based on the choice of refractive indices (real or complex-valued), we can model both dielectric or conductive materials via the same expression.

In the rough case, the half-vector $\boldsymbol{\omega}_{\mathrm{h}}$ takes the role of the surface normal to establish a suitable coordinate frame for evaluating the Fresnel equations.

Normal distribution function D. The NDF is the statistical distribution of microfacet normal orientations on the rough surface. An important constraint is that its projection onto the macroscopic surface normal $\mathbf{n}$ has to be normalized:

$$\int_{S^2} D(\mathbf{m})\ (\mathbf{m} \cdot \mathbf{n})\, \mathrm{d}\Omega(\mathbf{m}) = 1. \tag{2.39}$$

Evaluating D based on the half-vector $\boldsymbol{\omega}_{\mathrm{h}}$ in Equations (2.36) and (2.37), can intuitively be understood as measuring how much of the microfacet area is aligned to result in specular reflection (or refraction).

The two most popular choices for NDFs in computer graphics are the *Beckmann* [27] and *GGX* distributions [31, 32]:

$$D^{\text{Beck.}}(\mathbf{m}) = \chi^+(\mathbf{m} \cdot \mathbf{n}) \frac{e^{-\frac{\tan^2\theta_{\mathrm{m}}}{\alpha^2}}}{\pi\alpha^2\cos^4\theta_{\mathrm{m}}} \tag{2.40}$$

$$D^{\text{GGX}}(\mathbf{m}) = \chi^+(\mathbf{m} \cdot \mathbf{n}) \frac{\alpha^2}{\pi\cos^4\theta_{\mathrm{m}}(\alpha^2 + \tan^2\theta_{\mathrm{m}})^2} \tag{2.41}$$

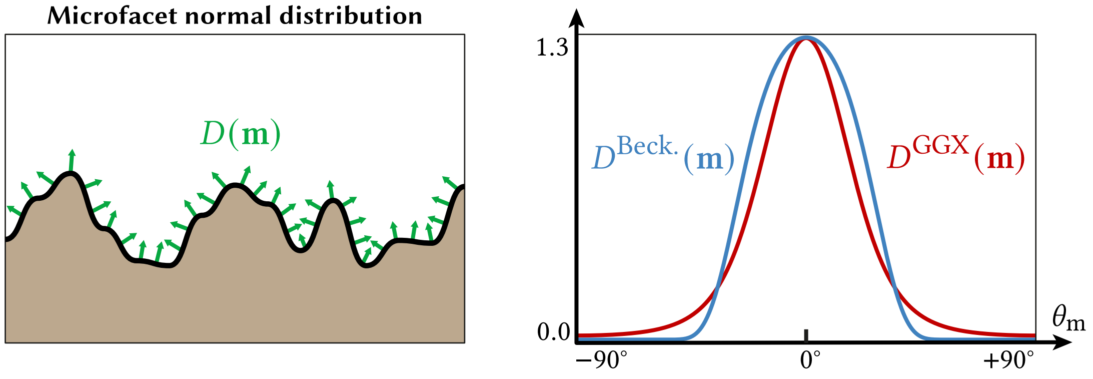


Figure 2.13: **Left**: A visualization of the normal distribution function (NDF) on a hypothetical rough surface. **Right**: Beckmann and GGX NDFs plotted over microfacet elevation angles θ_{m}. The wider tails of the GGX distribution are clearly visible. Both functions are evaluated for roughness $\alpha = 0.5$ here.

where the Heaviside function $\chi^+(x)$ is 1 if $x > 0$ and 0 otherwise; it constrains the microfacet normals $\mathbf{m}$ to vectors on the same side as the macroscopic normal $\mathbf{n}$.

Both NDFs are parametrized with a surface roughness parameter α that models the variation of the microfacet normals. It is related to the slope standard deviation σ of the microfacets via $\alpha = \sqrt{2}\sigma$. Note that the Beckmann distribution is based on a Gaussian distribution of microfacet slopes and GGX is the distribution of normals on an ellipsoid. Both options behave similarly but GGX is considered more plausible nowadays due to its wider tails that match many real measured materials more closely. See Figure 2.13 for an illustration of the NDF and a plot that contrasts the two distributions with each other.

There also exist several generalizations of these models in form of additional shape control [19, 33], anisotropic variants [34, 35] and off-centered distributions [36, 37].

Shadowing-masking G. In the BSDF model we also have to consider that not all microfacets will be visible from any given incident or outgoing direction $\boldsymbol{\omega}_{\text{i}}$ or $\boldsymbol{\omega}_{\text{o}}$. Instead, some will be occluded behind other facets, see Figure 2.14 (a). The solution to this is *Smith's shadowing-masking function* $G_1(\boldsymbol{\omega}, \mathbf{m})$ [38] which measures the amount of microfacets with orientation $\mathbf{m}$ that are visible from $\boldsymbol{\omega}$. For obvious reasons, this function also depends on the underlying choice of the NDF. It can be computed as

$$G_1(\boldsymbol{\omega}, \mathbf{m}) = \chi^+(\boldsymbol{\omega} \cdot \mathbf{m}) \frac{\cos\theta}{\int_{\mathcal{S}^2} D(\mathbf{m}) \langle \boldsymbol{\omega} \cdot \mathbf{m} \rangle \, \mathrm{d}\Omega(\mathbf{m})} \tag{2.42}$$

with $\langle \mathbf{a} \cdot \mathbf{b} \rangle$ denoting a clamped dot product that is zero for $\mathbf{a} \cdot \mathbf{b} < 0$.

Fortunately, analytic solutions to this are available for both Beckmann and GGX

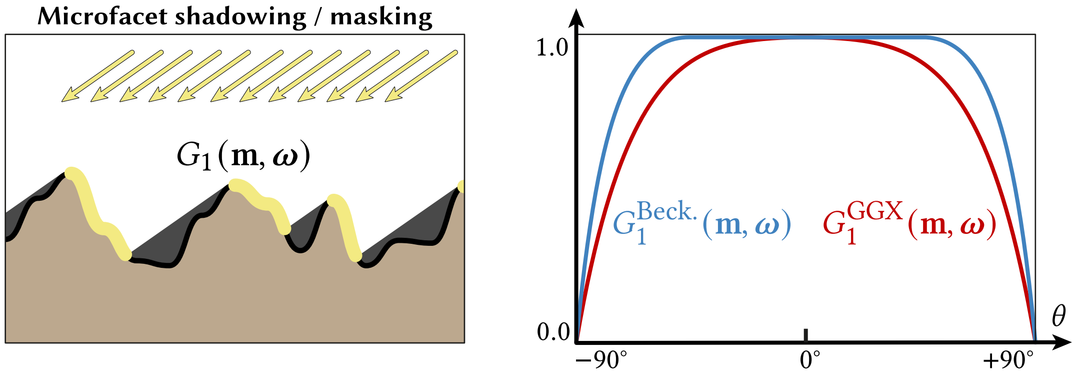


Figure 2.14: **Left**: The shadowing (and masking) functions express how much of the microfacet surface is illuminated (or visible) from a given direction $\boldsymbol{\omega}$. **Right**: Comparison of the shadowing function derived from the Beckmann and GGX NDFs (at $\alpha = 0.5$), plotted over the elevation angle of the incident direction.

distributions:

$$G_1^{\text{Beck.}}(\boldsymbol{\omega}, \mathbf{m}) = \chi^+(\boldsymbol{\omega} \cdot \mathbf{m}) \frac{2}{1 + \operatorname{erf}(a) + \frac{1}{a\sqrt{\pi}} e^{-a^2}} \tag{2.43}$$

$$G_1^{\text{GGX}}(\boldsymbol{\omega}, \mathbf{m}) = \chi^+(\boldsymbol{\omega} \cdot \mathbf{m}) \frac{2}{1 + \sqrt{1 + \frac{1}{a^2}}} \tag{2.44}$$

with the error function $\operatorname{erf}(x) = {}^2\!/\!\sqrt{\pi} \int_0^x e^{-t} dt$ and $a := (\alpha \tan \theta)^{-1}$ Again, these two are contrasted against each other in Figure 2.14 (b).

The final term G in the BSDF model should account for shadowing and masking from both directions $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$. A simple option is to assume that these two events are uncorrelated with each other. In that case,

$$G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \mathbf{m}) = G_1(\boldsymbol{\omega}_\mathrm{i}, \mathbf{m})\, G_1(\boldsymbol{\omega}_\mathrm{o}, \mathbf{m}). \tag{2.45}$$

A more accurate option is to take the correlation due to microfacet heights into account. The higher up a facet lies in the microstructure, the more likely it is unoccluded by another one. This results in the slightly more complex expression

$$G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \mathbf{m}) = \frac{G_1(\boldsymbol{\omega}_\mathrm{i}, \mathbf{m})\, G_1(\boldsymbol{\omega}_\mathrm{o}, \mathbf{m})}{G_1(\boldsymbol{\omega}_\mathrm{i}, \mathbf{m}) + G_1(\boldsymbol{\omega}_\mathrm{o}, \mathbf{m}) - G_1(\boldsymbol{\omega}_\mathrm{i}, \mathbf{m})\, G_1(\boldsymbol{\omega}_\mathrm{o}, \mathbf{m})}. \tag{2.46}$$

Note that the shadowing-masking term effectively reduces the standard microfacet BSDFs to a *single-scattering* model and all light removed by G would instead scatter again between multiple microfacets. In practice, this leads to a severe energy loss issue especially at high roughness values. We ignore this problem here but note that some approaches exist that tackle this issue [39, 40, 41].

Derivations and full discussions of all terms in this section are out of the scope of this thesis. Please refer to Heitz' excellent technical report [35] for all details regarding shadowing-masking and microfacet theory in general.

2.2.6 Layered materials

While the models in Figure 2.6 (a–e) are well understood and accurate for their respective use cases, they only represent a relatively small set of materials found in the real world and most typical surfaces exhibit mixtures of different scattering types.

A popular approach to handle this inside of a physically based rendering system is to model the general case as *layered materials*, i.e. a stack of different BSDFs each with their own scattering properties and indices of refraction. Figure 2.6 (f–g) for instance show the common combination of a diffuse substrate with either smooth or rough dielectric coatings applied on top of it. These are sometimes referred to as "plastic-like" materials and can plausibly approximate a large number of real-world materials such as ceramics, car paint, finished woods, or different types of paint.

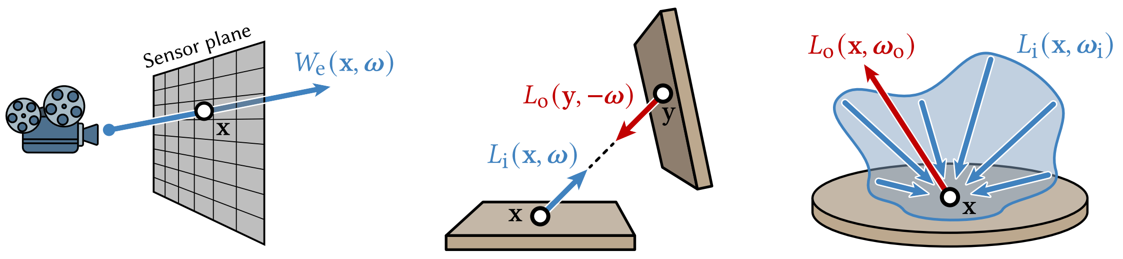
In its most basic interpretation, layering is just a linear combination of several individual BSDFs. But from a more physical perspective, the contribution of each layer is highly dependent on its Fresnel reflectance (and thus varies with the incident angle) and there can be a considerable amount of multiple-scattering inside and between the layers themselves that causes changes in color saturation and perceived roughness.

Approaches for layered materials are roughly split into three groups:

1. Analytic models which only approximate the real effect of multiple-scattering but are also usable in real-time applications [42, 43, 44].

2. A discretized Fourier basis representation of the BSDFs that allows the use of the *adding-doubling method* [45] to precompute the layered material before rendering [3, 46]. These approaches are accurate but can be very memory intensive.

3. Light scattering through a stack of layers can also be simulated as a random walk [47, 48, 49, 50]. This is very general and does not require any extra memory, but instead causes additional variance due to the stochastic nature of the simulation.


Sensor plane
$W_e(\mathbf{x}, \boldsymbol{\omega})$
x
$L_o(\mathbf{y}, -\boldsymbol{\omega})$
y
$L_i(\mathbf{x}, \boldsymbol{\omega})$
x
$L_o(\mathbf{x}, \boldsymbol{\omega}_o)$
$L_i(\mathbf{x}, \boldsymbol{\omega}_i)$
x

(a) Measurement **(b) Transport** **(c) Scattering**

Figure 2.15: Three fundamentals steps of light transport simulation. **(a)** Measurement converts incident radiance into recorded pixel values on the film of a conceptual sensor. **(b)** Transport moves radiation between surfaces, and **(c)** scattering models the surface's response to incident illumination.

2.3 Light transport

Following the discussion of light (Section 2.1) and materials (Section 2.2), we can now turn to the description of light transport that we aim to simulate during the rendering process. This can be summarized with three equations illustrated in Figure 2.15: *measurement*, *transport*, and *scattering*. Together, they explain how light is emitted from light sources, propagates through a scene, and is ultimately collected by a sensor.

We will discuss the equations from the perspective of three formulations using different parameterizations of the problem domain, either with directional integrals over solid angles (Section 2.3.1), area integrals over scene surfaces (Section 2.3.2), or finally an integral over a general space of light paths (Section 2.3.3).

Light transport satisfies a central symmetry condition, which states that the role of light source and sensor can be exchanged without changing the actual measurement. We briefly remark on this in Section 2.3.4.

2.3.1 Solid angle formulation

Measurement. We use the *measurement equation* by Nicodemus [51] to compute the final pixel values $I_1, \ldots, I_n$ in an image. In particular, the value[9] of pixel k is

$$I_k = \int_{\mathcal{M}} \int_{\mathcal{S}^2} W_e^{(k)}(\mathbf{x}, \boldsymbol{\omega}) \, L_i(\mathbf{x}, \boldsymbol{\omega}) \, d\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega}) \, dA(\mathbf{x}) \tag{2.47}$$

which is a product integral of the incident radiance L_i arriving at a sensor position $\mathbf{x}$ from direction $\boldsymbol{\omega}$ times the *importance function* $W_e^{(k)}$. The importance describes the sensitivity

[9]This is a scalar quantity that results in monochromatic images. Alternatively, we can solve for multiple independent color channels or wavelengths here.

profile of pixels on the film of the simulated sensor and depends on the sensor's optical system. As a result, we are integrating over both incident directions and sensor positions, though $W_\mathrm{e}^{(k)}$ is usually non-zero for only a small region around a given pixel k.

Transport. Like previously mentioned in Section 2.1.2, radiance stays constant along a ray of light, assuming there is no blocking obstacle or volumetric interaction. This means that we can easily convert between incident and outgoing radiance at two points using the *transport equation*

$$L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = L_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega}). \tag{2.48}$$

Here, $\mathbf{r}(\mathbf{x}, \boldsymbol{\omega})$ is the *ray tracing operator* that returns the closest surface position found along a ray with origin $\mathbf{x}$ pointing in direction $\boldsymbol{\omega}$.

Scattering. The way light scatters on surfaces follows an energy balance equation between emission, absorption, or reflection/refraction based on the BSDF. From the earlier definition of the BSDF (2.13) we can directly derive the scattering equation by rearranging terms and then integrating over solid angles:

$$f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}) = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, |\cos\theta_\mathrm{i}|\; \mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i})}$$

$$\Leftrightarrow$$

$$f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, |\cos\theta_\mathrm{i}| = \frac{\mathrm{d}L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o})}{\mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i})}$$

$$\Leftrightarrow$$

$$\int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, |\cos\theta_\mathrm{i}|\; \mathrm{d}\Omega(\boldsymbol{\omega}_\mathrm{i}) = L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) \tag{2.49}$$

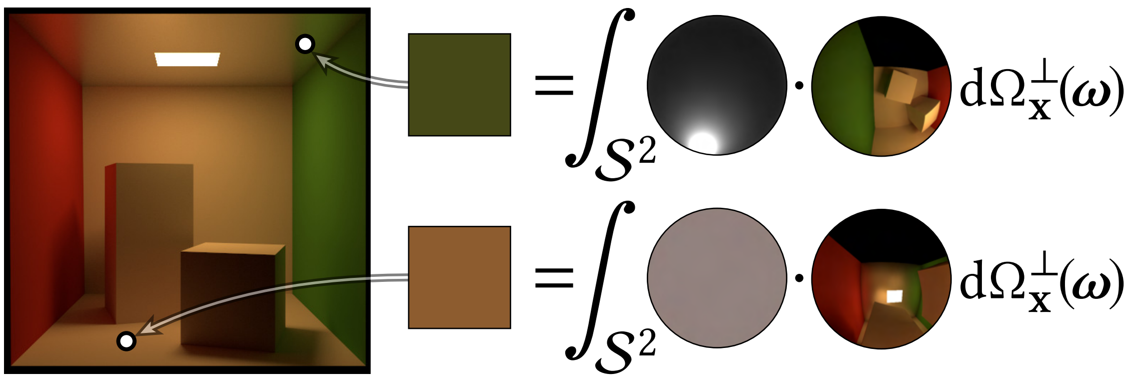
This yields the famous *rendering equation* by Kajiya [52] where an added term L_e captures any *emitted radiance* directly at the surface interaction, and integration now takes place in the projected solid angle measure:

$$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega}_\mathrm{i}). \tag{2.50}$$

This makes intuitive sense: the outgoing radiance in direction $\boldsymbol{\omega}_\mathrm{o}$ is the sum of directly visible emission L_e at $\mathbf{x}$ and all incident radiance visible from that point weighted by the BSDF. The effect of this product integral is visualized in Figure 2.16.

Another common formulation uses the transport equation (2.48) to substitute the incident radiance inside the integral with another outgoing radiance term:

$$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}), -\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega}_\mathrm{i}). \tag{2.51}$$



$= \int_{S^2}$
$\cdot$
$d\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega})$
$= \int_{S^2}$
$\cdot$
$d\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega})$

Figure 2.16: The radiance that a scene location reflects towards the camera can be computed with the rendering equation: a product integral involving the BSDF at that point and the incident radiance. We illustrate this for two points in the Cornell box scene with glossy **(top)** or diffuse **(bottom)** materials. We visualize the upper hemispheres of the spherical functions (relative to the surface normal) projected onto the disk.

This reveals that the rendering equation has a self-referential form[10], meaning each radiance term inside the integral is again the result of the same integral.

This observation is the basis of many solution techniques to light transport that we will discuss in Section 2.5.

2.3.2 Area formulation

Integrals often admit equivalent parameterizations on other domains, which can be more convenient from a numerical or computational point of view. One such example is accounting for the reflected radiance due to an emissive area light (Section 2.5.1) where it is more appropriate to directly integrate over its surface.

This is also known as the *three-point form* of the rendering equation

$$L(\mathbf{x}'\to\mathbf{x}) = L_e(\mathbf{x}'\to\mathbf{x}) + \int_{\mathcal{M}} f_s(\mathbf{x}''\to\mathbf{x}'\to\mathbf{x})\, L(\mathbf{x}''\to\mathbf{x}')\, G(\mathbf{x}'\leftrightarrow\mathbf{x}'')\, dA(\mathbf{x}''), \tag{2.52}$$

where all terms are parameterized based on surface positions and, similarly to the previous definition of the BSDF, the arrows "$\to$" denote the direction of light flow. More precisely, the radiance and BSDF are rewritten as $L(\mathbf{x}\to\mathbf{y}) := L_o(\mathbf{x},\boldsymbol{\omega})$ with (normalized) vector $\boldsymbol{\omega} = \overrightarrow{\mathbf{xy}}$ and $f_s(\mathbf{x}\to\mathbf{y}\to\mathbf{z}) := f_s(\mathbf{y},\boldsymbol{\omega}_i\to\boldsymbol{\omega}_o)$ with $\boldsymbol{\omega}_i := \overrightarrow{\mathbf{yx}}$ and $\boldsymbol{\omega}_o := \overrightarrow{\mathbf{yz}}$. See also the diagram in Figure 2.17.

The geometric term G is a symmetric function that combines a visibility function[11]

[10]To be precise, it is a Fredholm integral equation of the second kind.

[11]This was not necessary in the previous solid angle formulation because it was encoded implicitly via the ray tracing step in the transport equation.

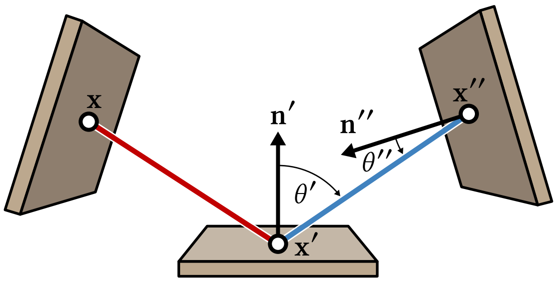


Figure 2.17: Geometric configuration of the rendering equation in its three-point form. The geometric term $G(\mathbf{x}'\leftrightarrow\mathbf{x}'')$ captures the relative orientation (θ', θ'') and squared distance between $\mathbf{x}', \mathbf{x}''$.

$V(\mathbf{x}'\leftrightarrow\mathbf{x}'')$, which evaluates to 1 in case $\mathbf{x}'$ and $\mathbf{x}''$ are mutually visible and 0 otherwise, and a conversion factor between the projected solid angle and area measures

$$\begin{aligned} G(\mathbf{x}'\leftrightarrow\mathbf{x}'') &= V(\mathbf{x}'\leftrightarrow\mathbf{x}'')\left|\frac{\mathrm{d}\Omega^{\perp}_{\mathbf{x}'}(\overrightarrow{\mathbf{x}'\mathbf{x}''})}{\mathrm{d}A(\mathbf{x}'')}\right| \\ &= V(\mathbf{x}'\leftrightarrow\mathbf{x}'')\frac{|\cos\theta'|\,|\cos\theta''|}{\|\mathbf{x}'-\mathbf{x}''\|^2}. \end{aligned} \tag{2.53}$$

As shown in the diagram, θ' and θ'' refer to the angles between the light direction and the surface normals at positions $\mathbf{x}'$ and $\mathbf{x}''$ respectively.

Exactly the same change of variables can also be applied to the measurement equation

$$I_k = \int_{\mathcal{M}^2} W_{\mathrm{e}}^{(k)}(\mathbf{x}'\to\mathbf{x})\, L(\mathbf{x}'\to\mathbf{x})\, G(\mathbf{x}\leftrightarrow\mathbf{x}')\, \mathrm{d}A(\mathbf{x}')\, \mathrm{d}A(\mathbf{x}), \tag{2.54}$$

this time using $W_{\mathrm{e}}^{(k)}(\mathbf{x}\to\mathbf{y}) := W_{\mathrm{e}}^{(k)}(\mathbf{y}, \boldsymbol{\omega})$ with $\boldsymbol{\omega} = \overrightarrow{\mathbf{y}\mathbf{x}}$.

2.3.3 Path integral formulation

The *path space integral* formulation introduced by Veach [7], is particularly useful when designing more advanced light transport algorithms. Instead of computing nested integrals over solid angles or surface areas, we directly integrate over full light transport paths connecting sensors and light sources. A light path here refers to a sequence of straight line segments representing a possible trajectory of light. We define the set of all light paths of length n (meaning n path segments and $n + 1$ vertices) as

$$\mathcal{P}_n := \{\mathbf{x}_0 \ldots \mathbf{x}_n \,|\, \mathbf{x}_0, \ldots, \mathbf{x}_n \in \mathcal{M}\}, \tag{2.55}$$

where the starting vertex $\mathbf{x}_0$ is a sensor position and the last vertex $\mathbf{x}_n$ lies on a light source as shown in Figure 2.18.

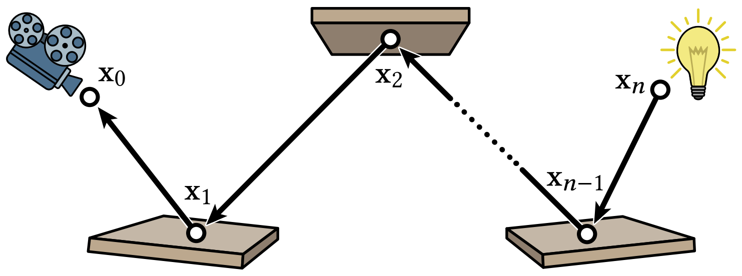


Figure 2.18: An example light path $\bar{\mathbf{x}} = \mathbf{x}_0 \ldots \mathbf{x}_n$ of length n. We use the convention that the initial vertex $\mathbf{x}_0$ is located on the sensor whereas the last vertex $\mathbf{x}_n$ lies on a light source.

Path space $\mathcal{P}$ is then the union of all sets representing paths of any length:

$$\mathcal{P} = \bigcup_{n=1}^{\infty} \mathcal{P}_n \tag{2.56}$$

and we use overlined symbols $\bar{\mathbf{x}} = \mathbf{x}_0 \ldots \mathbf{x}_n \in \mathcal{P}$ to denote entire light paths.

Starting with the measurement equation (2.54) and recursively expanding the radiance term using the rendering equation (2.52) we arrive at an infinite sum of integrals of increasing dimensionality that corresponds to the number of light bounces through the scene:

$$\begin{aligned}
I_k = &\int_{\mathcal{M}^2} L_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_0)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_\mathrm{e}^{(k)}(\mathbf{x}_1 \to \mathbf{x}_0)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_0) \\
&+ \int_{\mathcal{M}^3} L_\mathrm{e}(\mathbf{x}_2 \to \mathbf{x}_1)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, f_s(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) \\
&\qquad\qquad \cdot G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_\mathrm{e}^{(k)}(\mathbf{x}_1 \to \mathbf{x}_0)\, \mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_0) \\
&+ \ \ldots \\
&= \sum_{j=1}^{\infty} \int_{\mathcal{M}^{j+1}} L_\mathrm{e}(\mathbf{x}_j \to \mathbf{x}_{j-1})\, G(\mathbf{x}_{j-1} \leftrightarrow \mathbf{x}_j) \left[\prod_{i=1}^{j-1} f_s(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})\, G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right] \\
&\qquad\qquad \cdot W_\mathrm{e}^{(k)}(\mathbf{x}_1 \to \mathbf{x}_0)\, \mathrm{d}A(\mathbf{x}_j) \ldots \mathrm{d}A(\mathbf{x}_0).
\end{aligned} \tag{2.57}$$

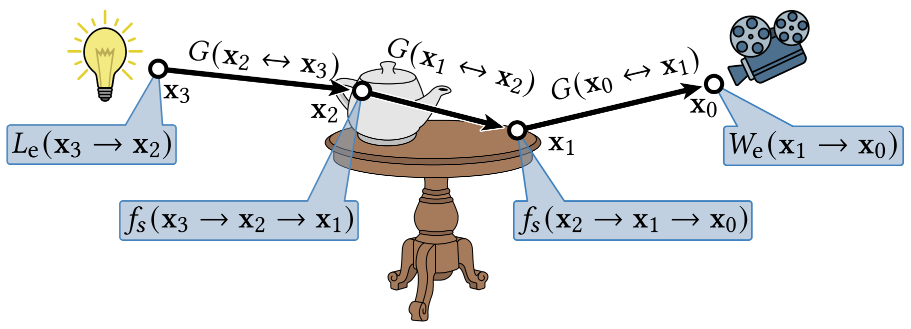
Alternatively, this can be cast into a more concise notation

$$I_k = \int_{\mathcal{P}} f^{(k)}(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{2.58}$$

where f is the path contribution and $\mathrm{d}\mu(\bar{\mathbf{x}})$ refers to the product measure derived from the area measures $\mathrm{d}A(\mathbf{x})$ over $\mathcal{M}$.

The contribution for a given light path $\bar{\mathbf{x}}$ is then

$$f^{(k)}(\bar{\mathbf{x}}) = f^{(k)}(\mathbf{x}_0 \ldots \mathbf{x}_n) = L_\mathrm{e}(\mathbf{x}_n \to \mathbf{x}_{n-1})\, T(\bar{\mathbf{x}})\, W_\mathrm{e}^{(k)}(\mathbf{x}_1 \to \mathbf{x}_0) \tag{2.59}$$



$G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3)$
$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$
$G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)$
$\mathbf{x}_3$
$\mathbf{x}_2$
$\mathbf{x}_1$
$\mathbf{x}_0$
$L_e(\mathbf{x}_3 \to \mathbf{x}_2)$
$W_e(\mathbf{x}_1 \to \mathbf{x}_0)$
$f_s(\mathbf{x}_3 \to \mathbf{x}_2 \to \mathbf{x}_1)$
$f_s(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0)$

Figure 2.19: The light path contribution is the product of emitted radiance L_e, BSDF values f_s, geometric terms G, and the sensor importance W_e.

where the *path throughput* is defined as

$$T(\bar{\mathbf{x}}) = G(\mathbf{x}_{n-1} \leftrightarrow \mathbf{x}_n) \left[\prod_{i=1}^{n-1} f_s(\mathbf{x}_{i+1} \to \mathbf{x}_i \to \mathbf{x}_{i-1})\, G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right]. \tag{2.60}$$

Figure 2.19 shows the individual path contribution terms for a simple (length 3) light path $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ through a scene.

We will revisit the path space formulation later in Section 3.1.1 in the context of *specular* light paths where at least one of the path vertices is associated with a purely specular reflective or refractive BSDF involving a Dirac delta distribution.

2.3.4 Symmetry of light transport

When examining the terms in Equations (2.59) and (2.60) we realize they are all symmetric with respect to the direction of light flow, with a few notable exceptions where the BSDF terms f_s are non-symmetric:

1. As discussed earlier, refractive BSDFs are only reciprocal when adding another scaling factor (2.32) related to the indices of refraction.
2. The use of shading normals that deviate from the true object geometry can also break symmetry. Veach [7] describes this in more details and outlines the necessary scaling factors to avoid the issue.
3. Simulating effects like polarization or fluorescence involves generalized forms of the BSDF that are inherently non-symmetric [53].

Apart from these cases, the pixel intensity I_k will be unchanged when exchanging the sensor and light source [17]. This means we can equivalently view the importance $W_\mathrm{e}^{(k)}$

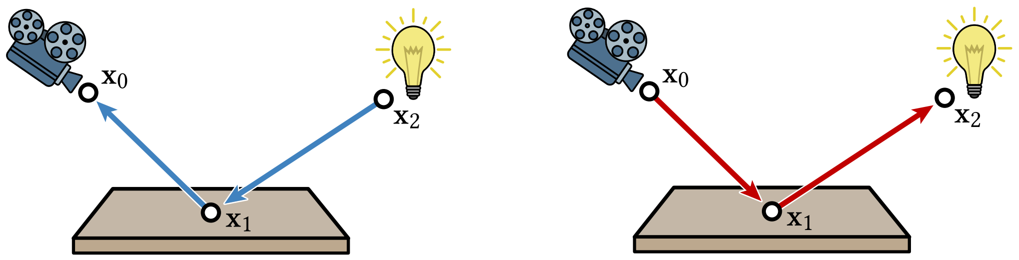


Figure 2.20: There are two symmetric interpretations of light transport. **(a)** Radiance is emitted from light sources and is eventually received by a sensor. **(b)** Importance is emitted from a sensor and received by light sources.

as an "emitted"[12] quantity that is sent out into the scene from the camera, see Figure 2.20. Importance then scatters the same as radiance, until it is eventually "received" by a light source. We can therefore, analogously to Section 2.3.1, define the relevant light transport equations based on an energy balance between incident and outgoing importance W_i, W_o:

$$\textbf{Measurement}\text{: } I_k = \int_{\mathcal{M}} \int_{\mathcal{S}^2} L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega})\, W_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}) \tag{2.61}$$

$$\textbf{Transport}\text{: } W_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = W_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega}) \tag{2.62}$$

$$\textbf{Scattering}\text{: } W_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = W_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \int_{\mathcal{S}^2} f_\mathrm{s}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o} \to \boldsymbol{\omega}_\mathrm{i})\, W_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega}_\mathrm{i}) \tag{2.63}$$

Not only is this symmetry exploited by many efficient rendering techniques (Section 2.5.4) but we will later see that a very similar idea is the key for designing efficient *differentiable rendering* techniques in the context of inverse rendering (Section 4.1.2).

[12]The subscript "e" is in fact chosen to convey this other role as an emitted quantity.

2.4 Monte Carlo integration

As alluded to previously, most rendering techniques used today rely at least partially on *Monte Carlo (MC)* integration methods. As this is a randomized simulation process, it is appropriate to first briefly review the basics of probability theory and statistics in Section 2.4.1. We then introduce the basic Monte Carlo estimator in Section 2.4.2 which approximates a given integration problem stochastically. While it determines the true value of the integral in expectation, the result is is contaminated by additional variance. We therefore also discuss some of the most well known variance reduction techniques from the Monte Carlo literature in Sections 2.4.3 to 2.4.6.

2.4.1 Basics of probability theory

We limit ourselves to a concise summary of the probability theory concepts that are relevant for the remainder of Section 2.4. For a more thorough introduction to the topic, and the related measure theory, see for example Billingsley [54].

Cumulative distribution and probability density functions. The *cumulative distribution function (CDF)* of a random variable $X \in \mathcal{X}$ is

$$P(\mathcal{D}) = \text{Prob}\{X \in \mathcal{D}\}, \tag{2.64}$$

i.e. the probability that X lies in a measurable subset $D \subset \mathcal{X}$. By taking the *Radon-Nikodym* derivative we then arrive at

$$p(\mathbf{x}) = \frac{\mathrm{d}P}{\mathrm{d}\mu}(\mathbf{x}), \tag{2.65}$$

which is the *probability density function (PDF)* associated with the CDF. Equivalently, this can be viewed as the function that gives back $P(\mathcal{D})$ when integrating it over $\mathcal{D}$:

$$P(\mathcal{D}) = \int_{\mathcal{D}} p(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}). \tag{2.66}$$

The definition of the PDF is thus tightly coupled with a corresponding measure $\mathrm{d}\mu(\mathbf{x})$. In simple euclidean spaces $\mathbb{R}^n$, this can be understood as measuring length ($\mathcal{X} = \mathbb{R}$), area ($\mathcal{X} = \mathbb{R}^2$), or volumes ($\mathcal{X} = \mathbb{R}^3$), but generalizations also exist for other spaces. An important space for defining probability densities in rendering is the unit sphere ($\mathcal{X} = \mathcal{S}^2$) with the associated *solid angle measure* encountered already in Section 2.1.1.

Expected value and variance of random variables. If the random variable X is distributed according to density $p(\mathbf{x})$, the *expected value* or *mean* of any function $f(X)$ is defined as the integral

$$\mathrm{E}[f(X)] = \int_{\mathcal{X}} f(\mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}). \tag{2.67}$$

Similarly, its *variance* is given as another integral

$$\begin{aligned}\mathrm{Var}[f(X)] &= \int_{\mathcal{X}} (f(\mathbf{x}) - \mathrm{E}[f(X)])^2\, p(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) \\ &= \mathrm{E}[(f(\mathbf{x}) - \mathrm{E}[f(X)])^2],\end{aligned} \tag{2.68}$$

where sometimes also the *standard deviation* or *root mean square error* $\sigma[X] = \sqrt{\mathrm{Var}[X]}$ is used.

The expected value is a linear operator, meaning it satisfies

$$\mathrm{E}[a\,X + b\,Y] = a\,\mathrm{E}[X] + b\,\mathrm{E}[Y] \tag{2.69}$$

for any two random variables X, Y and constants a, b. Note that, only when X and Y are *independent* variables, the additional relation $\mathrm{E}[X \cdot Y] = \mathrm{E}[X] \cdot \mathrm{E}[Y]$ holds.

It is also worth mentioning a few counterexamples where an expectation does not commute with a nonlinear transformation:

$$\mathrm{E}[1/X] \neq 1/\mathrm{E}[X] \tag{2.70}$$

$$\mathrm{E}[\exp(X)] \neq \exp(\mathrm{E}[X]). \tag{2.71}$$

The variance does not behave linearly but instead we have the simple rule

$$\mathrm{Var}[a\,X + b\,Y] = a^2\,\mathrm{Var}[X] + b^2\,\mathrm{Var}[Y] + 2ab\,\mathrm{Cov}[X, Y], \tag{2.72}$$

where the *covariance* is defined as

$$\mathrm{Cov}[X, Y] = \mathrm{E}[(X - \mathrm{E}[X])\,(Y - \mathrm{E}[Y])]. \tag{2.73}$$

With the identities above, we can also rewrite the definition of variance as a more convenient expression

$$\mathrm{Var}[X] = \mathrm{E}[X^2] - \mathrm{E}[X]^2. \tag{2.74}$$

Joint, marginal, and conditional probability densities. Given two random variables $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ with measures $\mathrm{d}\mu_X(\mathbf{x})$, $\mathrm{d}\mu_Y(\mathbf{y})$, their joint density $p(\mathbf{x}, \mathbf{y})$ is defined such that its integral gives the probability that $(X, Y) \in \mathcal{D}$ for some $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$:

$$P(\mathcal{D}) = \mathrm{Prob}\{(X, Y) \in \mathcal{D}\} = \int_{\mathcal{D}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mu_X(\mathbf{x})\, \mathrm{d}\mu_Y(\mathbf{y}). \tag{2.75}$$

When working with multivariate distributions we often also care about a *marginal distribution*

$$p(\mathbf{x}) = \int_{\mathcal{Y}} p(\mathbf{x}, \mathbf{y})\, \mathrm{d}\mu_Y(\mathbf{y}), \tag{2.76}$$

where the integration over one of the function arguments is referred to as *marginalization.* The *conditional density* gives the density of Y given some known realization of X:

$$p(\mathbf{y} \,|\, \mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x})}. \tag{2.77}$$

In case of independent X and Y this simplifies to $p(\mathbf{y} \,|\, \mathbf{x}) = p(\mathbf{y})$ and we thus have $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}) \cdot p(\mathbf{y})$.

2.4.2 Monte Carlo estimator

The core goal of Monte Carlo integration [55, 56, 57] is the computation of an integral

$$I = \int_{\mathcal{X}} f(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) \tag{2.78}$$

of the function $f(\mathbf{x})$ with $\mathbf{x} \in \mathcal{X}$. In many cases, the function f is not amenable to analytic approaches, and we must resort to numerical approximation. Monte Carlo methods in particular achieve this in a stochastic way: the integral is sampled at various random locations and the corresponding integrand values are aggregated into an *estimate* of the true integral. This is another random variable that matches the true integral I in expectation, but is subject to some variance.

The most basic one-sample Monte Carlo estimator is given by

$$\langle I \rangle = \frac{f(\mathbf{x})}{p(\mathbf{x})}, \tag{2.79}$$

where sample $\mathbf{x}$ is drawn from the density $p(\mathbf{x})$ and the ratio $f(\mathbf{x})/p(\mathbf{x})$ is required to be finite whenever $f(\mathbf{x}) \neq 0$. It is easy to see that the expected value of this is then indeed the desired integral:

$$\begin{aligned} \mathrm{E}\left[\langle I \rangle\right] &= \mathrm{E}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right] \\ &= \int_{\mathcal{X}} \frac{f(\mathbf{x})}{p(\mathbf{x})}\, p(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) \\ &= \int_{\mathcal{X}} f(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) \\ &= I. \end{aligned} \tag{2.80}$$

In practice, we will mostly rely on the multi-sample estimator

$$\langle I \rangle_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}, \tag{2.81}$$

which is simply the average of N separate realizations of Equation (2.79). Note that the samples $\mathbf{x}_i$ are usually assumed to be statistically independent of each other, though *sample stratification* or *low-discrepancy sequences* [55, 56, 57] can be effective ways to lower variance in many applications, including rendering.

To understand the general convergence rate of Equation (2.81), i.e. how quickly the estimation error goes down when increasing N, we can analyze the estimator variance

$$\begin{aligned} \mathrm{Var}\left[\langle I \rangle_N\right] &= \mathrm{Var}\left[\frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] \\ &= \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}\left[\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] \\ &= \frac{1}{N} \mathrm{Var}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right], \end{aligned} \tag{2.82}$$

which gives us an expression of the root mean square error

$$\sigma\left[\langle I \rangle_N\right] = \frac{1}{\sqrt{N}} \sigma\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right]. \tag{2.83}$$

The variance and error therefore decrease with rates of $O(1/N)$ and $O(1/\sqrt{N})$ respectively. This is generally considered to be quite slow (the number of samples needs to be quadrupled in order to reduce the error by half), especially compared to simpler quadrature techniques that can converge much more rapidly in low-dimensional 1D or 2D settings. However, a big advantage of Monte Carlo integration is that its convergence rate does not suffer from the "curse of dimensionality" and does not scale with the dimension of the problem[13]. We will see later that rendering can be interpreted as solving an

[13]Technically, there are some regularity conditions on the integrand for this to hold, see Donoho [58].

infinite-dimensional integral which makes Monte Carlo the only suitable choice.

The standard Monte Carlo estimator is known to be *unbiased* because, as shown above, its expected value matches the true value of the integral:

$$\text{Bias}\left[\langle I \rangle\right] = \text{E}\left[\langle I \rangle\right] - I = 0. \tag{2.84}$$

We will later also see occurrences of *biased* estimators where this is not the case. In practice, some bias can be acceptable, especially when the estimator is also *consistent*, meaning its approximation error tends to zero with probability one when increasing the number of samples:

$$\lim_{N\to\infty} \text{Prob}\left\{ I - \langle I \rangle_N = 0 \right\} = 1. \tag{2.85}$$

See Hachisuka [59] for a simple overview on bias and consistency specifically in computer graphics that also includes a discussion of common misconceptions.

When the integration domain is the unit-hypercube $[0,1)^d$, we commonly use a uniform sampling density $p(\mathbf{x}) = 1$. Then, Equation (2.81) simplifies to

$$\langle I \rangle_N^{(\text{uniform})} = \frac{1}{N} \sum_{i=1}^{N} f(\mathbf{x}_i). \tag{2.86}$$

We can considerably reduce the estimator variance however when adapting $p(\mathbf{x})$ to the integrand, a concept known as *importance sampling*[14].

2.4.3 Importance sampling

Let us consider an idealized estimator where we sample based on a density function p that is proportional to f up to a constant scale c to ensure the density is properly normalized:

$$p(\mathbf{x}) = c \cdot f(\mathbf{x}). \tag{2.87}$$

The estimate after taking a single sample is then simply a constant value with zero variance:

$$\langle I \rangle = \frac{f(\mathbf{x})}{p(\mathbf{x})} = \frac{f(\mathbf{x})}{c \cdot f(\mathbf{x})} = \frac{1}{c}. \tag{2.88}$$

[14]There is an unfortunate overlap here between terminology of *importance sampling* and the previously encountered *importance function* in Section 2.3.1 that defines a camera sensor response.

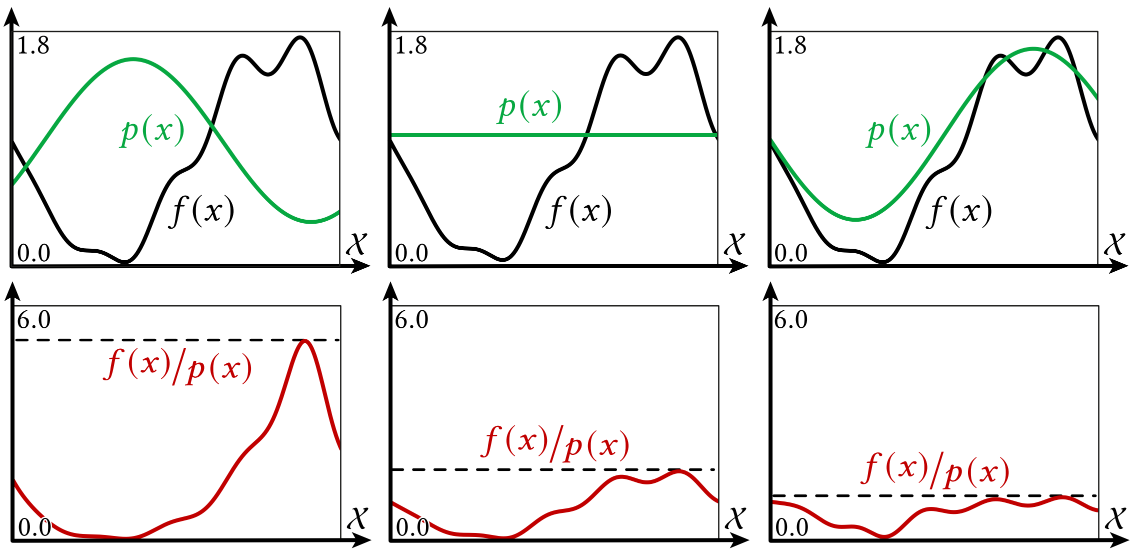


Figure 2.21: **Top**: Illustration of importance sampling a 1D function $f(x)$ using three different sampling densities $p(x)$. **Bottom**: The resulting sample weights f/p (red line) and its maximum value (dashed black line). **Left**: Poorly chosen p which results in large variance. **Middle**: Uniform sampling. **Right**: Well chosen p that causes f/p to be more constant and thus produces lower variance.

Though this is great in theory, unfortunately the task of finding the right constant c in this case of course requires us to compute the same integral we wanted to compute in the first place:

$$c = \frac{1}{\int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})}. \tag{2.89}$$

Nonetheless, this insight gives us a clear objective to keep variance minimal: the sampling density p should match f as closely as possible in order for the sample weights f/p to be as constant as possible. We illustrate this idea in Figure 2.21 for three different densities p.

This previous observation implicitly assumed the integrand f to be positive valued but unfortunately it does not apply to signed functions. Even a sampling density p that is perfectly proportional to the absolute value $|f(\mathbf{x})|$ will still exhibit potentially high variance due to sign changes in the estimate. While this is usually not a concern in forward rendering techniques (there is no “negative light” in physically based light transport), this can be a serious problem for differentiable rendering methods where the derivatives of the rendering process cause the integrand to also have negative parts. Specialized techniques exist for this more general case [57].

We now turn to a question that we completely glossed over so far: how can we actually generate samples from a given probability density function? In rendering, there are three main approaches that we will discuss in turn.

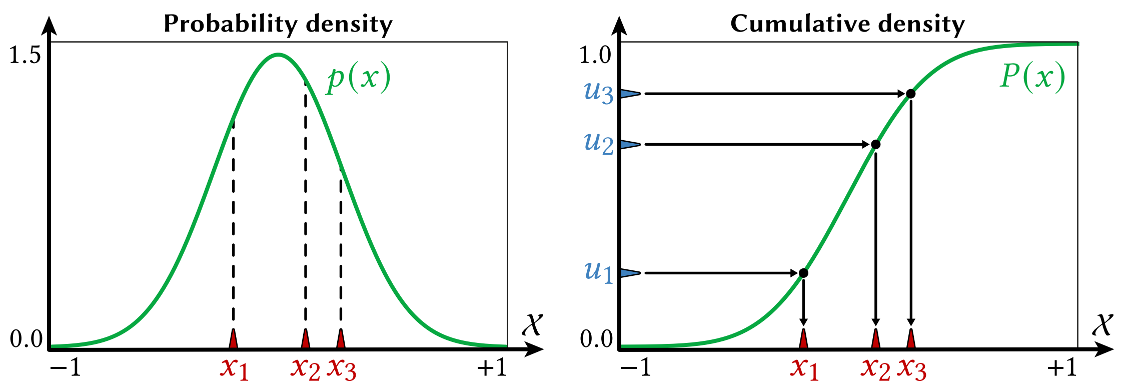


Figure 2.22: Inverse transform sampling generates samples x_1, x_2, x_3 from the PDF $p(x)$ (**left**) by mapping uniform variates u_1, u_2, u_3 through the inverse of the corresponding CDF $P(x)$ (**right**).

Inverse transform sampling. This process, also known as *inversion method* [60] samples from a target density $p(\mathbf{x})$ with d dimensions in three steps:

1. Compute the corresponding CDF $P(\mathbf{x})$ from $p(\mathbf{x})$ via integration.
2. Invert the function to arrive at the transformation $\mathbf{x} = T(\mathbf{u}) = P^{-1}(\mathbf{u})$
3. Evaluate $T(\mathbf{u})$, that maps a uniformly random sample $\mathbf{u}$ in $\mathcal{U} = [0,1)^d$ through the inverse transform.

The correctness of the method follows directly from the definition of the CDF (2.64). For example, the 1D CDF of a random variable X on $\mathbb{R}$ is

$$P(x) = \text{Prob}\,\{X \leq x\} \tag{2.90}$$

and the transformed variable $T(U)$, with U drawn uniformly from $[0,1)$, has CDF $P(x)$ as well:

$$\text{Prob}\,\{T(U) \leq x\} = \text{Prob}\left\{P^{-1}(U) \leq x\right\} = \text{Prob}\,\{U \leq P(x)\} = P(x). \tag{2.91}$$

This process is visualized in Figure 2.22. Note that the method also works in higher dimensions. For a 2D distribution $p(x,y)$ for instance, we can first sample from the marginalized density $p(x)$, followed by sampling the conditional density $p(y \,|\, x)$.

When using the inversion method for importance sampling during Monte Carlo integration, the above steps can also be understood as a change of variables of the target integral, where we switch from the integration domain $\mathcal{X}$ to the unit-hypercube $\mathcal{U} = [0,1)^d$ of matching dimension d:

$$I = \int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}) = \int_{\mathcal{U}} f(T(\mathbf{u}))\,|J_T(\mathbf{u})|\,\mathrm{d}\mu(\mathbf{u}) = \int_{\mathcal{U}} \frac{f(T(\mathbf{u}))}{p(T(\mathbf{u}))}\,\mathrm{d}\mu(\mathbf{u}). \tag{2.92}$$

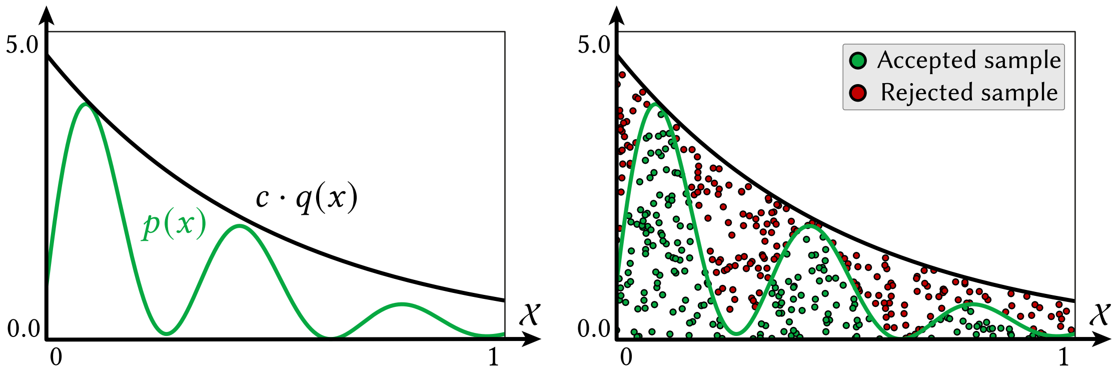


Figure 2.23: Illustration of the rejection sampling process on a 1D example. **Left**: We want to generate samples from a PDF $p(x)$ by sampling an auxiliary density $q(x)$ where $p(x)$ is bounded from above by $c \cdot q(x)$ for some constant c. **Right**: Points (x_i, y_i) where x_i are sampled according to $q(x)$ and y_i are uniformly sampled between 0 and $c \cdot q(x_i)$. We only accept samples that fall below the $p(x)$ curve (green), whose x-coordinate are then distributed according to $p(x)$.

Here, the Jacobian determinant $|J_T(\mathbf{u})|$ is necessary to account for the expansion (or contraction) of space and is exactly $1/p(\mathbf{x})$. This should be no surprise, as this results again in the the standard Monte Carlo estimator from Equation (2.79) with samples $\mathbf{u}$ drawn uniformly at random.

Inverse transform sampling is used extensively in rendering due to its simplicity and efficiency. The mapping T uses a fixed number of random numbers, is usually efficient to evaluate, and preserves potential sample stratification. However, it is not able to sample from distributions where the CDF is not available in analytic form (step 1 above). Similarly, the CDF inversion (step 2) can be problematic in practice, though it can be performed numerically (e.g. with root-finding methods) or circumvented entirely using other clever observations [61].

Rejection sampling. This method was introduced by Neumann [62] and can in principle sample from arbitrary target densities $p(\mathbf{x})$, even if e.g. the inversion method is not applicable. Interestingly, there is also no requirement that p is normalized so it is sufficient if we can evaluate it up to some unknown scale factor.

As suggested by its name, there is however the downside that a number of auxiliary samples might need to be *rejected* before a single sample distributed according to p can be returned from the process.

A sample $\mathbf{x}$, distributed according to $p(\mathbf{x})$ is generated in the following way:

1. Generate a proposal $\mathbf{x}'$ drawn from an auxiliary distribution $q(\mathbf{x})$ that we can easily sample from. We also need a constant c such that $c \cdot q(\mathbf{x})$ is an upper bound of $p(\mathbf{x})$.
2. Generate a uniform 1D variable u drawn from $[0, 1)$.
3. Accept the proposal $\mathbf{x}'$ as a sample of p (i.e. return $\mathbf{x} = \mathbf{x}'$) if $u \leq \frac{p(\mathbf{x}')}{c \cdot q(\mathbf{x}')}$. Otherwise, reject the proposal and restart at step 1.

Figure 2.23 gives the intuition of why this method works on a simple 1D example. The proposed samples essentially lie in the 2D space with their x-value distributed according the the auxiliary density, and their y-value drawn uniformly between 0 and $c \cdot q(x)$. The accepted samples then all lie below the target density with their x-values distributed proportionally to $p(x)$.

Note that the number of attempts before a successful sample is returned is unbounded, but equal to the constant c in expectation. This means, the efficiency of rejection sampling directly depends on how well chosen the upper bound is. Unfortunately it can be hard in practice to find an auxiliary density that is both a tight upper bound and that can be sampled easily. Compared to the inverse transform sampling, rejection sampling is incompatible with structured and low-discrepancy point sets.

Markov chain Monte Carlo. Another general technique that can sample from arbitrary target densities is the *Metropolis-Rosenbluth-Hastings algorithm* [63, 64] in the larger class of *Markov chain Monte Carlo (MCMC)* methods [56, 65].

A *Markov chain* is a sequence of random variables $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \ldots$ in some state space (e.g. $\mathbb{R}$) where the value of the next variable is determined via a (randomized) *mutation* or *perturbation strategy*, i.e. by sampling from a transition density $T(\mathbf{x} \to \mathbf{x}')$. Markov chains are “memory-less” because a mutation is only allowed to depend on the current state $\mathbf{x}$ of the sequence and the history of all previous states is “forgotten” after each mutation.

If these transitions fulfill relatively mild conditions[15], the Markov chain will converge to a unique *stationary distribution* after some finite warm-up period. After that, subsequent states $\mathbf{x}_i$ are valid samples from that distribution.

[15]The Markov chain needs to be *ergodic*, meaning there needs to be a non-zero probability that each possible $\mathbf{x}'$ can be reached from any other $\mathbf{x}$ (after potentially many steps), and there cannot be any set of states that are visited regularly. See Meyn and Tweedie [65] for details.

The key observation that leads to the Metropolis-Rosenbluth-Hastings algorithm is that a Markov chain can be constructed in a way that converges to a specific stationary distribution

$$p(\mathbf{x}) = \frac{g(\mathbf{x})}{\int_{\mathcal{X}} g(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})}, \tag{2.93}$$

where the normalization constant can be unknown. This is achieved by introducing an additional step at each mutation: similarly to rejection sampling, each new state is only accepted with a specific probability

$$a(\mathbf{x}\to\mathbf{x}') = \min\left\{1, \frac{g(\mathbf{x}')\,T(\mathbf{x}'\to\mathbf{x})}{g(\mathbf{x})\,T(\mathbf{x}\to\mathbf{x}')}\right\}. \tag{2.94}$$

Concretely, the process to determine the next sample $\mathbf{x}_{i+1}$ from $\mathbf{x}_i$ is:

1. Generate a proposal $\mathbf{x}'$ drawn from the transition density $T(\mathbf{x}_i\to\mathbf{x}')$.
2. Accept the sample, i.e. set $\mathbf{x}_{i+1} = \mathbf{x}'$, with probability $a(\mathbf{x}_i\to\mathbf{x}')$. Otherwise, reject the sample, i.e. repeat the previous sample and set $\mathbf{x}_{i+1} = \mathbf{x}_i$.

This simple method can be effective in cases where sampling from complex distributions would otherwise be prohibitively expensive. However, even though this will produce samples that are distributed proportionally to $p(x)$, samples can be highly correlated with each other, which can harm Monte Carlo convergence in practice.

2.4.4 Multiple importance sampling

It can be very challenging to design a sampling technique that is well-suited for a complex integrand $f(\mathbf{x})$. However, it is often possible to find a set of simpler sampling densities $p_j(\mathbf{x})$ ($j = 1, \ldots, M$) that only target parts of it like in the 1D example shown in Figure 2.24 (left).

While each individual estimator ${}^{f(\mathbf{x})}/_{p_j(\mathbf{x})}$ can cause high variance, a natural question is if we can somehow turn these into a unified sampling technique that combines their individual strengths. At first this is not obvious. For instance, a naïve average of two estimators

$$\langle I\rangle^{(\text{avg.})} = \frac{1}{2}\left(\frac{f(\mathbf{x}_1)}{p_1(\mathbf{x}_1)} + \frac{f(\mathbf{x}_2)}{p_2(\mathbf{x}_2)}\right) \tag{2.95}$$

is often ineffective because variance is additive:

$$\operatorname{Var}\left[\langle I\rangle^{(\text{avg.})}\right] = \frac{1}{4}\left(\operatorname{Var}\left[\frac{f(\mathbf{x}_1)}{p_1(\mathbf{x}_1)}\right] + \operatorname{Var}\left[\frac{f(\mathbf{x}_2)}{p_2(\mathbf{x}_2)}\right]\right). \tag{2.96}$$

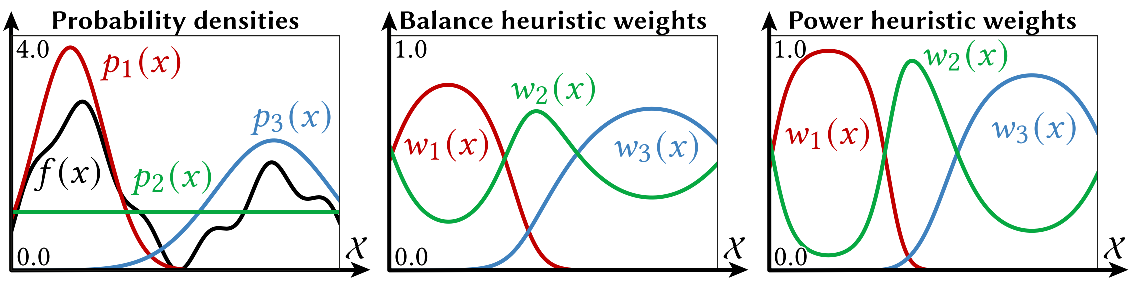


Figure 2.24: Illustration of multiple importance sampling in 1D. **Left**: Three sampling densities p_1, p_2, p_3 that are designed to target different areas of the integrand f. **Middle**: The corresponding multiple importance sampling weights w_1, w_2, w_3 computed from the balance heuristic. **Right**: The same situation again but using the power heuristic with $\beta = 2$ instead, which gives higher weights to individual strategies.

The right approach is to use a weighted average with weights $w_j(\mathbf{x})$ that can locally give higher priority to sampling techniques that work well while downweighting others that perform poorly. This concept is called *multiple importance sampling (MIS)* and was introduced by Veach and Guibas [66]. The multi-sample MIS estimator is:

$$\langle I \rangle^{(\mathrm{MIS})} = \sum_{j=1}^{M} \frac{1}{N_j} \sum_{i=1}^{N_j} w_j(\mathbf{x}_{j,i}) \frac{f(\mathbf{x}_{j,i})}{p_j(\mathbf{x}_{j,i})}, \tag{2.97}$$

where $\mathbf{x}_{j,i}$ denotes the i-th sample generated from strategy j, and a different number of samples N_j can be allocated to each technique.

In order for Equation (2.97) to be an unbiased estimator of I, the MIS weights w_j are required to satisfy the following two conditions:

1. They need to form a partition of unity on the non-zero regions of the integrand, I.e. $\sum_{j=1}^{M} w_j(\mathbf{x}) = 1$ for $f(\mathbf{x}) \neq 0$.

2. If a strategy j does not cover a subregion of the integration domain ($p_j(\mathbf{x}) = 0$ for some $\mathbf{x}$), the associated weight $w_j(\mathbf{x})$ needs to be zero. I.e. $w_j(\mathbf{x}) = 0$ whenever the ratio $f(\mathbf{x})/p_j(\mathbf{x})$ is not finite.

Interestingly, it is valid for MIS weights to be negative, and it was recently shown that this is in fact necessary to find optimal weights that minimize variance [67].

One way to arrive at Equation (2.97) is to partition the integral I based on the weights

$$I = \int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}) = \int_{\mathcal{X}} \underbrace{\sum_{j=1}^{M} w_j(\mathbf{x})}_{=1}\, f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}) = \sum_{j=1}^{M} \underbrace{\int_{\mathcal{X}} w_j(\mathbf{x})\, f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})}_{I_j}, \tag{2.98}$$

and then to estimate each integral I_j with the standard Monte Carlo estimator from Equation (2.81).

Alternatively, we can also define the one-sample MIS estimator

$$\langle I \rangle^{(\text{MIS-1})} = \frac{w_k(\mathbf{x}) f(\mathbf{x})}{\mathrm{P}_k \cdot p_k(\mathbf{x})}, \tag{2.99}$$

where only a single sample from strategy k is drawn after selecting one of the integrals with a discrete probability P_k.

The most popular instance of MIS weights in practice are defined via the *balance heuristic*

$$w_j(\mathbf{x}) = \frac{N_j \, p_j(\mathbf{x})}{\sum_{k=1}^{M} N_k \, p_k(\mathbf{x})}, \tag{2.100}$$

that is based on PDF ratios of the involved sampling techniques. We show in Figure 2.24 (middle) how this applies for the previous 1D example. A nice intuition behind the balance heuristic is to interpret it as sampling from a (weighted) average of the individual PDFs. This can be seen by plugging Equation (2.100) into the multi-sample MIS estimator and cancelling out some terms:

$$\begin{aligned}
\langle I \rangle^{(\text{Bal.})} &= \sum_{j=1}^{M} \frac{1}{N_j} \sum_{i=1}^{N_j} w_j(\mathbf{x}_{j,i}) \, \frac{f(\mathbf{x}_{j,i})}{p_j(\mathbf{x}_{j,i})} \\
&= \sum_{j=1}^{M} \frac{1}{N_j} \sum_{i=1}^{N_j} \frac{N_j \, p_j(\mathbf{x}_{j,i}))}{\sum_{k=1}^{M} N_k \, p_k(\mathbf{x}_{j,i})} \frac{f(\mathbf{x}_{j,i})}{p_j(\mathbf{x}_{j,i})} \\
&= \sum_{j=1}^{M} \sum_{i=1}^{N_j} \frac{f(\mathbf{x}_{j,i})}{\sum_{k=1}^{M} N_k \, p_k(\mathbf{x}_{j,i})} \\
&= \frac{1}{N} \sum_{j=1}^{M} \sum_{i=1}^{N_j} \frac{f(\mathbf{x}_{j,i})}{\sum_{k=1}^{M} c_k \, p_k(\mathbf{x}_{j,i})} = \frac{1}{N} \sum_{j=1}^{M} \sum_{i=1}^{N_j} \frac{f(\mathbf{x}_{j,i})}{\hat{p}(\mathbf{x}_{j,i})},
\end{aligned} \tag{2.101}$$

where in the last two steps, N is the number of total samples $N = \sum_{j=1}^{M} N_j$, the ratio of samples allocated to each technique is $c_k = {}^{N_k}/_{N}$, and $\hat{p}(\mathbf{x}) = \sum_{k=1}^{M} c_k \, p_k(\mathbf{x})$. In the simple example of $M = 2, N_1 = N_2 = 1$ this is just a standard Monte Carlo estimator where samples are drawn based on the average of the two PDFs:

$$\langle I \rangle^{(\text{Bal.})} = \frac{1}{2} \sum_{j=1}^{2} \frac{f(\mathbf{x}_{j,i})}{\frac{1}{2}\left(p_1(\mathbf{x}_{j,i}) + p_2(\mathbf{x}_{j,i})\right)}. \tag{2.102}$$

The balance heuristic is provably optimal in the case of the one-sample estimator (2.99) though in general, better weights can be found [67]. A particularly simple extensions that often produces lower variance is the *power heuristic*

$$w_j(\mathbf{x}) = \frac{(N_j\, p_j(\mathbf{x}))^\beta}{\sum_{k=1}^{M}(N_k\, p_k(\mathbf{x}))^\beta}, \tag{2.103}$$

where usually $\beta = 2$ gives good results. We also illustrate these alternate weights in Figure 2.24 (right).

As multiple importance sampling can lead to very robust estimators it has become a key component behind most modern rendering techniques. It is also still an ongoing direction of research. For instance, MIS can be extended to work on an space with a continuous number of strategies [68], and improved weight heuristics can be designed based on sample variance [69] or correlation [70].

2.4.5 Antithetic sampling

Another variance reduction strategy is *antithetic sampling* [71] which works by exploiting correlation between generated samples. Written as an estimator it is

$$\langle I\rangle^{(\mathrm{AS})} = \frac{1}{N}\sum_{i=1}^{N/2}\left(\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} + \frac{f(\tilde{\mathbf{x}}_i)}{p(\tilde{\mathbf{x}}_i)}\right), \tag{2.104}$$

where the number of samples N has to be even and $\tilde{\mathbf{x}}_i$ is the *antithetic sample* to $\mathbf{x}_i$ that is usually placed symmetrically to $\mathbf{x}_i$ in some way that creates cancellation between the two averaged terms.

The variance of the estimator is:

$$\begin{aligned}
\mathrm{Var}\left[\langle I\rangle^{(\mathrm{AS})}\right] &= \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N/2}\left(\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} + \frac{f(\tilde{\mathbf{x}}_i)}{p(\tilde{\mathbf{x}}_i)}\right)\right] \\
&= \frac{1}{N^2}\sum_{i=1}^{N/2}\mathrm{Var}\left[\frac{f(\mathbf{x}_i)}{p(\mathbf{x}_i)} + \frac{f(\tilde{\mathbf{x}}_i)}{p(\tilde{\mathbf{x}}_i)}\right] \\
&= \frac{N/2}{N}\mathrm{Var}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})} + \frac{f(\tilde{\mathbf{x}})}{p(\tilde{\mathbf{x}})}\right] \\
&= \frac{1}{2N}\left(\mathrm{Var}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right] + \mathrm{Var}\left[\frac{f(\tilde{\mathbf{x}})}{p(\tilde{\mathbf{x}})}\right] + 2\mathrm{Cov}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}, \frac{f(\tilde{\mathbf{x}})}{p(\tilde{\mathbf{x}})}\right]\right).
\end{aligned} \tag{2.105}$$

Compared to a standard Monte Carlo estimator, this can have significantly lower variance in case of some anti-correlation between $f(\mathbf{x})/p(\mathbf{x})$ and $f(\tilde{\mathbf{x}})/p(\tilde{\mathbf{x}})$.

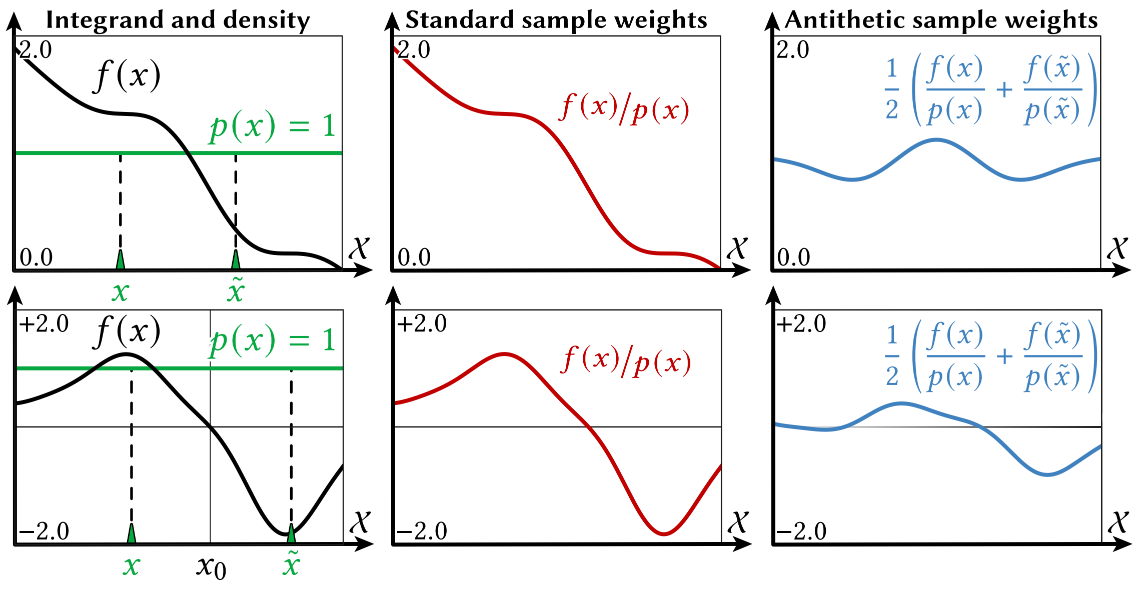


Figure 2.25: Illustration of antithetic sampling in 1D. **Left**: Integrand f and uniform sampling density p. **Middle**: The standard Monte Carlo sample weights $f/p = f$. **Right**: Sample weights produced by antithetic sampling. Note how they are much more constant due to cancellation, which will result in a lower-variance estimate. **Top**: Monotonic function with antithetic samples $\tilde{x} = 1 - x$. **Bottom**: Signed function where $\tilde{x}$ are equal to x flipped around the root x_0 of the function.

A well known case where antithetic sampling is effective are (approximately) monotonic functions. We show this in Figure 2.25 (top) on a 1D function $f(x)$ in $\mathcal{X} = [0, 1)$. There, the function value decreases as x increases, so we use antithetic samples $\tilde{x} = 1 - x$ to achieve a variance reduction.

Another case where antithetics can be beneficial are signed integrals as shown in Figure 2.25 (bottom). There is considerable potential for cancellation if we can sample such that $f(\mathbf{x}) \approx -f(\tilde{\mathbf{x}})$. In the shown 1D example, we therefore choose the antithetic sample $\tilde{x}$ as a mirrored version of x around the root of the function.

Even though common integrands in forward rendering are strictly positive and only rarely monotonic, antithetic sampling has been shown to improve variance in some cases [72]. More recently, is proved to be a valuable technique for reducing variance in differentiable rendering scenarios where derivative integrals always exhibit positive and negative regions [2, 73, 74] and we will revisit this later in Chapter 4.

2.4.6 Control variates

A third common approach for variance reduction are *control variates* [55]. Here, we rewrite the integral I based on an auxiliary function $g(\mathbf{x})$ with a known analytic solution to its integral $G = \int_{\mathcal{X}} g(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})$:

$$I = \int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}) \int_{\mathcal{X}} f(\mathbf{x}) - \beta\, g(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x}) + \beta\, G, \tag{2.106}$$

where the constant β controls how strongly the control variate is applied. The corresponding estimator is therefore

$$\langle I\rangle^{(\mathrm{CV})} = \frac{1}{N}\sum_{i=1}^{N} \frac{f(\mathbf{x}_i) - \beta\, g(\mathbf{x}_i)}{p(\mathbf{x}_i)} + \beta\, G, \tag{2.107}$$

with variance

$$\begin{aligned}
\mathrm{Var}\left[\langle I\rangle^{(\mathrm{CV})}\right] &= \mathrm{Var}\left[\frac{1}{N}\sum_{i=1}^{N} \frac{f(\mathbf{x}_i) - \beta\, g(\mathbf{x}_i)}{p(\mathbf{x}_i)} + \beta\, G\right] \\
&= \frac{1}{N^2}\sum_{i=1}^{N} \mathrm{Var}\left[\frac{f(\mathbf{x}_i) - \beta\, g(\mathbf{x}_i)}{p(\mathbf{x}_i)}\right] \\
&= \frac{1}{N}\mathrm{Var}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})} - \beta\,\frac{g(\mathbf{x})}{p(\mathbf{x})}\right] \\
&= \frac{1}{N}\left(\mathrm{Var}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right] + \beta^2\,\mathrm{Var}\left[\frac{g(\mathbf{x})}{p(\mathbf{x})}\right] - 2\beta\,\mathrm{Cov}\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}, \frac{g(\mathbf{x})}{p(\mathbf{x})}\right]\right).
\end{aligned} \tag{2.108}$$

Depending on the value of β, the function g should be either strongly correlated (for $\beta > 0$) or anti-correlated (for $\beta < 0$) with f to achieve an overall variance reduction.

It is also possible for the integral G to be an estimator itself, although this lowers the effectiveness due to the additional randomness. This is known as *multilevel Monte Carlo* [75].

2.5 Rendering algorithms

We now discuss algorithms that solve the light transport problem by targeting the equations discussed in Section 2.3 using the Monte Carlo integration techniques from Section 2.4. We will restrict ourselves to a high-level discussion and mainly cover their strengths and weaknesses. For more complete descriptions and implementation details, refer to Pharr et al. [8].

2.5.1 Direct illumination

Recall the measurement equation (2.47) from earlier:

$$I_k = \int_{\mathcal{M}} \int_{\mathcal{S}^2} W_\mathrm{e}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, L_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}). \tag{2.109}$$

Direct application of the Monte Carlo estimator (2.79) gives

$$\langle I_k \rangle = \frac{W_\mathrm{e}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, L_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega})}{p(\mathbf{x}, \boldsymbol{\omega})}, \tag{2.110}$$

which still requires a suitable sampling density $p(\mathbf{x}, \boldsymbol{\omega})$. Given that L_o represents an unknown and complicated function, it makes sense to only focus on the importance term $W_\mathrm{e}^{(k)}$. For basic camera models, this is a simple analytic expression with only a small non-zero region around pixel k. In case of a pinhole camera, we can for instance sample a position $\mathbf{x}$ uniformly inside the pixel k and trace a ray from the pinhole through $\mathbf{x}$ into the scene. This process is actually an ideal sampling scheme, and its PDF perfectly cancels out the $W_\mathrm{e}^{(k)}$ term in the numerator. This only leaves L_o, which is itself an integral:

$$L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{e}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}), -\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega}_\mathrm{i}). \tag{2.111}$$

For simplicity, we replaced the L_o term inside with just the emission term L_e at the next intersection, instead of the full recursive definition. This amounts to only considering light paths with a maximum length of two, i.e. *direct illumination*. Unsurprisingly, we then also estimate this integral with Monte Carlo:

$$\langle L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) \rangle = L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) + \frac{f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \rightarrow \boldsymbol{\omega}_\mathrm{o}) L_\mathrm{e}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}), -\boldsymbol{\omega}_\mathrm{i})}{p(\boldsymbol{\omega}_\mathrm{i})}. \tag{2.112}$$

Here, the choice of sampling density is less obvious. Ideally, we would like to sample proportionally to the full product $f_s \cdot L_\mathrm{e}$, but this is generally infeasible without making strong assumptions on the incident illumination and the shading model [76, 77, 78].

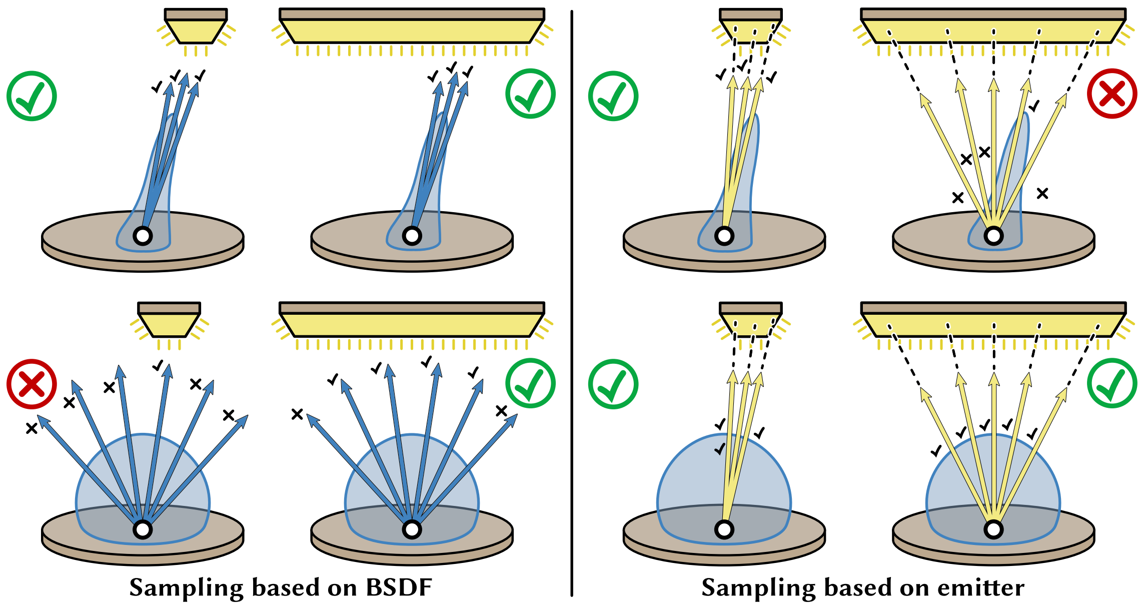


Figure 2.26: We can use two different strategies to importance sample the product integral of BSDF (blue) and incident illumination (yellow). For both options, the drawings show the possible combinations of small or large emitters, and specular or rough BSDFs. **Left**: BSDF sampling is effective in many cases but struggles in cases of small light sources where directions sampled from rough BSDFs will often miss the lights. **Right**: Emitter sampling has the benefit of easily connecting to small light sources, assuming there is no occlusion. Directions generated due to large light sources on the other hand might miss the important parts of the BSDF when the material is specular.

A simple workaround that is robust in most cases is to rely on two sampling techniques, which are tailored to the f_s and L_e term individually, and then combined using MIS from Section 2.4.4.

We briefly discuss both strategies separately before looking at the combined MIS estimator. See also the corresponding illustration in Figure 2.26.

BSDF sampling. This strategy attempts to sample (roughly) proportional to the cosine weighted[16] BSDF $f_s(\mathbf{x}, \boldsymbol{\omega}_i \to \boldsymbol{\omega}_o) \cdot \cos\theta_i$ using a density $p^{(\mathrm{BSDF})}(\boldsymbol{\omega}_i)$ that is implicitly conditioned on the position $\mathbf{x}$ and direction $\boldsymbol{\omega}_o$. Thanks to the significant attention given to BSDF sampling strategies in the last decades we now have access to very robust sampling techniques for common cases such as microfacet BSDFs [79].

BSDF sampling is useful when the model is only non-zero on a small part of the unit sphere. We essentially concentrate our sampling on those regions instead of tracing rays in directions where the full product $f_s \cdot L_e$ is likely to be small.

[16]We usually factor the cosine of the projected solid angle measure $\mathrm{d}\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega})$ into the BSDF term for the purpose of importance sampling.

On the other hand, if the BSDF is less concentrated, the sampled directions will be spread out accordingly. This can cause high variance in presence of small light sources as these will only be hit with a low probability.

Most rendering systems also support light sources that have an emission term which include either a spatial or directional Dirac delta function. Common examples are point, spotlight, or directional emitters. Rays generated from BSDF sampling cannot hit these types of emitters at all because they represent only a zero measure direction on the unit sphere.

Emitter sampling. Alternatively, we can sample based on the incident illumination L_e, a process also known as *next event estimation.* In its basic form, this entails sampling an explicit endpoint $\mathbf{x}'$ on the emitter (often with a uniform density on the emissive surface) and explicitly ray tracing towards this point. For common light sources (such as points, quads, spheres, ellipsoids, or environment maps), more sophisticated techniques exist that more accurately account for the projected light source geometry and emission profile.

While sample generation can fail in case of occluding geometry, this is a great strategy for small light sources that would be difficult to find using other sampling strategies. The downside is that the BSDF term is now not taken into account at all which can also cause high variance in case of specular materials. As a limit case, perfectly specular BSDFs involving a Dirac delta function (Sections 2.2.3 and 2.2.4) are incompatible with emitter sampling.

It should also be noted that virtual scenes might include a large number of light sources. In fact, it is not uncommon for today's production scenes to be lit by millions of unique emitters. Choosing one of these to sample from is therefore also a challenge in itself as often only a small subset contributes significantly to the integrand at a position. This was classically approached using simple heuristics [80, 81] whereas today, either complex light hierarchies [82, 83] or importance resampling [84] are used.

Multiple importance sampling. Given how well the advantages and shortcomings of these two techniques complement each other, their combination is a natural next step. This observation led to the development of multiple importance sampling by Veach and Guibas [66].

We revisit a well-known test scene from that publication here in Figure 2.27 to illustrate its effectiveness. It features a set of surfaces with increasing roughness from top to bottom, as well as various emitter sizes. Rendering this test image with either of the two

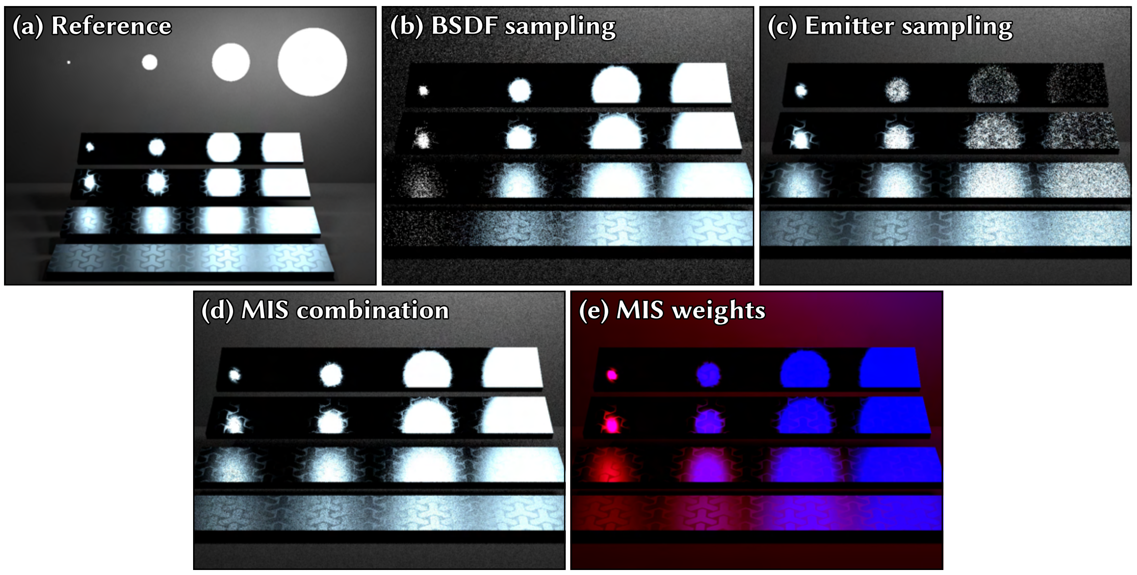


Figure 2.27: **(a)** Recreation of a well-known test scene by Veach and Guibas [66] involving surfaces of varying roughness levels lit by light sources of different sizes. **(b)** BSDF sampling works well except for cases with high roughness and small emitters (bottom left). **(c)** Emitter sampling is well-suited for that region but struggles in the case of low roughness and large emitters (top right). **(d)** Combining both strategies with MIS results in a robust estimate in all parts of the image. **(e)** A false color visualization of the same scene that shows the MIS weights for the two strategies (blue: BSDF sampling, red: emitter sampling). **(c–d)** use 64 samples per pixel whereas **(d)** uses 32 samples for each of the two strategies in order to make the comparison as fair as possible. All three renderings take roughly equal time.

proposed techniques has serious variance issues in two different regions of the image that align with the discussions above, see Figure 2.27 (b–c). Only their combination via MIS in Figure 2.27 (d) produces a robust estimate with overall low variance.

2.5.2 Path tracing

We now lift the previous limitation of only accounting for direct illumination and turn to the full light transport problem, i.e. the rendering equation (2.51). This now includes global illumination by simulating light paths of *any* length. See Figure 2.28 for an illustration of how increasing path length affects the rendering of a simple test scene.

Kajiya's seminal *path tracing* algorithm [52] is the most common solution approach for this problem today. It starts out with the same measurement equation estimate (2.110) from earlier but then recursively also estimates the series of nested integrals. Concretely, a full light path $\bar{\mathbf{x}} = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$ is generated by subsequently sampling a new direction (usually from the BSDF sampling density) followed by ray tracing to arrive at the next path vertex. During the process, the method also tracks the current path through-

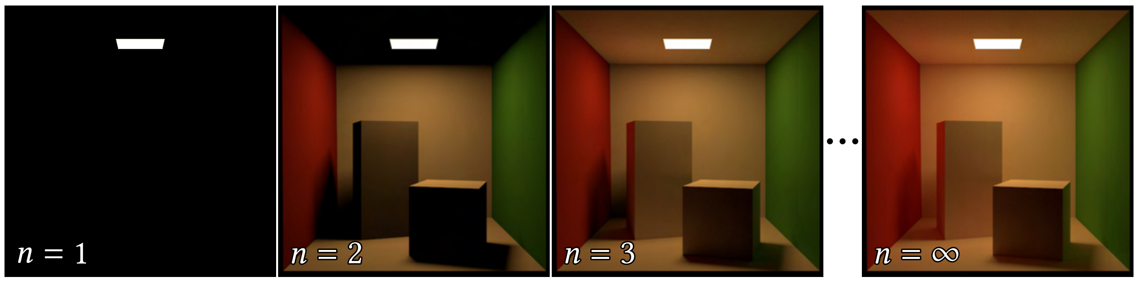


Figure 2.28: Rendering of the Cornell box scene with increasing number of bounces. A path with length $n = 1$ will only be able to connect directly visible lights to the camera. For rendering direct illumination (as in Section 2.5.1), $n = 2$ is sufficient. A higher number of bounces ($n \geq 3$) leads to the progressive addition global illumination effects due to the interreflection of light.

put (2.60) and the final radiance estimate is updated whenever the path encounteres an emissive surface.

In practice, it is also a good idea to still rely on the MIS technique between BSDF and emitter sampling at each path vertex. Because there is no simple way to sample the *indirect* radiance arriving at a point, MIS usually still focuses solely on the directly incident illumination whereas the indirect contribution is accounted for by the recursive part of the estimator.

But how do we actually terminate this recursion? While the sampled light paths might naturally stop eventually (e.g. due to absorption or after "missing" the scene geometry completely and escaping into infinity), there are many cases where there is potential for an infinite recursion. Manually setting a maximum number of computed bounces will lead to energy loss and thus the estimator would be biased. The solution is a simple modification called *Russian Roulette* which terminates long paths stochastically instead.

Consider one of the radiance estimates $\langle L \rangle$ that includes a recursion. We can replace this estimator with a modified version $\langle L \rangle^{(\mathrm{RR})}$ that only continues the recursion with a continuation probability P_R and scales the estimate accordingly:

$$\langle L \rangle^{(\mathrm{RR})} = \begin{cases} \langle L \rangle / P_R, & \text{if } u < P_R \\ 0, & \text{otherwise,} \end{cases} \tag{2.113}$$

where u is a uniform variate in $[0, 1)$. This still has the same expected value

$$\mathrm{E}\left[\langle L \rangle^{(\mathrm{RR})}\right] = P_R \cdot \langle L \rangle / P_R + (1 - P_R) \cdot 0 = \langle L \rangle \tag{2.114}$$

and is thus unbiased. It is convenient to set P_R adaptively based on the current path throughput. This way, low energy paths are terminated with a higher probability which

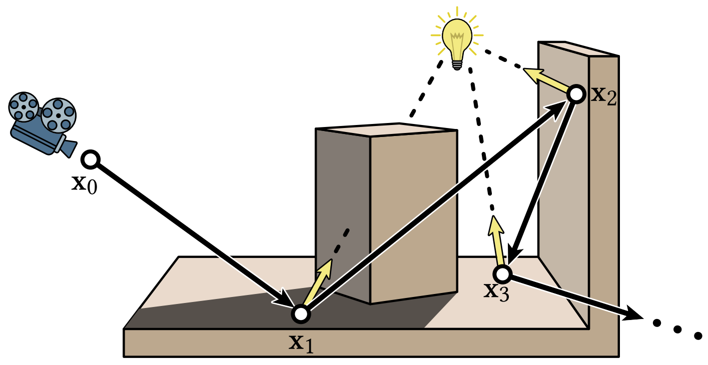


Figure 2.29: An illustration of the path tracing algorithm with multiple importance sampling. We start tracing rays from the camera into the scene. At each shading point $\mathbf{x}_i$ we perform MIS between BSDF sampling (black arrow) and emitter sampling (yellow arrow) that attempts to connect to a light source. The path is then continued recursively by following the sampled direction from BSDF sampling (black arrow).

saves computation that would not contribute much to the final image.

The final sequence of steps is illustrated in Figure 2.29. Note how the algorithm has linear complexity in the number of bounces. In particular, the rays used for emitter sampling (yellow arrows in the illustration) are only used to directly connect to lights and there is no branching which would lead to exponential path growth.

Path tracing is hugely popular and is the default approach in almost all rendering systems. This is mostly due to its favorable tradeoffs between simplicity, computational efficiency, and its robustness to many situations. However, one should recognize that there are many types of scenes that are challenging—if not impossible—to render using this method.

The default emitter sampling strategy tends to reach emitters that illuminate the scene indirectly (i.e. via interreflection) with too low a probability. They have to be discovered mostly by BSDF sampling which can cause high variance in case the emitters are spatially or directionally small.

Another issue is rendering of caustics, where light is either reflected or refracted by specular surfaces. This also “disables” emitter sampling and the only remaining option is BSDF sampling. Even worse, if the emitter causing the caustic has a Dirac delta term in its emission profile, no path tracing strategy can generate suitable samples, and the caustic will be missing entirely from the rendered image.

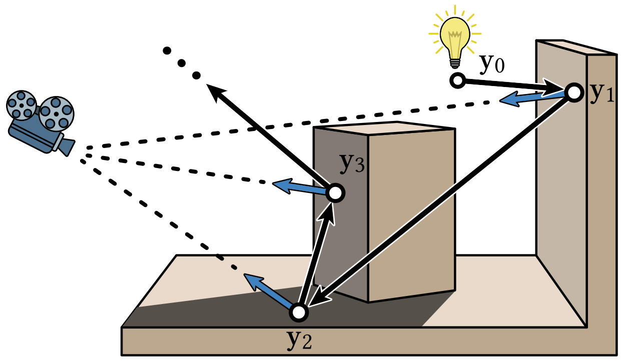


Figure 2.30: Light tracing works analogously to path tracing (Figure 2.29) but starts tracing rays from light sources and tries to create an explicit connection to the camera at each bounce.

2.5.3 Light tracing

It might seem like we can resolve many of the problems with path tracing simply by sampling paths starting from the light sources. The symmetry of light transport, as discussed earlier in Section 2.3.4, makes this a perfectly valid approach. As shown in Figure 2.30, the resulting algorithm, called *light tracing*, is almost equivalent to path tracing. Rays are traced starting from the other end now but are still scattered based on BSDF sampling at each bounce. Emitter sampling is replaced with "camera sampling" where an explicit connection to the camera's sensor plane is made and any potential contribution is directly added to the corresponding image pixel.

Light tracing is a fairly robust estimate for directly visible caustics or emitters that are hidden from the camera view. But similar to emitter sampling, camera sampling is very ineffective when connecting to the sensor from a highly specular BSDF, or even invalid when the BSDF contains a delta distribution. Unfortunately, hitting the camera randomly by scattering based on the BSDF is usually an even worse strategy because the camera aperture is in most cases tiny or infinitesimally small. Thus, there is no effective strategy for directly visible specular objects, which is a severe limitation.

2.5.4 Bidirectional techniques

One option to circumvent the previously discussed problems is to switch to the path space formulation of light transport (Section 2.3.3) with an estimator

$$\langle I \rangle = \frac{f(\bar{\mathbf{x}})}{p(\bar{\mathbf{x}})}, \tag{2.115}$$

where a light path $\bar{\mathbf{x}}$ was constructed based on some arbitrary probability measure $p(\bar{\mathbf{x}})$. This opens the door to many more path construction strategies. A particularly popu-

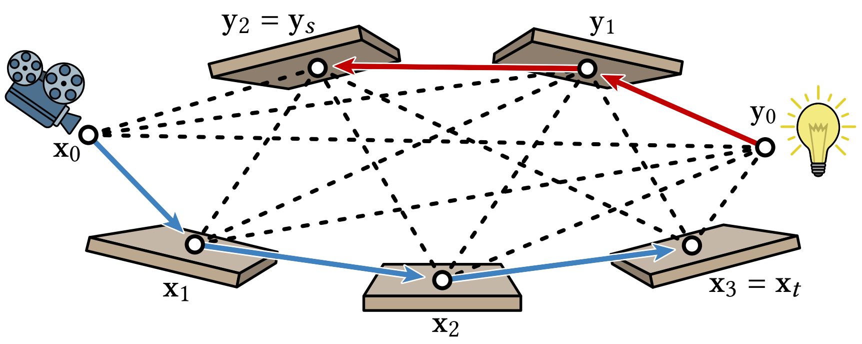


Figure 2.31: Bidirectional path tracing constructs light paths by tracing a length t subpath from the camera and a length s subpath from the light sources. In total, this results in $(t+1)\cdot(s+1)$ possible paths.

lar class of approaches are *bidirectional* techniques, where paths are constructed from both the camera and light source simultaneously. In principle, generalizations such as *tridirectional* approaches [85] are also applicable in more specific use cases.

Bidirectional path tracing. *Bidirectional path tracing (BDPT)* [86, 87] is a well known example that is regarded as being robust in many scenes. It generates a camera subpath $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t$ and a light subpath $\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_s$ of lengths t and s respectively. From this, we can construct $(t+1)\cdot(s+1)$ complete light paths by connecting one vertex from each subpath as shown in Figure 2.31.

Conversely, this means each complete light path of length n can be sampled with $n+1$ different strategies $\{(t=0, s=n), (t=1, s=n-1), \ldots, (t=n, s=0)\}$. Depending on the situation, only a subset of these will produce a good estimate, so they are usually all combined using MIS:

$$\langle I\rangle^{MIS} = \sum_{j=1}^{M} w_j(\bar{\mathbf{x}})\,\frac{f(\bar{\mathbf{x}})}{p_j(\bar{\mathbf{x}})}. \tag{2.116}$$

All of this suggests a large computational overhead compared to the simpler unidirectional algorithms. On simple scenes, BDPT is therefore often outperformed by path tracing in equal time comparisons. Additionally, there still exists a large number of scenes where BDPT struggles to sample high contributing light paths. In particular there are two main issues:

1. In scenes with complex geometry it is actually very common that many of the path connection strategies in Figure 2.31 will fail due to occlusions. Consider also the common case of an interior scene that is lit by the sun from the outside. Many of the light subpaths will not even make it through the room's windows and cannot

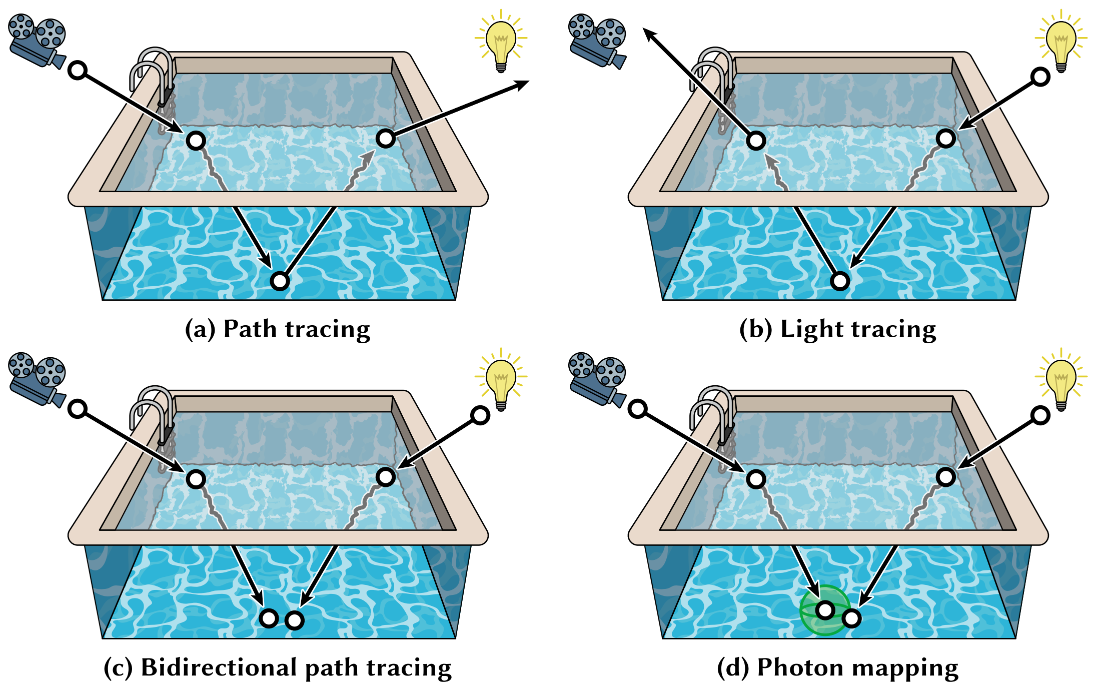


Figure 2.32: Specular-diffuse-specular (SDS) paths are particularly challenging for many types of rendering algorithms. This is illustrated here on the well known example of a swimming pool scene, but there exist many equivalent scenarios. **(a)** Path tracing will generally miss the light source after refracting out of the water. **(b)** Light tracing has exactly the same problem, but on the camera side. Note that in both cases, explicit connection to the emitter or camera is invalid due to the specular path vertex. **(c)** Bidirectional path tracing does not improve the situation either, as now both subpaths will not end up in the same location on the diffuse surface in the middle. **(d)** Biased techniques, such as photon mapping, can circumvent this, e.g. by performing density estimation of previously stored photons in the vicinity of the diffuse path vertex.

possibly connect to a camera subpath. As a consequence, most of the computation is wasted.

2. None of the many path sampling strategies can effectively deal with *specular-diffuse-specular (SDS)* light paths that have only a single diffuse vertex surrounded by specular vertices, see Figure 2.32 (a–c).

Photon mapping. A closely related algorithm is *photon mapping* [88, 89] which splits the overall rendering process into two steps illustrated in Figure 2.33:

1. Sample paths starting from the light sources. At each encountered scene surface, store information about the current path throughput in a *photon map* data structure.

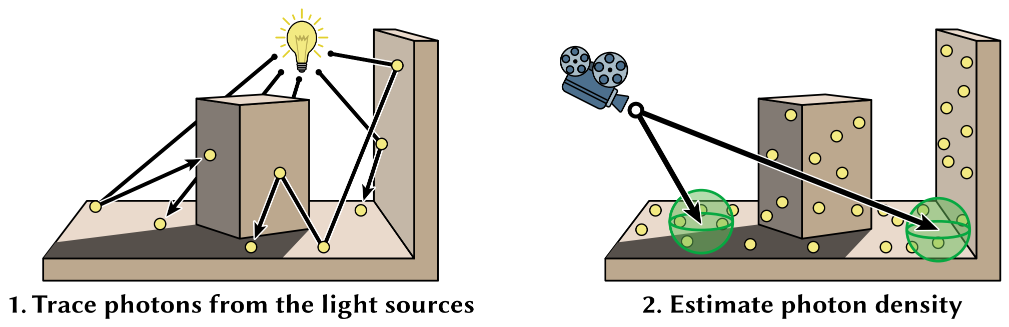

1. Trace photons from the light sources
2. Estimate photon density

Figure 2.33: Illustration of the two-phase photon mapping algorithm. First, paths are traced from the emitter side and a photon map is created by storing the incident illumination information at each path vertex (yellow dots). Second, rays are traced from the camera and at each shading point we estimate the incident illumination via density estimation inside a small kernel (green spheres).

2. Sample paths from the camera and at each hit point, query the stored photon information within a small kernel to estimate the radiance arriving at that location.

To keep memory storage size manageable, the two steps are usually repeated multiple times with a fixed number of emitted photons at each iteration.

The second step, which is essentially a *density estimation* process, allows us to connect path vertices with each other that do not perfectly meet in the same location. For example, we can connect the two middle vertices on the diffuse surface in the SDS example in Figure 2.32 (d).

However, as the kernel has a finite size, the photon mapping estimator is biased and in practice generates a slightly blurred result. Most implementations today produce consistent estimates by progressively reducing the kernel size [90, 91], so it will at least converge to the correct solution as the number of samples and iterations increases. It has also been shown that photon mapping can, in principle, be turned into an unbiased estimator [92]. But this trades off the bias against higher variance, and especially the challenging SDS paths are no longer well supported.

To make photon mapping practical, additional care is needed to not waste photon map computation and storage in scene regions that do not significantly contribute to the image, for instance by adaptively guiding light paths to important locations [93, 94].

Interestingly, even biased versions of photon mapping can be incorporated into the path space integral formulation which means it can be combined with BDPT via MIS [95, 96].

2.5.5 Metropolis light transport

A completely different rendering technique is *Metropolis light transport (MLT)*, built on top the Metropolis-Rosenbluth-Hastings algorithm (Section 2.4.3). This aims to sample light paths in the scene proportionally to the resulting image contribution by carefully designing a set of mutation strategies. These are either defined directly in path space [7] or in the underlying hyper-cube of random numbers (also known as *primary sample space* in this context) that is then mapped to light paths [97, 98]. Recently, methods that exploit both spaces simultaneously have also been considered [99, 100, 101].

Once a path with a high throughput is found (e.g. via a bidirectional sampling technique), MLT can locally explore the surrounding region of path space to find similar paths. Compared to other techniques that immediately throw away each sample once it has been used once, this therefore ammortizes the cost of finding challenging paths. This process requires a subtle balance between local and global exploration steps. While MLT is regarded as a powerful rendering technique that can deal exceptionally well with complex light transport, its adoption in production setting has been limited. The MCMC process is prone to generating highly correlated samples and the image often convergences non-uniformly, which can be problematic when rendering animations that should be temporally coherent. Recent progress in this direction investigates better stratification in image space [102] or selectively applying MLT only to the subset of light transport that is actually not well-sampled with simpler Monte Carlo techniques [103].

2.5.6 Path guiding

A recent line of works on rendering algorithms uses reinforcement learning embedded in the rendering process to build a representation of the 5D (spatial and directional) radiance field in the scene. This data structure gives information about direct *and indirect* radiance arriving at a location which can be used to improve local importance sampling decisions. This process, called *path guiding*, can boost the efficiency of unidirectional algorithms significantly.

Common radiance representations include Gaussian mixture models [104], spatial-directional trees [105], full light paths [106], or neural networks [107].

Path guiding is a practical option to reduce path tracing variance in challenging scenes [108]. It can however still struggle in case the radiance field has high-frequency contents. This happens, for example, for caustics due to complex specular geometry. In that case, the data structures require high fidelity to describe the radiance information.

2.6 Forward and reverse mode differentiation

Let us consider a function $\mathbf{y} = f(\mathbf{x})$ that map between spaces $\mathbb{R}^n$ and $\mathbb{R}^m$. Its *Jacobian matrix*

$$J_f(\mathbf{x}) = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_1}{\partial x_n}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1}(\mathbf{x}) & \dots & \frac{\partial f_m}{\partial x_n}(\mathbf{x}) \end{pmatrix} \tag{2.117}$$

collects all partial derivatives with respect to its inputs and outputs. Two notable special cases are scalar functions ($n = m = 1$), where the derivative $\partial y / \partial x = f'(x)$ is simply the *slope* of f at x, and the case of a scalar function with multivariate input ($n > 1, m = 1$) where the Jacobian simplifies to a row vector $\partial y / \partial \mathbf{x} = \nabla^T f(\mathbf{x})$, i.e. the transpose of the *gradient* $\nabla f(\mathbf{x})$ that points into the direction of steepest ascent in $\mathbb{R}^n$.

It is also well known from calculus that the Jacobian can be used to build a linear approximation of f around a point $\mathbf{x}'$ as

$$\mathbf{y} = f(\mathbf{x}) \approx f(\mathbf{x}') + J_f(\mathbf{x}') \cdot (\mathbf{x} - \mathbf{x}'). \tag{2.118}$$

We extensively use information from first-order function derivatives[17] in this thesis for both numerical root-finding (Chapter 3) or optimization tasks for solving inverse problems (Chapter 4). Before that, we turn to to the question of how to differentiate functions that are implemented as computer programs. There exist two main methods with their own respective use cases. Often, a full computation of the Jacobian matrix can be avoided.

We will illustrate both options on a simple example function $\mathbf{y} = f(\mathbf{x})$ with 3 inputs and 2 outputs

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1 x_2 x_3 \\ x_1 x_2 + x_3 \end{pmatrix} \tag{2.119}$$

with Jacobian matrix

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \frac{\partial y_1}{\partial x_3} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_2}{\partial x_3} \end{pmatrix} = \begin{pmatrix} x_2 x_3 & x_1 x_3 & x_1 x_2 \\ x_2 & x_1 & 1 \end{pmatrix}. \tag{2.120}$$

[17] Though not perfectly precise, we use the terms *derivatives* and *gradients* interchangeably in this thesis.

1. **Forward mode**: This computes the Jacobian-vector product

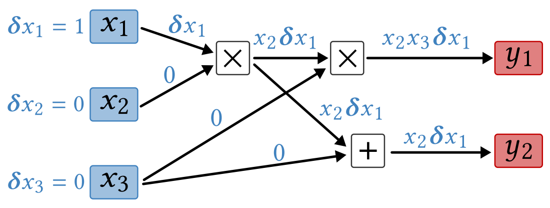
$$\boldsymbol{\delta}\mathbf{y} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \cdot \boldsymbol{\delta}\mathbf{x} \tag{2.121}$$

that tells us how a perturbation $\boldsymbol{\delta}\mathbf{x} = (\boldsymbol{\delta}x_1, \ldots, \boldsymbol{\delta}x_n)^T \in \mathbb{R}^n$ of the inputs is mapped to a perturbation $\boldsymbol{\delta}\mathbf{y} = (\boldsymbol{\delta}y_1, \ldots, \boldsymbol{\delta}y_m)^T \in \mathbb{R}^m$ of the function outputs. This is most commonly used to find the gradient of the outputs with respect to a *single* input coordinate x_i, which amounts to setting $\boldsymbol{\delta}\mathbf{x}$ to the standard basis vector

$$\mathbf{e}_i = (0, \ldots, 0, \underbrace{\;1\;}_{\text{entry } i}, 0, \ldots, 0)^T. \tag{2.122}$$

Note how the Jacobian-vector product then simply extracts the i-th column $^{\partial \mathbf{y}}/_{\partial x_i}$ of the full Jacobian matrix. In this case, a perturbation $\boldsymbol{\delta}v$ can also be interpreted as the numerical value of the derivative $^{\partial v}/_{\partial x_i}$ that is propagated through the forward computation.

Forward mode differentiation computes derivatives for all output variables and resembles how one would manually write down such a computation with pencil and paper. An example is shown here:



To compute the derivative for variable y_2, the derivative $\boldsymbol{\delta}x_1$ in variable x_1 is scaled by the value x_2 and subsequently added to the (zero-valued) derivative $\boldsymbol{\delta}x_3$. An implementation of this process is relatively straightforward: each variable v of a program needs to additionally track its x_i-derivative $\boldsymbol{\delta}v$, and all arithmetic operations acting on it need to also apply the corresponding derivative computation.

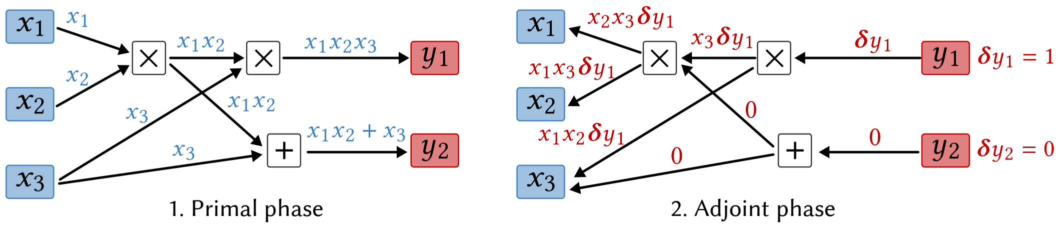
Forward mode differentiation becomes less useful when the function f has many inputs (i.e. its Jacobian has many columns). In that case, one such pass through the program is required *per input*.

2. **Reverse mode**: This alternate option computes the transposed Jacobian-vector product

$$\boldsymbol{\delta}\mathbf{x} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T \cdot \boldsymbol{\delta}\mathbf{y}, \tag{2.123}$$

which maps a perturbation $\boldsymbol{\delta}\mathbf{y} \in \mathbb{R}^m$ of the function outputs back to its inputs. Usually, we care about input derivatives with respect to a single output y_i, in which case we can again plug in a standard basis vector ($\boldsymbol{\delta}\mathbf{y} = \mathbf{e}_i$). This time, this extract the i-th row ${}^{\partial y_i}/_{\partial \mathbf{x}}$ of the Jacobian matrix and a perturbation $\boldsymbol{\delta}v$ should be interpreted as the numerical value of the derivative ${}^{\partial y_i}/_{\partial v}$ that it is propagated in reverse through the computation.

This is problematic however as we need access to all intermediate program variable states which are unkown when simply stepping backwards through the program. In general, this means that the program has to first run normally while the relevant intermediate results are stored in a *computation graph* or *tape* to be later reused in a reverse (a.k.a. *adjoint*) phase. Here we illustrate how this is done to compute the gradients of all inputs with respect to changes of a single output variable:



In the context of training neural networks, this exact procedure is usually referred to as *backpropagation*. Due to this split into two disjoint phases of computation, implementations of reverse mode differentiation are usually much more involved compared to forward mode.

Discussion. Which one of these methods will be more efficient in practice depends on the structure of the program that is being differentiated. On the one hand, if the program has many outputs but few inputs, the simpler forward mode is appropriate, and it will require as many passes through the program as there are input variables. If, on the other hand, there are more inputs than outputs, it is preferrable to make use of reverse mode differentiation as it requires one primal pass through the program, followed by one reverse pass per output.

For completeness, we mention that there also exists a *mixed mode*. Consider a very complex program that is differentiated using reverse mode—but it has a very costly subcomputation step that only has a single input. In that case, it is more efficient to use forward mode for that part locally, inside a scheme that is based on reverse mode overall.

The actual derivative expressions can either be derived by hand and manually implemented as *derivative programs*, or alternatively computed via *automatic differentiation (AD)*. There exist excellent tools nowadays that perform AD of numerical code [109, 110, 111, 112, 113, 114]. However, we will later show that direct application of such tools to complete rendering algorithms can produce unsatisfactory results or even completely wrong derivatives due to non-differentiable components related to the computation of discontinuous integrals. Additional care is required in our applications to ensure both correctness and adequate performance.

We refer to Griewank and Walther [115] for a much more detailed overview of AD and the different approaches for differentiation in general.

3 | Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints

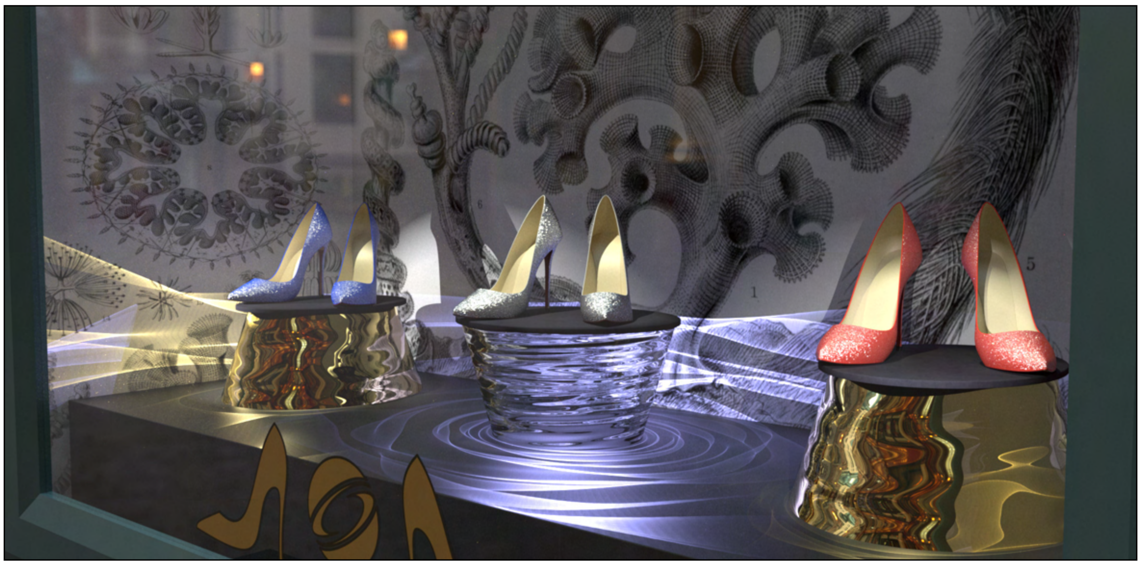


Figure 3.1: Rendering of a shop window featuring a combination of challenging-to-sample light transport paths with specular-diffuse-specular (SDS) interreflection: the two golden normal-mapped pedestals are illuminated by spot lights and project intricate caustic patterns following a single reflection from the metallic surface, while the transparent center pedestal is lit from the inside which generates caustics via double refraction. The glinty appearance of the shoes arises due to specular microgeometry encoded in a high-frequency normal map. This image was rendered by an ordinary unidirectional path tracer using our new specular manifold sampling strategy. The remaining noise is due to indirect lighting by caustics, which is not explicitly sampled by our technique.

In this chapter we focus on specular light paths, i.e. paths involving interactions with smooth metallic or refractive surfaces. Their constrained nature makes these difficult to find for many existing methods: at specular interfaces, light must satisfy the law of reflection or refraction, which drastically lowers the probability of sampling a valid configuration connecting the camera to a light source. Caustics or glittery surfaces that exhibit random patterns of highlights due to specular microgeometry are visually striking examples of such specular paths, though they are often much more subtle while still having a negative effect on overall convergence. The large family of specular-diffuse-specular (SDS) paths with only a single diffuse vertex surrounded by specular chains simply cannot be found at all and is normally absent in rendered images. The caustics seen through the shop window in Figure 3.1 are an example of this configuration—they are only visible thanks to the proposed sampling strategy. The glints on the shoes are even more challenging and involve *only specular vertices*: a point spotlight, perfectly specular microstructure, the glass window, and a pinhole camera.

In this chapter, we present a remarkably simple technique for sampling specular chains connecting two specified shading points—including glints and SDS paths—in an unbiased manner. Our approach builds on the theory of specular manifolds but significantly improves its practicality in formerly challenging cases, e.g. when working with high-frequency displaced or normal-mapped geometry. Concretely, our contributions are:

1. A unified manifold sampling strategy that is compatible with rendering both reflective and refractive caustics.
2. A specialized variant for rendering glints, which reduces memory usage hundredfold compared to prior work.
3. A biased variant of the method with reduced variance.
4. A two-pass sampling strategy for normal-mapped surfaces.
5. Changes to the specular manifold constraints of Jakob and Marschner [116] that improve robustness and convergence.
6. We show how our method can be deployed in a unidirectional path tracer and compare its performance to prior work.

The main limitation of our sampling technique is that variance increases significantly for longer specular chains, hence our experiments mainly focus on short chains with one or two vertices. That said, our approach is highly extensible, and we believe that future work could address this limitation.

Before delving into specifics about our method, Section 3.1 first reviews prior work in the areas of caustic and glint rendering techniques.

3.1 Background

While current unidirectional unidirectional [52] or bidirectional [86, 87] path sampling strategies are very robust in many common rendering scenarios, they can still be surprisingly brittle and subject to catastrophically poor convergence in the presence of SDS paths. This can even be the case if the path generation is backed with sophisticated guiding strategies that exploit past observations to improve the quality of generated samples [104, 105, 106, 107].

Methods based on photon maps (Section 2.5.4) are able to resolve the issues with SDS paths by introducing spatial blurring that relaxes the original problem. While they are an excellent choice for certain path classes, they can introduce objectionable blur, and they do not handle important cases including caustics on non-diffuse surfaces or glints.

Similarly, path space can also be selectively mollified to introduce bias only for path types that were otherwise challenging or impossible to sample [117, 118]. In contrast, our work focuses on general path sampling in the original non-relaxed problem.

Fermat's principle states that specular paths are *extremal*, i.e., they locally maximize or minimize the time that light requires to travel from one end to the other. One way of generating such paths thus entails optimizing path length or solving an equivalent root-finding problem. In fact, many forward rendering techniques that target specular interreflections exploit some form of gradient information computed based on the light path geometry. This idea was pioneered by Mitchell and Hanrahan [119], who render caustics from implicitly defined curved reflectors using interval arithmetic and Newton-Raphson iteration to find extremal one-bounce specular reflection paths connecting a given pair of vertices. We also experimented with a similar approach in an early stage of this work and found that the highly conservative nature of interval arithmetic often causes intervals to be too large to allow for systematic pruning of the solution space. Another approach involves precomputed hierarchical data structures that partition triangle meshes and bound the positions and normals of each subtree. Building on such a position-normal hierarchy, Walter et al. [120] propose an efficient pruning strategy for specular chains with a single specular refraction (e.g. underwater caustics).

Other forward rendering applications also make use of light path gradients, e.g. gradient-domain rendering [121], adaptive sampling and reconstruction [122], and the interpolation of local solutions of diffuse [123] and non-diffuse [124] global illumination. Analytic and approximate ray-space derivatives are also part of the standard graphics pipeline that computes texture-space footprints to filter texture lookups via mip-mapping and elliptically weighted averages [125]. Such footprints can be further propagated following interaction with smooth [126] and rough [127] materials.

Our algorithm builds on top Jakob's specular path space manifolds [128] that analyze the differential light path geometry of valid specular paths. It was originally developed as part of a Markov chain Monte Carlo perturbation strategy called *manifold exploration (ME)* [116] that makes proposals in the framework of Metropolis Light Transport [7]. The natural constraint representation [129, 130] significantly expands on this idea in the more general case of multiple glossy interactions. One downside of these perturbation strategies is that they can only make very small changes to a specular path; exploration of the larger space relies on many repetitions of this basic operation. When the geometry is characterized by high-frequency detail, the manifold of valid specular paths tends to become very complex, and simple local steps are insufficient for global exploration. The theory of specular manifolds is also highly related to the previously proposed framework by Chen and Arvo [131] who used perturbation theory to render specular reflections on curved surfaces.

The *manifold next event estimation (MNEE)* technique of Hanika et al. [132] applies the ME equation-solving iteration in a pure Monte Carlo context. Starting with an incorrect initial specular path, MNEE iteratively attempts to walk towards a valid solution via projection and tangential steps on the specular manifold, similar to standard numerical root-finding. Due to a simple deterministic initialization, it can find at most a single solution, which works well on smooth geometry (e.g. spheres) but breaks down in more challenging cases. MNEE was later adapted as a connection strategy for bidirectional path tracing [133] and other recent work has explored the potential of next event estimation strategies that sample multiple vertices in the context of volume rendering [134, 135].

Our proposed method, named *specular manifold sampling (SMS)*, is a generalization of the MNEE approach: using a stochastic initialization and an unbiased sample weight estimator, we are able to find solutions on complex geometry where manifold-based techniques were previously inapplicable. We also demonstrate that SMS straightforwardly generalizes to the related problem of rendering glints.

Previous specialized glint rendering methods implement an effective BRDF that averages the behavior of a high-frequency specular surface over a given surface region [136, 137, 138]. The method of Yan et al. [137] does so by constructing position-normal hierarchies in texture space to solve an approximate form of this problem for normal-mapped surfaces. These tree data structures tend to become extremely large, requiring tens of gigabytes of memory in our experiments. Procedural glints [139, 140, 141, 142, 143, 144] can be rendered with a much lower storage footprint, but are of course significantly less flexible. Recent work on glint rendering has focused on incorporating wave-optical effects [11], and directional bases for filtering product integrals involving glints [145].

We note that some of the discussed caustic and glint rendering techniques (Mitchell and Hanrahan [119], Walter et al. [120], and Yan et al. [136, 137]) find all possible solutions within their supported path classes and are thus fully deterministic. This yields converged renderings with a single sample per pixel when the scene contains no other sources of variance, but this is rarely the case in practice. In contrast, our method stochastically samples individual solutions; the added variance due to this random decision can then be reduced along with complementary sources of variance (environment mapping, indirect illumination, depth of field, and so on).

Concurrently to our work, Loubet et al. [6] developed an algorithm inspired by Walter et al.'s method [120] that also generalizes to rendering of specular reflections and glints. They achieve this by deriving analytic expressions for the radiance due to interaction with a single triangle with a microfacet BSDF.

Since the original publication of our method [1], Pediredla et al. [146] used a method similar to ours in the context of refractive radiative transfer [147]. A root-finding method related to SMS was also used by Wang et al. [148] in order to render light paths consisting exclusively of specular interactions, e.g. a series of refractions connecting a point light and a pinhole camera.

The remainder of this section covers more details regarding the specular path space manifolds [128] underlying our method. In particular, we review the definition of the specular manifold (Section 3.1.1), the ME perturbation strategy (Section 3.1.2), and the MNEE sampling technique (Section 3.1.3). Though our method is mostly aimed at purely specular light transport, like previous works it can also be applied to glossy materials (Section 3.1.4). Lastly, we highlight the underlying challenges of glint rendering together with previous solution approaches (Section 3.1.5).

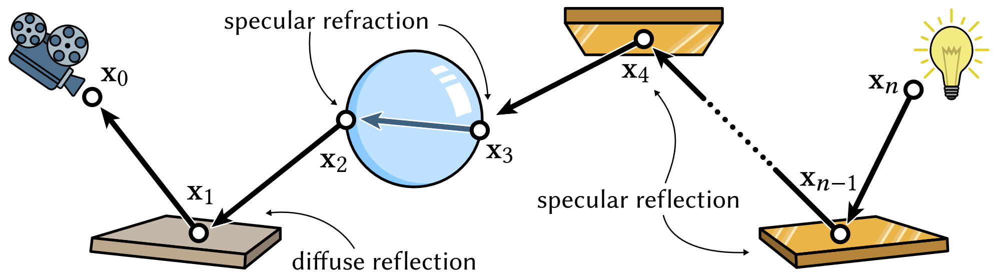


Figure 3.2: A light path $\bar{\mathbf{x}}$ between the camera and emitter position that involves a series of specular reflections and refractions.

3.1.1 Specular path-space manifolds

Consider a light transport path $\bar{\mathbf{x}} = \mathbf{x}_0, \ldots, \mathbf{x}_n$ containing a chain of specular vertices between two non-specular endpoints $\mathbf{x}_1$ and $\mathbf{x}_n$. For example, $\mathbf{x}_1$ could be a shading point on a diffuse surface, and $\mathbf{x}_n$ a position on a light source (Figure 3.2). Although this path is an element of a large and high-dimensional path space (Section 2.3.3), it effectively lies on a much lower-dimensional subspace, since each specular vertex imposes physical constraints (in the form of Dirac delta functions) that collapse some dimensions of the ambient space. For example, when transmission takes place at a vertex, it must satisfy the law of refraction, removing all its continuous degrees of freedom.

Jakob and Marschner [116] characterize these constraints via a function $\mathbf{c}_i \in \mathbb{R}^2$ associated with each vertex $\mathbf{x}_i$. Concretely, $\mathbf{c}_i$ projects the generalized half-vector $\boldsymbol{\omega}_\mathrm{h}(\mathbf{x}_i, \boldsymbol{\omega}, \boldsymbol{\omega}')$ by Walter et al. [31], see also Equation (2.38), onto the local tangent space at the vertex:

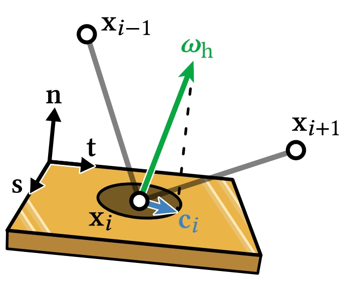
$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{T}(\mathbf{x}_i)^T \; \boldsymbol{\omega}_\mathrm{h}(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}), \tag{3.1}$$

$$\boldsymbol{\omega}_\mathrm{h}(\mathbf{x}_i, \boldsymbol{\omega}, \boldsymbol{\omega}') = \frac{\eta(\mathbf{x}_i, \boldsymbol{\omega})\, \boldsymbol{\omega} + \eta(\mathbf{x}_i, \boldsymbol{\omega}')\, \boldsymbol{\omega}'}{\|\eta(\mathbf{x}_i, \boldsymbol{\omega})\, \boldsymbol{\omega} + \eta(\mathbf{x}_i, \boldsymbol{\omega}')\, \boldsymbol{\omega}'\|}. \tag{3.2}$$

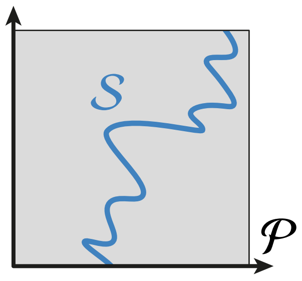
Here, $\mathbf{T}(\mathbf{x}_i)$ is a 3×2 matrix of tangent vectors $\mathbf{s}$ and $\mathbf{t}$ at $\mathbf{x}_i$, and $\eta(\mathbf{x}, \boldsymbol{\omega})$ is the refractive index associated with position $\mathbf{x}$ and direction $\boldsymbol{\omega}$. The configuration is illustrated in Figure 3.3 (a) for the case of specular reflection.

The constraint function takes on zero values, i.e. $\mathbf{c}_i = (0, 0)^T$, when the vertices $\mathbf{x}_{i-1}$, $\mathbf{x}_i$, and $\mathbf{x}_{i+1}$ are aligned in order for the relevant physical laws to hold. Note that the use of the generalized half-vector succinctly covers both cases of specular reflection (2.19) and refraction (2.20).

A valid specular path can then be summarized as a root of the combined function $\mathbf{C}(\bar{\mathbf{x}}) = (\mathbf{c}_2, \ldots, \mathbf{c}_{n-1})^T$ that stacks all specular path constraints. The *specular manifold* $\mathcal{S} = \{\bar{\mathbf{x}} \mid \mathbf{C}(\bar{\mathbf{x}}) = \mathbf{0}\}$ is the set of all such light paths and forms a subset of the full path space $\mathcal{P}$, as visualized abstractly in Figure 3.3 (b).



(a) Local specular constraint



(b) Path space illustration

Figure 3.3: **(a)** The specular constraint by Jakob and Marschner [116] projects the (generalized) half-vector ω_h onto the local tangent frame $(\mathbf{s}, \mathbf{t})$. A configuration where $\mathbf{c}_i = (0,0)^T$ therefore corresponds to a valid specular reflection or refraction. **(b)** An abstract visualization of the full path space $\mathcal{P}$ with a lower-dimensional specular manifold $\mathcal{S}$ embedded inside.

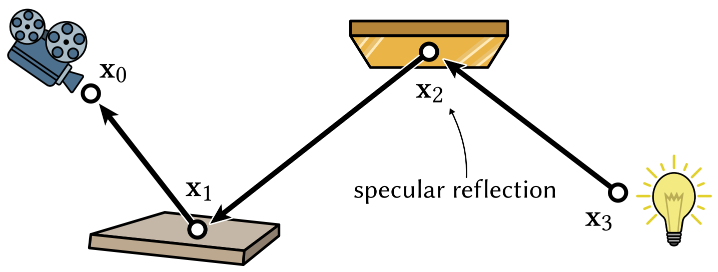


Figure 3.4: An example light path $\bar{\mathrm{x}} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ of length three with one specular reflection at vertex $\mathbf{x}_2$.

This collapse of dimensions caused by the specular constraints also influences the computation of the path contribution function (2.59) over path space. Consider the example light path of length three in Figure 3.4. Ordinarily, the corresponding contribution involves the BSDF f_s evaluated for each surface interaction and one geometry term G between each set of subsequent vertices:

$$\begin{aligned}\int_{\mathcal{M}}\int_{\mathcal{M}}\int_{\mathcal{M}}\int_{\mathcal{M}} & L_\mathrm{e}(\mathbf{x}_3 \to \mathbf{x}_2)\, G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3) \\ & \cdot f_s(\mathbf{x}_3 \to \mathbf{x}_2 \to \mathbf{x}_1)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, f_s(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0) \\ & \cdot G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1)\, W_\mathrm{e}(\mathbf{x}_1 \to \mathbf{x}_0)\, \mathrm{d}A(\mathbf{x}_3)\, \mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_0). \end{aligned} \tag{3.3}$$

Recall from Section 2.3.2 that the geometric terms encode a change of variables between the projected solid angle and area measure of two vertices, as well as the binary visibility function V:

$$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = V(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \cdot \left| \frac{\mathrm{d}\Omega^\perp_{\mathbf{x}_1}(\overrightarrow{\mathbf{x}_1\mathbf{x}_2})}{\mathrm{d}A(\mathbf{x}_2)} \right|. \tag{3.4}$$

Due to the specular vertex $\mathbf{x}_2$, the path throughput includes a Dirac delta function and one of the integrals disappears:

$$\begin{aligned}\int_{\mathcal{M}}\int_{\mathcal{M}}\int_{\mathcal{M}} & L_e(\mathbf{x}_3 \to \mathbf{x}_2)\, F(\mathbf{x}_3 \to \mathbf{x}_2 \to \mathbf{x}_1) \\ & \cdot G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3)\, f_s(\mathbf{x}_2 \to \mathbf{x}_1 \to \mathbf{x}_0)\, G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\ & \cdot W_e(\mathbf{x}_1 \to \mathbf{x}_0)\, \mathrm{d}A(\mathbf{x}_3)\, \mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_0). \end{aligned} \tag{3.5}$$

This causes two further changes: the BSDFs at specular vertices are replaced with a reflectance value F (usually the Fresnel terms that are part of the specular BSDFs, see Sections 2.2.3 and 2.2.4), and the geometry term must be replaced by a more general version that accounts for specular interreflection involving multiple vertices. Like before, this term still encodes a change of variables: this time, it relates the projected solid angle measure at one end, and the area measure on the other end of the specular chain:

$$G(\mathbf{x}_1 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_k) = V(\ldots) \cdot \left| \frac{\mathrm{d}\Omega^{\perp}_{\mathbf{x}_1}(\overrightarrow{\mathbf{x}_1\mathbf{x}_2})}{\mathrm{d}A(\mathbf{x}_k)} \right| = V(\ldots) \cdot \left| \frac{\mathrm{d}\Omega^{\perp}_{\mathbf{x}_1}(\overrightarrow{\mathbf{x}_1\mathbf{x}_2})\, \mathrm{d}A(\mathbf{x}_2)}{\mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_k)} \right|. \tag{3.6}$$

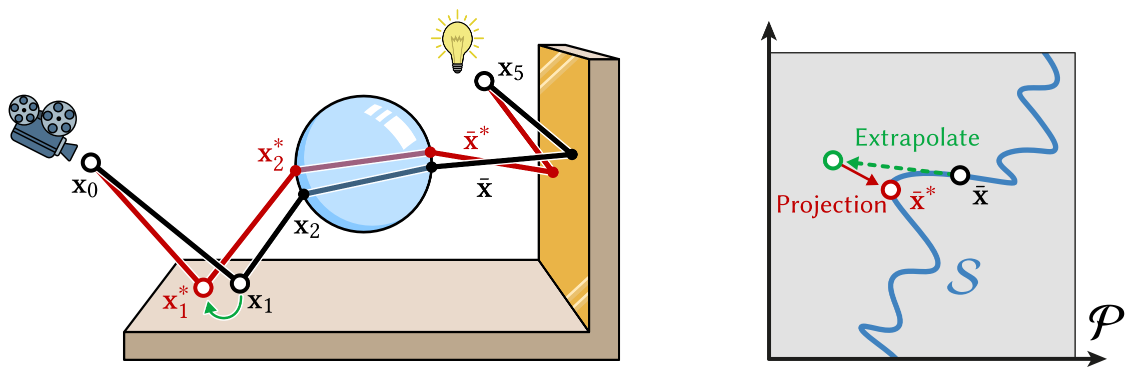
This expression can either be computed using ray differentials [126] or using the derivatives of the constraint function as described in the next section.

3.1.2 Manifold exploration

The first application of path-space manifolds was the ME method by Jakob and Marschner [116] which is a mutation strategy for use in the MLT algorithm [7]. As the name suggests its goal is to “explore” the regions surrounding valid specular light paths (sampled by other methods) while staying on the specular manifold. The high-level idea is shown in Figure 3.5 where a small perturbation of an endpoint leads to a global update step to keep all specular constraints fulfilled.

This is accomplished using the *Implicit Function Theorem (IFT)* [149] which describes the neighborhood of a given path configuration using the Jacobian matrix $J_C(\bar{\mathbf{x}})$ of the constraint function. As illustrated in Figure 3.6, this matrix has a tri-diagonal block structure as each constraint only depends on the corresponding vertex and its two neighbors along the path.

The IFT allows us to define a tangent space of the specular manifold that we can use to navigate the high dimensional space in a more informed manner. In particular, the tangents specify—to a first-order approximation—how the specular vertices should adjust due to changes in either of the two non-specular endpoints. They can be computed



(a) MLT mutation strategy **(b) Path space view**

Figure 3.5: **(a)** As part of a MLT mutation, we want to move a non-specular end point $\mathbf{x}_1$ of light path $\bar{\mathbf{x}}$. The manifold exploration method can apply a global change to the remaining specular vertices such that this updated path $\bar{\mathbf{x}}'$ still lies on the specular manifold. **(b)** Abstract visualization of the underlying manifold walk algorithm. We first take a step along the tangent space of $\mathcal{S}$ before projecting back to a valid light path.

easily from the columns of the Jacobian matrix as

$$\frac{\partial(\mathbf{x}_2 \dots \mathbf{x}_{n-1})}{\partial \mathbf{x}_1} = -\left(\frac{\partial \mathrm{C}}{\partial(\mathbf{x}_2 \dots \mathbf{x}_{n-1})}\right)^{-1} \cdot \frac{\partial \mathrm{C}}{\partial \mathbf{x}_1} \tag{3.7}$$

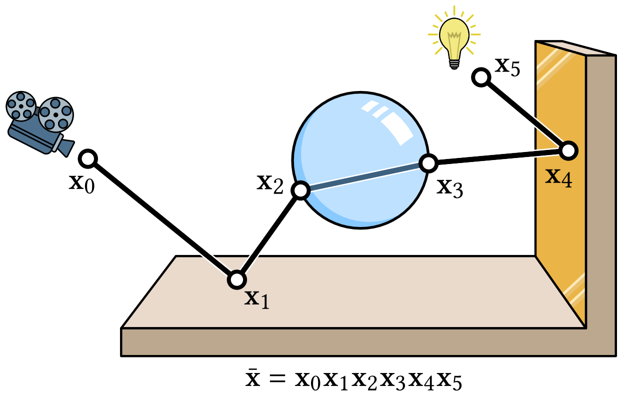
and

$$\frac{\partial(\mathbf{x}_2 \dots \mathbf{x}_{n-1})}{\partial \mathbf{x}_n} = -\left(\frac{\partial \mathrm{C}}{\partial(\mathbf{x}_2 \dots \mathbf{x}_{n-1})}\right)^{-1} \cdot \frac{\partial \mathrm{C}}{\partial \mathbf{x}_n}, \tag{3.8}$$

see also again Figure 3.6. The matrix inversion here can be computed efficiently due to its sparse structure, even in the case of long specular chains where n is large.

For convenience it make sense to express the function C and its derivatives not based on the world space 3D vertex positions $\mathbf{x}_i$, but using a 2D coordinate system based on their surface position. This way, the input and output of each vertex constraint $\mathbf{x}_i$ have matching dimensions and each block in the matrix has size 2×2. The UV texture coordinates already provide such a 2D coordinate system and are thus a logical choice. Furthermore, thanks to the IFT, a purely local parameterization (e.g. per-triangle UV coordinates) suffices to compute all necessary derivatives. We evaluate all derivatives using manually designed code due to their relative simplicity. Dedicated tools for automatic differentiation could be used alternatively.

Depending on the choice of coordinate system, movement along the manifold tangents will not only step off the specular manifold, but will generally produce vertices that do not even lie on the scene geometry. To recover suitable vertex positions, an additional reprojection step is required, either via ray tracing from one of the endpoints [116], or



$\mathbf{x}_5$
$\mathbf{x}_0$
$\mathbf{x}_2$
$\mathbf{x}_3$
$\mathbf{x}_4$
$\mathbf{x}_1$
$\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4\mathbf{x}_5$

$$C(\bar{\mathbf{x}}) = \begin{pmatrix} \mathbf{c}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \\ \mathbf{c}_3(\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) \\ \mathbf{c}_4(\mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5) \end{pmatrix}$$

$$J_C(\bar{\mathbf{x}}) = \left(\begin{array}{c:ccc:c} \boxed{\frac{\partial \mathbf{c}_2}{\partial \mathbf{x}_1}} & \boxed{\frac{\partial \mathbf{c}_2}{\partial \mathbf{x}_2}} & \boxed{\frac{\partial \mathbf{c}_2}{\partial \mathbf{x}_3}} & 0 & 0 \\ 0 & \boxed{\frac{\partial \mathbf{c}_3}{\partial \mathbf{x}_2}} & \boxed{\frac{\partial \mathbf{c}_3}{\partial \mathbf{x}_3}} & \boxed{\frac{\partial \mathbf{c}_3}{\partial \mathbf{x}_4}} & 0 \\ 0 & 0 & \boxed{\frac{\partial \mathbf{c}_4}{\partial \mathbf{x}_3}} & \boxed{\frac{\partial \mathbf{c}_4}{\partial \mathbf{x}_4}} & \boxed{\frac{\partial \mathbf{c}_4}{\partial \mathbf{x}_5}} \end{array}\right)$$

$$\underbrace{}_{\frac{\partial C}{\partial \mathbf{x}_1}} \quad \underbrace{}_{\frac{\partial C}{\partial(\mathbf{x}_2\mathbf{x}_3\mathbf{x}_4)}} \quad \underbrace{}_{\frac{\partial C}{\partial \mathbf{x}_5}}$$

Figure 3.6: **Left**: An example light path with a set of specular vertices $\mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4$. **Right**: The constraint function C and its Jacobian matrix J_C with a tri-diagonal block structure. The illustration partitions the matrix into three components that are relevant for computing the tangent space of the specular manifold.

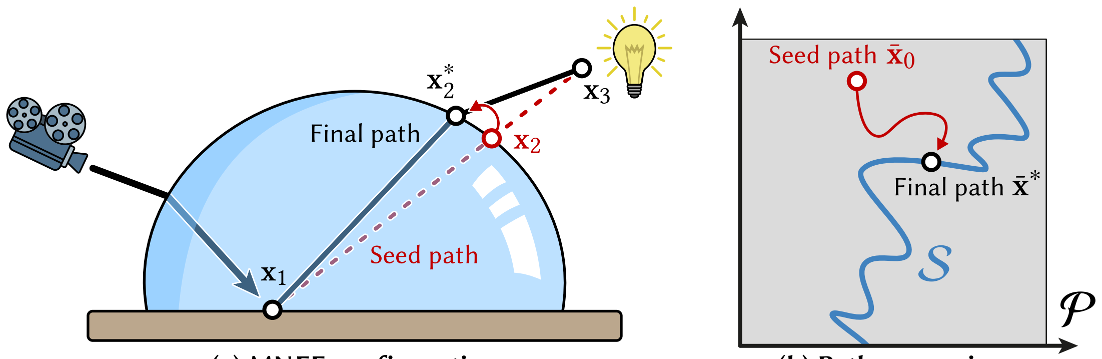
directly moving to the nearest surface location [129, 132].

Consider again Figure 3.5 for a more detailed discussion of how this is applied to the MLT mutation strategy. First, an offset $\mathbf{x}_1 \rightarrow \mathbf{x}_1^*$ is applied to a non-specular endpoint of the specular path $\bar{\mathbf{x}}$. This breaks the light path in the sense that it will no longer fulfill the specular constraint at $\mathbf{x}_2$. ME then performs a global update to all vertex positions to move the path back to the manifold. It achieves this with two steps:

Extrapolation: Translate the endpoint offset $\mathbf{x}_1 \rightarrow \mathbf{x}_1^*$, into offsets along the specular manifold tangent (3.7) and update all specular vertices accordingly. Note that the new intermediate positions $\mathbf{x}_i'$ are now generally disconnected from scene geometry.

Projection: Project the specular vertices back onto the geometry via ray tracing: starting from the endpoint $\mathbf{x}_n$, connect to the last specular vertex $\mathbf{x}_{n-1}'$ and trace $n-1$ more bounces until a new path is formed with tentative endpoint $\mathbf{x}_1'$.

While this usually moves the vertex $\mathbf{x}_1$ towards $\mathbf{x}_1^*$, its new location is generally not yet at the desired target ($\mathbf{x}_1' \neq \mathbf{x}_1^*$). Jakob and Marschner therefore let these two steps repeat iteratively until the process converges to a valid path $\bar{\mathbf{x}}^*$ that includes the desired vertex $\mathbf{x}_1^*$. This is called a *manifold walk* and resembles standard numerical root-finding algorithms like Newton's method. In particular, it also exhibits quadratic convergence when we are sufficiently close to a solution, which is usually the case here as the perturbations $\mathbf{x}_1 \rightarrow \mathbf{x}_1^*$ tend to be small. However, there are no guarantees and the manifold walk might also fail to converge to a suitable path.



(a) MNEE configuration **(b) Path space view**

Figure 3.7: **(a)** Finding a specular refraction through a spherical interface using manifold next event estimation. Note how the (straight-line) seed path is usually very close to the valid specular path in this scenario, which means that the manifold walk has a high probability of successfully converging to a valid solution. **(b)** Abstract path space visualization of the process. The manifold walk tries to find a valid path on $\mathcal{S}$ using an (initially invalid) seed path.

We note that the Jacobian J_C is also useful for the computation of the generalized geometry term (3.6) from above as

$$G(\mathbf{x}_1 \leftrightarrow \ldots \leftrightarrow \mathbf{x}_k) = V(\ldots) \cdot \left| \mathbf{P}_2 \cdot \left(\frac{\partial \mathrm{C}}{\partial(\mathbf{x}_2 \ldots \mathbf{x}_{k-1})} \right)^{-1} \cdot \frac{\partial \mathrm{C}}{\partial \mathbf{x}_k} \right| \cdot G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \tag{3.9}$$

$$= V(\ldots) \cdot \left| \mathbf{P}_{k-1} \cdot \left(\frac{\partial \mathrm{C}}{\partial(\mathbf{x}_2 \ldots \mathbf{x}_{k-1})} \right)^{-1} \cdot \frac{\partial \mathrm{C}}{\partial \mathbf{x}_1} \right| \cdot G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k), \tag{3.10}$$

where $G(\mathbf{x} \leftrightarrow \mathbf{x}')$ is the usual geometry term between two vertices and $\mathbf{P}_i$ is a projection matrix that extracts the 2×2 block associated with vertex i. See Jakob's thesis [128] for details.

3.1.3 Manifold next event estimation

Despite its origin as an MCMC method, manifold walks can also be applied in a regular Monte Carlo context, as demonstrated by the MNEE method by Hanika et al. [132]. Instead of applying a mutation to an existing valid light path, they start with a tentative *seed path* $\bar{\mathbf{x}}_{(0)}$ that is not yet on the specular manifold. The manifold walk is then equivalent to a root-finding method that attempts to find a point on the specular manifold. Solutions to the equation $\mathrm{C}(\bar{\mathbf{x}}) = 0$ can be found using the multivariate Newton-Raphson method

$$\bar{\mathbf{x}}_{(i+1)} = \bar{\mathbf{x}}_{(i)} - J_\mathrm{C}(\bar{\mathbf{x}}_{(i)})^{-1} \cdot \mathrm{C}(\bar{\mathbf{x}}_{(i)}) \tag{3.11}$$

and, like previously, an additional projection step is needed after each iteration to ensure the updated vertex positions lie on actual scene geometry.

```
# Estimate radiance travelling from emitter position $x_n$ to shading point $x_1$.
def mnee($x_1, x_n$):
    # Collect specular vertices along ray $x_1 \to x_n$
    $s = (x_2, \dots, x_{n-1})$ = specular_ray_intersections($x_1, x_n$)
    # Run Newton solver to find valid connection
    $s^* = (x_2^*, \dots, x_{n-1}^*)$ = manifold_walk($x_1, s, x_n$)
    # Contribution divided by PDF of sampling the emitter position
    return $F(s^*) \cdot G(x_1 \leftrightarrow x_2^* \leftrightarrow \dots \leftrightarrow x_{n-1}^* \leftrightarrow x_n) \cdot L_e(x_n) \,/\, p(x_n)$
```

Algorithm 3.1: Manifold next event estimation [132].

MNEE is specific to refractive chains and initializes the Newton iteration with a seed path corresponding to a straight-line connection of a given shading point and an emitter position (Figure 3.7 and Algorithm 3.1).

For caustic rendering applications (e.g. Figure 3.8), the probability of finding a valid path in this way is generally much higher compared to standard sampling techniques in a path tracer, and the variance of rendered images is thus reduced drastically. However, the manifold walk can also *fail* in case the seed path is too far away from any solution, for instance if the geometry is very complex.

Prior to MNEE, the fundamental problem impeding the integration of manifold walks into pure Monte Carlo methods had been the determination of the probability $p(\bar{\mathbf{x}}^*)$ of finding a solution path $\bar{\mathbf{x}}^*$ given an initial state $\bar{\mathbf{x}}_{(0)}$ chosen with density $p(\bar{\mathbf{x}}_{(0)})$. This probability is given by a marginalization over the path space $\mathcal{P}$ [132]:

$$p(\bar{\mathbf{x}}^*) = \int_{\mathcal{P}} p(\bar{\mathbf{x}}^* \,|\, \bar{\mathbf{x}}_{(0)})\, p(\bar{\mathbf{x}}_{(0)})\, \mathrm{d}\mu(\bar{\mathbf{x}}_{(0)}). \tag{3.12}$$

The computation of this expression is mathematically daunting because $p(\bar{\mathbf{x}}^* \,|\, \bar{\mathbf{x}}_{(0)})$ encapsulates the behavior of an unpredictable equation-solving iteration. MNEE sidesteps this problem using two clever tricks: First, it always uses the same initialization, turning $p(\bar{\mathbf{x}}_{(0)})$ into a Dirac delta function. Second, it relies on MIS (Section 2.4.4) to fall back to standard path tracing when the manifold walk fails. In this case, the remaining term $p(\bar{\mathbf{x}}^* \,|\, \bar{\mathbf{x}}_{(0)})$ cancels out in the estimator.

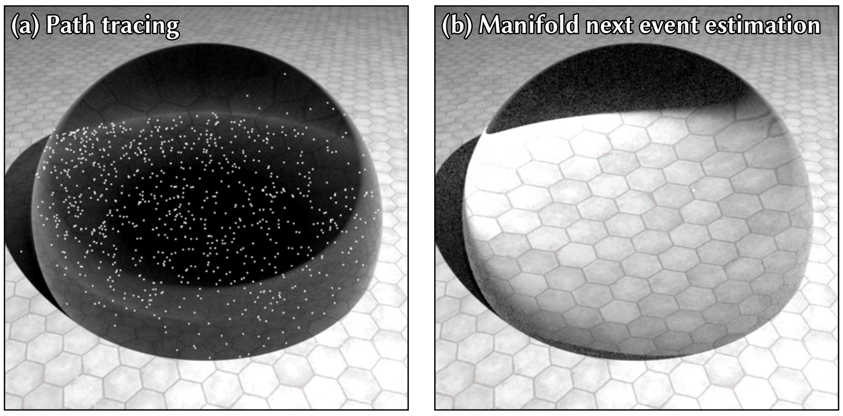


Figure 3.8: Equal time comparison of path tracing **(a)** and MNEE **(b)** applied to a dielectric sphere lit by a small area light. The path tracer struggles due to the SDS paths, while MNEE is easily able to sample the specular transport by refining the straight-line initialization using the Newton solver.

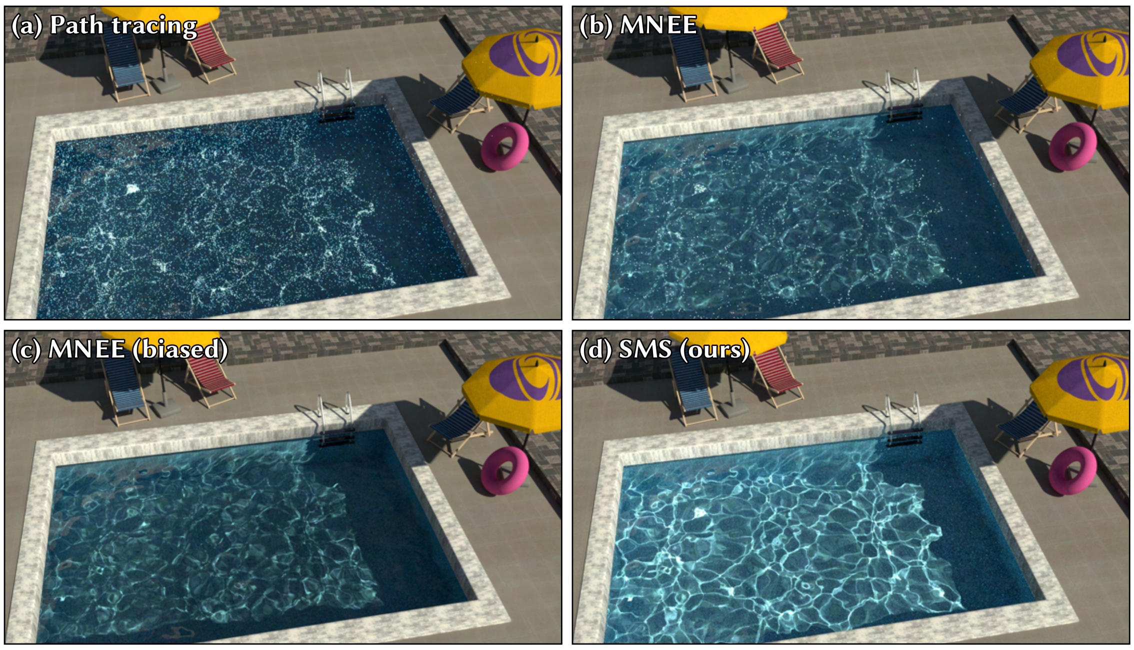


Figure 3.9: Equal time comparison of various methods on a swimming pool test scene with complex (normal-mapped) specular geometry. **(a)** Path tracing is not suited for rendering the challenging SDS paths present in this scene and produces high variance. **(b)** MNEE can reliably find *one* solution path (due to its deterministic initialization) but still suffers from high-frequency variance because it falls back to a path tracer for all other light connections. **(c)** In order to clean up the noisy image, all outliers due to the path traced solutions could be discarded. This leads to severe bias in the form of energy loss. **(d)** Our proposed specular manifold sampling method, significantly outperforms MNEE and can find all possible solution paths.

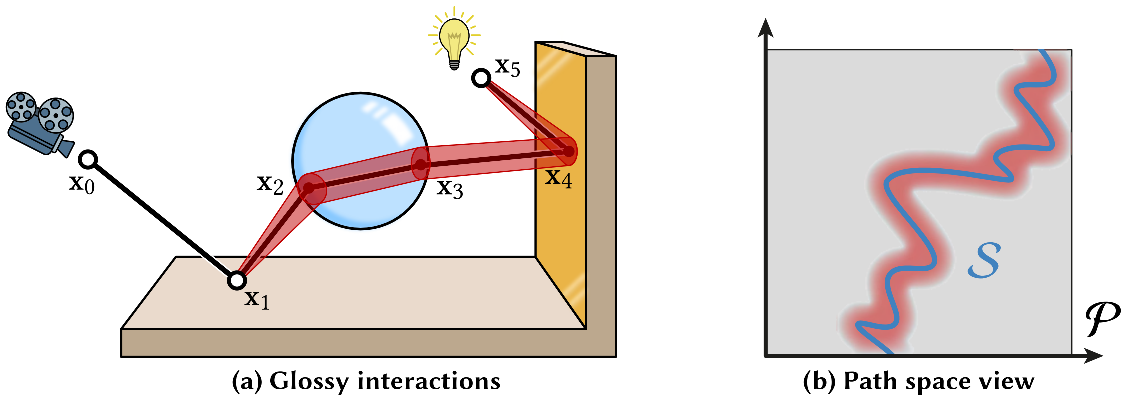


Figure 3.10: Generalizing to glossy materials admits a continuum of solution light paths between the path endpoints **(a)** (red outline) which means all parts of path space in a region around the specular manifold corresponds to valid paths **(b)**.

However, MNEE also suffers from several fundamental limitations that we intend to address with our method: because MNEE can find at most a single specular path, it does little to reduce variance when there are multiple solutions. Most of the examples in the original paper [132] are extremely simple shapes such as spheres[1] or cylinders, where a single solution indeed suffices to render most specular paths (Figure 3.8). However, this is clearly no longer the case when the specular geometry is more complex, as shown on a more challenging swimming pool scene in Figure 3.9 (a–c). Also, while the straight-line initialization proposed by Hanika et al. works well for refractive caustics, it is unclear how it could be generalized to the reflection case.

Our method tries to address these problems and can sample all possible solution paths, including reflections. A preview for the same swimming pool test scene is shown in Figure 3.9 (d).

3.1.4 Extension to glossy surfaces

The framework of specular path-space manifolds can also be extended to glossy materials, e.g., ones modeled using rough microfacet models. While high surface roughness significantly simplifies the light transport and standard emitter sampling strategies become viable, materials with low roughness can still benefit from a specialized treatment.

Surface roughness relaxes the set of valid connections to a continuous space where the manifold is now a more general, blurry region of path space, see Figure 3.10.

[1]Rendering of caustics in 3D models of human eyes constitutes an important use case of MNEE in the entertainment industry [150].

Jakob and Marschner [116] propose the following generalization of their specular path constraints (3.2) to the rough case:

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{T}(\mathbf{x}_i)^T \, \boldsymbol{\omega}_\mathrm{h}(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}) - \mathbf{T}(\mathbf{x}_i)^T \, \mathbf{m}_i. \tag{3.13}$$

Here, $\mathbf{m}_i$ corresponds to a microfacet normal (Section 2.2.5) that is also projected to the same tangent space. By setting it to a fixed value (e.g. sampled from a microfacet distribution) this still gives a clearly defined constraint that can be interpreted as a "specular" path involving specific microfacets instead of the (macroscopic) geometric normals. The generalized solution space is referred to as an *offset specular manifold*.

In the context of MLT mutations [116], this now allows additional freedom where we cannot only move the vertex positions on the specular path but also adjust the offset microfacets $\mathbf{m}_i$ in order to explore interesting regions of path space. These types of mutations were further extended by Kaplanyan et al. [129] and Hanika et al. [130] to achieve higher robustness and better exploration.

MNEE [132] can also be generalized to support rough materials. Here, one microfacet normal is sampled for each glossy vertex along the seed path before letting the Newton solver converge to a solution path that fulfills Equation (3.13) at all vertices.

In this thesis we mostly focus on the purely specular case, so please refer to the corresponding articles for more details, e.g., concerning the computation of the glossy path contribution.

3.1.5 Reflections from glinty microstructures

Another type of challenging specular light transport that we address in this chapter is reflection from high-frequency specular microstructures. This case is relevant when rendering small surface imperfections such as bumps, scratches, or other types of glints. Consider the setup in Figure 3.11 and recall from Section 2.5.1 that standard emitter sampling strategies are very ineffective on specular surfaces. This means we have to mainly rely on BSDF sampling where reflected rays need to hit the light sources. This is very unlikely for small, concentrated emitters like the sun and thus, configurations like these can cause unacceptably high variance.

A similar reflection geometry arises in the case of microfacet models (Section 2.2.5), though the two operate at different scales. Microfacets produce smooth, non-glinty highlights as the surface variations are *infinitesimally small* and cannot be discerned by an observer. These can be handled using default emitter sampling techniques. In contrast,

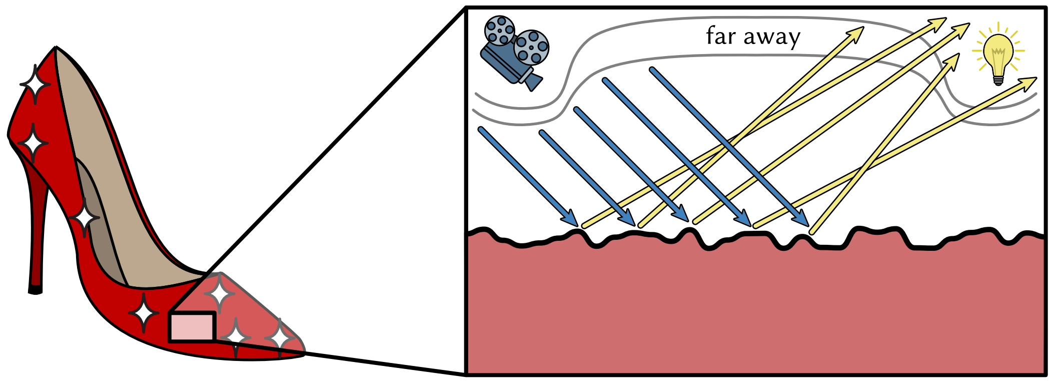


Figure 3.11: Rendering of glinty surfaces poses similar challenges as the previously discussed SDS light paths. When focusing on a small surface patch (zoomed-in view on the right), reflected rays only have a small probability of encountering small or distant light sources. As in the case of SDS paths, bidirectional techniques cannot efficiently handle this case, since they struggle to hit the tiny camera aperture or are unable to connect two subpaths that meet on the specular surface.

glinty highlights are the result of microstructure that can be directly perceived by the viewer.

Nonetheless, specialized glint rendering techniques generally build on top of microfacet theory and introduce a generalized form of the NDF that adapts to the scale of the problem. These approaches analyze the behavior of a specular microstructure over a small surface patch (usually the area within a single image pixel, a.k.a. the *pixel footprint*) to determine an averaged BRDF that accounts for glints.

For example, Yan et al. [136] introduce the $\mathcal{P}$-NDF[2] that takes this role for the case of microgeometry defined explicitly via a high-resolution normal map, see Figure 3.12. Evaluating a $\mathcal{P}$-NDFs is no easy task however. The initial method [136] explicitly integrates over a finely discretized normal map which can be very slow. A follow up work [137] improves efficiency considerably by converting the normal map into a spatio-directional 4D tree. However this comes at the cost of very high memory consumption to store the precomputed data structure.

Another type of approach was explored by Jakob et al. [139]. Here, a discretized version of the NDF is used where a large (but finite) number of microfacet normals represent the microgeometry. The NDF evaluation is then replaced by a counting process, i.e., we need to know how many of the individual normals have a suitable position and orientation for a given camera and light configuration.

While storing and querying a large number of normals might seem impractical at first, an efficient implementation is possible by generating and counting the normals

[2]The $\mathcal{P}$ stands for surface "patch" here.

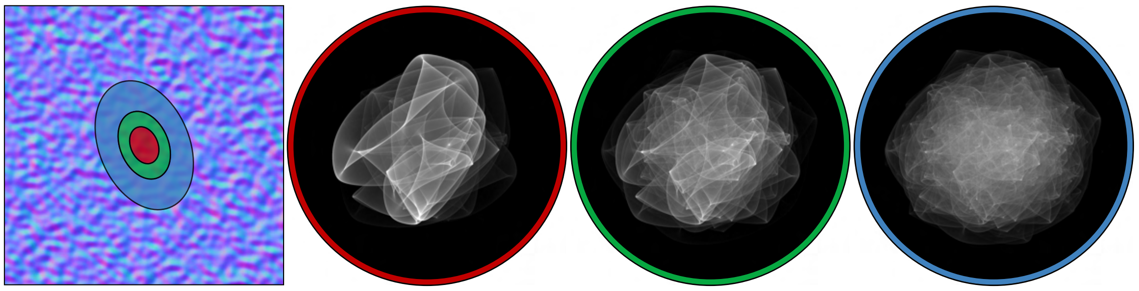


Figure 3.12: One approach to glint rendering is a microfacet model with a $\mathcal{P}$-NDF [136]. **Left**: a zoomed-in view on a high-resolution normal map representing a Gaussian height field. Elliptical pixel footprints of varying sizes (e.g. produced by viewing the surface from different distances) are overlaid on top. **Right**: corresponding $\mathcal{P}$-NDFs, visualized on the projected disk. Note the intricate details that is ultimately responsible for the glinty appearance during rendering. As the footprint size increases, the distributions tend to a smooth function. Standard analytic microfacets are the limit case of this averaging process for specific surface statistics.

only procedurally during the evaluation routine. This way, no storage of a data structure is necessary and memory consumption is low. The downside is that only certain types of microgeometry can be represented this way, as all discrete microfacets are still assumed to be drawn from one of the available smooth models.

Our proposed method lies somewhere in between these two works. We also use an explicit normal map representation but do not precompute any heavy data structures. Instead of evaluating the full $\mathcal{P}$-NDF we treat the problem from the perspective of light transport and sample individual solution light paths, leveraging the framework of specular manifolds for the first time in the context of glints.

3.2 Main method

We now turn to the proposed SMS method, initially focusing on a simple unbiased algorithm that generalizes to cases where multiple specular paths connect two given endpoints. We then present an alternate version with reduced variance, but at the cost of nonzero bias.

Section 3.3 will discuss several extensions to this: the first replaces the specular constraints of Jakob and Marschner [116] with improved variants, the second samples paths in two stages to improve performance on normal-mapped surfaces, and the last streamlines the algorithm for glint rendering. Many of these extensions are modular and can be combined as desired with both unbiased or biased SMS. Figure 3.13 shows a preview of several combinations applied to the problem of rendering refractive caustics, comparing our results to MNEE and a brute-force reference.

3.2.1 Finding multiple solutions

We initially restrict ourselves to specular chains with a single smooth reflection or refraction. Our technique however extends to chains with surface roughness and multiple interactions analogously to Hanika et al. [132]; see Sections 3.4.6 and 3.4.7. In this restricted setting, there is a discrete and finite set[3] of solutions connecting two given endpoints. Our approach to finding them is simple: whereas MNEE performs manifold walks using a fixed initialization, SMS randomly samples the initial guess from a probability distribution $p(\bar{\mathbf{x}}_{(0)})$. Newton's method exhibits quadratic convergence when the starting point is sufficiently close to a root, hence all solutions will be found with a nonzero probability[4]—however, the probability of successful convergence is unknown.

There is a great degree of latitude in the choice of an initial guess: we could uniformly generate positions on specular surfaces, or sample the BSDF of the preceding vertex $\mathbf{x}_1$ and find starting points via ray tracing. Regardless of what approach is used, we assume that the implementation of this sampling strategy takes two uniformly dis-

[3]We note that cases with a continuous 1D subspace of solutions can be constructed, for instance when $\mathbf{x}_1$ and $\mathbf{x}_3$ lie at the center-line of a perfect cylindrical mirror. This is of no relevance for rendering natural scenes, since an arbitrarily small perturbation of the surface geometry would break the symmetries that are needed to create a 1D solution subspace. We ignore this corner case similarly to prior work [120].

[4]While this generally holds true for Newton's method, our algorithm involves an additional projection operation that moves vertices back to valid geometry after each solver iteration. This could theoretically result in solution paths that cannot be found at all, though we did not find such cases in any of our tests. A more rigorous analysis or proof of the solver convergence would be a valuable future contribution.

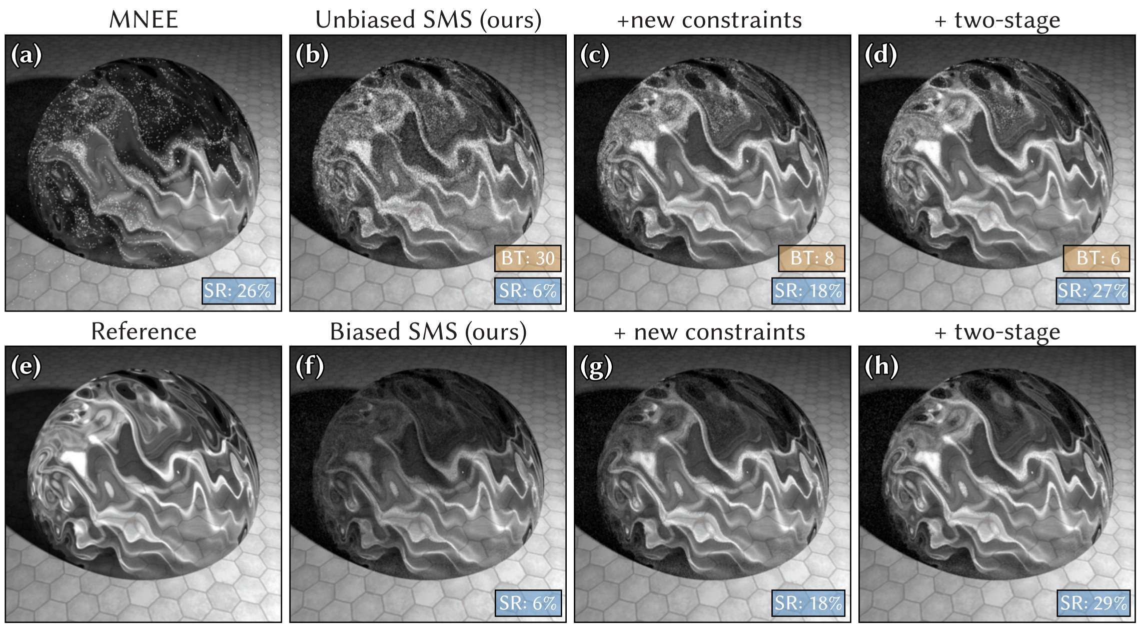


Figure 3.13: Equal time renderings (1 minute) of our two methods and their extensions illustrated on a normal-mapped dielectric sphere illuminated by a small area light. Small insets summarize manifold walk success rate (SR) and average number of Bernoulli trials (BT), where applicable. **(a)** Previous work (MNEE) fails to capture the full complexity of the caustic as it only finds at most one refractive path per shading point. For the remaining energy it falls back to path tracing with high variance. **(b)** Our unbiased SMS method on its own. **(c–d)** Adding the constraint and two-stage manifold walk improvements which increase the success rate while reducing the iteration count. **(f–h)** Same sequence but using our biased SMS variant which suppresses the noise of the reciprocal probability estimation. As the success rate increases from left to right, bias goes down. The remaining bias mainly manifests itself in the regions where the unbiased counterpart remains noisy. The biased version uses a sample set size of $M = 16$. **(e)** A path-traced reference that was rendered for 5 hours.

tributed random numbers $\mathbf{u} = (u_1, u_2) \in [0,1)^2 =: \mathcal{U}$ as input and warps them to the desired distribution. Our objective now is to generate an initial sample $\mathbf{u}$ that is close enough to the solution, so that the Newton iteration (3.11) will take us there.

When inspecting the convergence behavior of these manifold walks on the random number space $\mathcal{U}$, we generally observe multiple *basins of convergence* $\mathcal{B}_k \subseteq \mathcal{U}$, each containing a point $\mathbf{u}^{(k)}$ identified with a corresponding solution vertex $\mathbf{x}_2^{(k)}$. Figure 3.14 illustrates the situation on a simple scene with a cardioid caustic.

Newton's method is known to produce convergence basins that potentially have an extremely complex geometric structure [151] and can even be fractal. For example, Figure 3.15 (a) shows convergence towards three different solutions of a simple polynomial equation on the set of complex numbers and Figure 3.15 (b–c) show a particularly challenging situation we encountered after applying a normal map to the Ring scene.

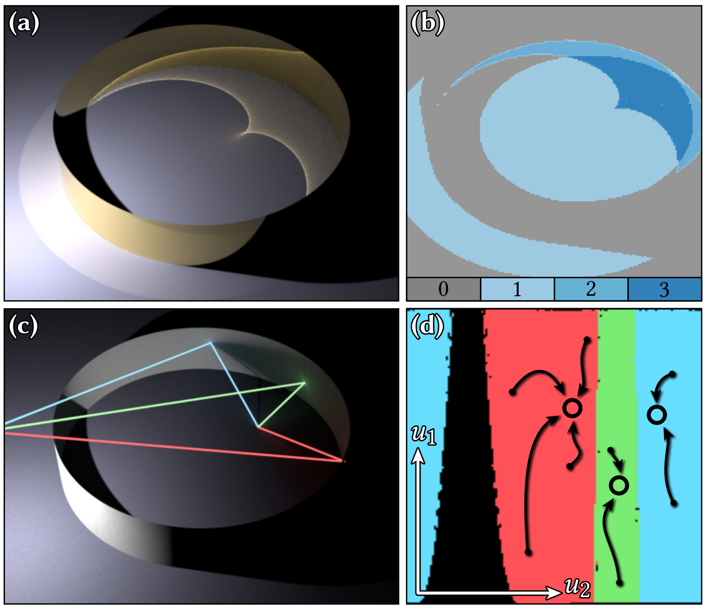


Figure 3.14: **(a)** Multiple solutions of the specular path constraint form a superposition of caustics in the Ring scene. **(b)** Color map showing the number of solutions at each shading point. **(c)** The three solution paths at a particular point. **(d)** The basins of convergence (in random number space) corresponding to those solution paths. All manifold walks started at a point inside a region (e.g. the black dots) converge to the associated solution (colored dots).

Recall that our goal is to use these solutions in a Monte Carlo estimator, where an algorithm that merely finds solutions is insufficient—we must also know the discrete probability P_k of finding a particular path vertex $\mathbf{x}_2^{(k)}$. An unbiased estimator is then given by a standard MC ratio $f(\bar{\mathbf{x}})/p(\bar{\mathbf{x}})$, where $p(\bar{\mathbf{x}})$ in the denominator contains P_k.

Since the samples $\mathbf{u}$ are uniformly distributed, this probability is simply the area of the associated convergence basin on $\mathcal{U}$:

$$P_k = \int_{\mathcal{U}} \mathbb{1}_{\mathcal{B}_k}(\mathbf{u})\,\mathrm{d}\mu(\mathbf{u}). \tag{3.14}$$

However, exact evaluation or precomputation of this integral is clearly infeasible: as discussed, $\mathcal{B}_k$ can have an extremely complex shape which also depends on the position of the path endpoints. On the other hand, a simple unbiased estimator is given by

$$\langle P_k \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}_{\mathcal{B}_k}(\mathbf{u}_i), \tag{3.15}$$

where $\mathbf{u}_i \in \mathcal{U}$ is a sequence of i.i.d. uniform variates. Unfortunately, this approach is flawed: usage of P_k occurs in the *denominator* of the path throughput weight. Since $\mathbb{E}[1/X] \neq 1/\mathbb{E}[X]$, using this estimator could introduce significant bias: for example, $\langle P_k \rangle$

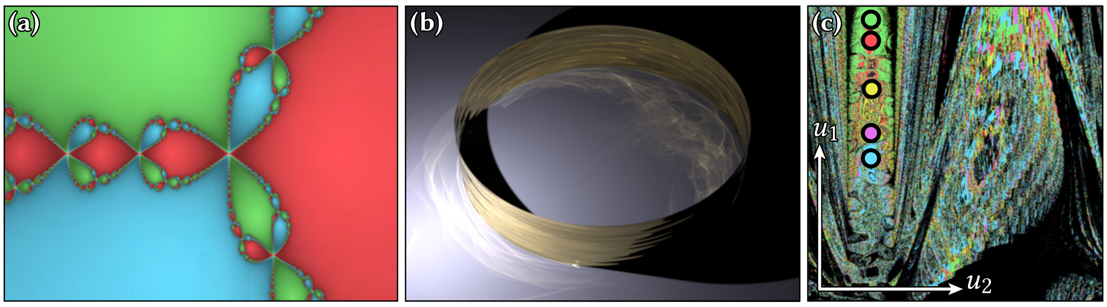


Figure 3.15: **(a)** Newton fractal for $z^3 - 1 = 0$ with $z \in \mathbb{C}$. **(b)** The same RING scene as in Figure 3.14, but with a normal map applied that causes a more high-frequency caustic pattern. **(c)** Complex convergence basins with 5 unique solutions encountered in this scene.

can equal zero if all N tries fail to converge to the basin $\mathcal{B}_k$, in which case the estimated path throughput weight would be infinite! Fortunately, an unbiased estimator for the inverse $\langle 1/P_k \rangle$ can be created using a simple iterative approach.

3.2.2 Unbiased algorithm

The problem of computing an unbiased MC estimate of the reciprocal of an integral was studied by Booth [152]. Recently, Qin et al. [92] built on this idea to create an unbiased photon gathering strategy. The key idea underlying Booth's approach is surprisingly simple: turning the inverse into a geometric series moves the problematic integral from the denominator to the numerator:

$$\frac{1}{P_k} = \frac{1}{\int_{\mathcal{U}} \mathbb{1}_{\mathcal{B}_k}(\mathbf{u})\,\mathrm{d}\mu(\mathbf{u})} = \frac{1}{1-a} = \sum_{i=0}^{\infty} a^i, \tag{3.16}$$

where $a = 1 - \int_{\mathcal{U}} \mathbb{1}_{\mathcal{B}_k}(\mathbf{u})\,\mathrm{d}\mu(\mathbf{u})$. This expansion is legal as long as $|a| < 1$, which in our case—integrating an indicator function over the unit square—is clearly satisfied. Unbiased estimation of the reciprocal then entails repeated manifold walks with i.i.d. initial points, denoted as $\langle a \rangle_j$ below:

$$\langle 1/P_k \rangle = 1 + \sum_{i=1}^{\infty} \prod_{j=1}^{i} \langle a \rangle_j. \tag{3.17}$$

Here, $\langle a \rangle_j = 0$ when manifold walk j has converged to root $\mathbf{u}^{(k)}$ and $\langle a \rangle_j = 1$ if it has found another root or diverged. The above expression can thus be understood as a simple counting process: we run repeated manifold walks *until* $\mathbf{u}^{(k)}$ is found, and the number of trials then provides an unbiased estimate of $1/P_k$. This result can also be understood in terms of the geometric distribution, which models the number of Bernoulli trials needed

```
 1  # Estimate radiance travelling from emitter position $\mathbf{x}_3$ to shading point $\mathbf{x}_1$.
 2  def sms_unbiased($\mathbf{x}_1, \mathbf{x}_3$):
 3      # Sample a specular vertex as initial position
 4      $\mathbf{x}_2$ = sample_seed_position(...)
 5      # Run Newton solver to find valid connection
 6      $\mathbf{x}_2^*$ = manifold_walk($\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$)
 7      # Estimate inverse probability of sampling $\mathbf{x}_2^*$
 8      $\langle 1/P_k \rangle = 1$
 9      while True:
10          # Sample vertex as above
11          $\mathbf{x}_2$ = sample_seed_position(...)
12          $\mathbf{x}_2'$ = manifold_walk($\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$)
13          if $\|\mathbf{x}_2' - \mathbf{x}_2^*\| < \varepsilon$:
14              break
15          $\langle 1/P_k \rangle = \langle 1/P_k \rangle + 1$
16      return $f_s(\mathbf{x}_2^*) \cdot G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2 \leftrightarrow \mathbf{x}_3) \cdot \langle 1/P_k \rangle \cdot L_e(\mathbf{x}_3) \,/\, p(\mathbf{x}_3)$
```

Algorithm 3.2: Unbiased specular manifold sampling.

until a certain event with probability P_k takes place. Here again, the expected number of attempts is $1/P_k$. Simulating a geometric distribution therefore provides an unbiased estimator of the sought reciprocal and constitutes the base ingredient of our unbiased SMS scheme, which we lay out in Algorithm 3.2.

Unbiased SMS is trivially added to any existing implementation of MNEE. Its runtime cost is directly linked to the "complexity" of specular paths in the scene: when the geometry is relatively smooth, $\mathcal{U}$ contains a small number of solutions that are surrounded by large convergence basins. Lines 3–6 then rapidly converge to a particular solution, and only a few iterations of lines 9–15 are required to find that same solution once more. However, when the geometry is complex, many solutions may exist, and their convergence basins also occupy smaller area in random number space. The required number of trial iterations in lines 9–15 is a potential cause for concern in this case. Furthermore, the variance of such an estimator for a specific solution P_k based on a geometric distribution is equal to $(1 - P_k) \,/\, P_k^2$, which can become very large when $P_k \approx 0$.

Another fundamental issue with unbiased SMS is that a considerable amount of computation is in some sense not used optimally: repeated Bernoulli trials potentially find many additional solutions, yet Algorithm 3.2 only cares about one bit of information: whether or not these match the solution that was previously found in line 6. It would be desirable that the method leverages this additional information to improve the esti-

```
# Biased estimate of radiance travelling from emitter position $\mathbf{x}_3$ to
# shading point $\mathbf{x}_1$ based on trial set size $M$.
def sms_biased($\mathbf{x}_1, \mathbf{x}_3, M$):
    # Keep a set of unique solutions
    $S = \{\}$
    for $i = 1, \ldots, M$:
        $\mathbf{x}_2$ = sample_seed_position(...)
        $\mathbf{x}_2^*$ = manifold_walk($\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$)
        $S = S \cup \{\mathbf{x}_2^*\}$
    # Accumulate contribution
    result = 0
    for $l = 1, \ldots, \text{size}(S)$:
        result += $f_s(\mathbf{x}_2^{(l)}) \cdot G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2^{(l)} \leftrightarrow \mathbf{x}_3) \cdot L_e(\mathbf{x}_3) \,/\, p(\mathbf{x}_3)$
    return result
```

Algorithm 3.3: Biased specular manifold sampling.

mates, but it is challenging to do without introducing bias. That said, the presence of some bias is often acceptable if this improves other aspects, such as running time or variance, without introducing undesirable visual artifacts in renderings. Motivated by this, we devise a biased variant of SMS that addresses the discussed concerns.

3.2.3 Biased algorithm

The biased variant of our method (Algorithm 3.3) replaces the unbounded number of trial iterations by a fixed budget of M samples. Furthermore, instead of sampling one path, and then performing Bernoulli trials to estimate its reciprocal probability, we simply cluster the M samples into a set of unique solutions $\mathbf{x}_2^{(l)}$ ($l = 1, \ldots, L$). A biased estimate of the reciprocal is then given by the relative number of occurrences n_l of each solution, which also avoids the potential issues with a division by zero discussed earlier. The (biased) estimator for the total throughput at shading point $\mathbf{x}_1$ then becomes:

$$\frac{1}{M}\sum_{l=1}^{L} n_l \frac{f(\mathbf{x}_2^{(l)})}{p(\mathbf{x}_2^{(l)})} \approx \frac{1}{M}\sum_{l=1}^{L} n_l\, f(\mathbf{x}_2^{(l)})\,\frac{M}{n_l} = \sum_{l=1}^{L} f(\mathbf{x}_2^{(l)}). \tag{3.18}$$

Compared to the original unbiased approach, this variant has a fixed iteration count, and it exploits the information provided by all samples. Note that it is *consistent* and will converge to the true solution as $M \longrightarrow \infty$. In our experiments, we observe that the resulting bias is manifested as energy loss: regions of complex caustics that are only rarely found by the Newton iteration appear darker, while the unbiased variant produces the correct intensity at the cost of substantially increased variance and runtime. We believe that trading variance for energy loss in this way could be preferable in applied contexts (e.g. visual effects). Section 3.4 presents a number of results contrasting the two methods.

When setting $M = 1$, biased SMS is closely related to a biased version of MNEE where no MIS is applied to account for paths that cannot be sampled using the manifold walk. The two algorithms are however not equivalent. Consider a situation where only one valid solution exists but the straight-line initialization does not converge to it. In this case, MNEE can produce arbitrarily high variance—or in case of point lights will miss the contribution entirely. SMS however will still find the solution with non-zero probability.

Another interesting aspect of the biased variant is that it generates many samples at once. Coherence in this computation could be amenable to vectorized execution using modern SIMD instruction sets, such as AVX512.

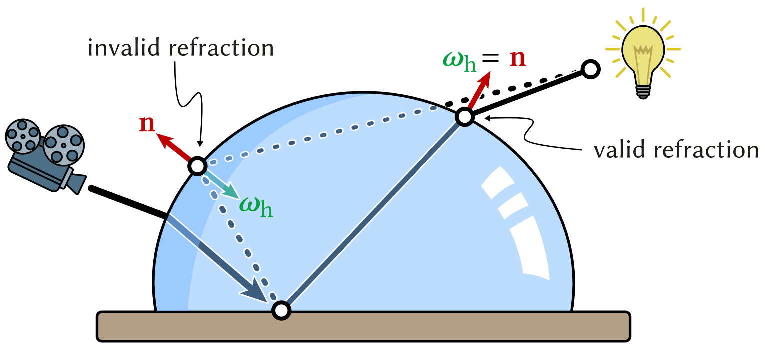


Figure 3.16: The half-vector constraint of Jakob and Marschner [116] often produces invalid back-facing solutions where only the *projected* half-vector and surface normal are equal.

3.3 Extensions

We now turn to the discussion of various extensions to our method, that either improve convergence (as previewed in Figure 3.13) or generalize the technique to rendering of glints.

3.3.1 Improved specular constraints

One significant difference of our method compared to all previous applications of manifold walks is that we require the Newton solver to take very large steps starting from an invalid state. In contrast, MNEE renders refractive caustics with a straight-line initialization that is generally already very close to the final solution. Applications of manifold walks to MCMC rendering [116, 153] only make small perturbations to existing valid paths, in which case the Newton iteration converges rapidly. During the development of our technique, we found that manifold walks would often converge surprisingly poorly when initialized randomly, which led to serious convergence issues even in the case of simple and smooth geometry (Figure 3.17). We realized that these two aspects are related: when taking large steps, Newton iterations based on the original specular manifold constraints often produce invalid back-facing solutions that impede convergence.

The main issue here is how the specular manifold constraint in Equation (3.2) encodes specular configurations via half-vector projections. While this term conveniently subsumes both reflective and refractive cases with one equation, the formulation does not distinguish between front- and back-facing solutions (Figure 3.16).

We propose a new constraint function that removes this ambiguity. After choosing between reflection and transmission, note that the law of reflection and refraction fully determine the scattered direction $\mathrm{S}(\boldsymbol{\omega}, \mathbf{n}, \eta)$. Its definition is stated previously in Equa-

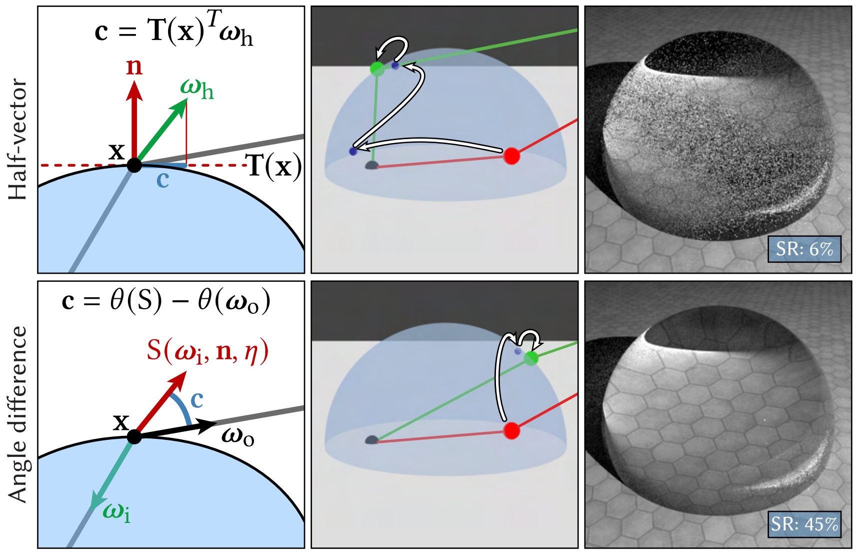


Figure 3.17: Comparison between the previous half-vector based constraint function (**top**) and our new one based on angle differences (**bottom**). **Left**: 2D illustrations of the constraints. **Center**: For a random initial point (red) on the sphere, the half-vector constraint converges to an invalid configuration after three iterations. Our improved formulation results in better global convergence of the Newton solver. **Right**: Corresponding equal-time renderings on a dielectric sphere lit by a small area light source. Manifold walks are started at uniformly sampled positions over the glass surface. The walk success rate (SR) in the insets is measured over the entire image, where pixels outside the caustic count as sampling failures.

tion (2.21) and depends on the incident direction $\boldsymbol{\omega}$, the surface normal $\mathbf{n}$, and the relative index of refraction η. When a vertex is in a valid physical configuration, its incident and outgoing directions satisfy $\boldsymbol{\omega}_\mathrm{o} = \mathrm{S}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta)$. Another way of encoding specular manifold constraints thus entails measuring the difference between $\boldsymbol{\omega}_\mathrm{o}$ and $\mathrm{S}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta)$.

We experimented with different implementations of such a distance function, including simple 1D angle measurements $\angle(\boldsymbol{\omega}_\mathrm{o}, \mathrm{S}(\boldsymbol{\omega}_\mathrm{i}, \mathbf{n}, \eta))$ that produce a non-square Jacobian J_C requiring the use of a pseudoinverse in the Newton iterations. Ultimately, we found that 2D constraints $\mathbf{c}_i : \mathbb{R}^2 \to \mathbb{R}^2$ as in prior work exhibit better behavior, and we therefore measure a difference in the spherical coordinates of both vectors:

$$\mathbf{c}_i = \begin{pmatrix} \theta(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}) - \theta(\mathrm{S}(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \mathbf{n}_i, \eta_i)) \\ \phi(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}) - \phi(\mathrm{S}(\overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \mathbf{n}_i, \eta_i)) \end{pmatrix} \tag{3.19}$$

where $\theta(\boldsymbol{\omega}) = \arccos(\omega_z)$ and $\phi(\boldsymbol{\omega}) = \mathrm{arctan2}(\omega_y, \omega_x)$ determine associated spherical coordinates. Figure 3.17 showcases the significantly improved convergence due to these constraints. See also Appendix A for a derivation of the necessary constraint derivatives.

There are two subtle details that need to be considered when implementing Equation (3.19):

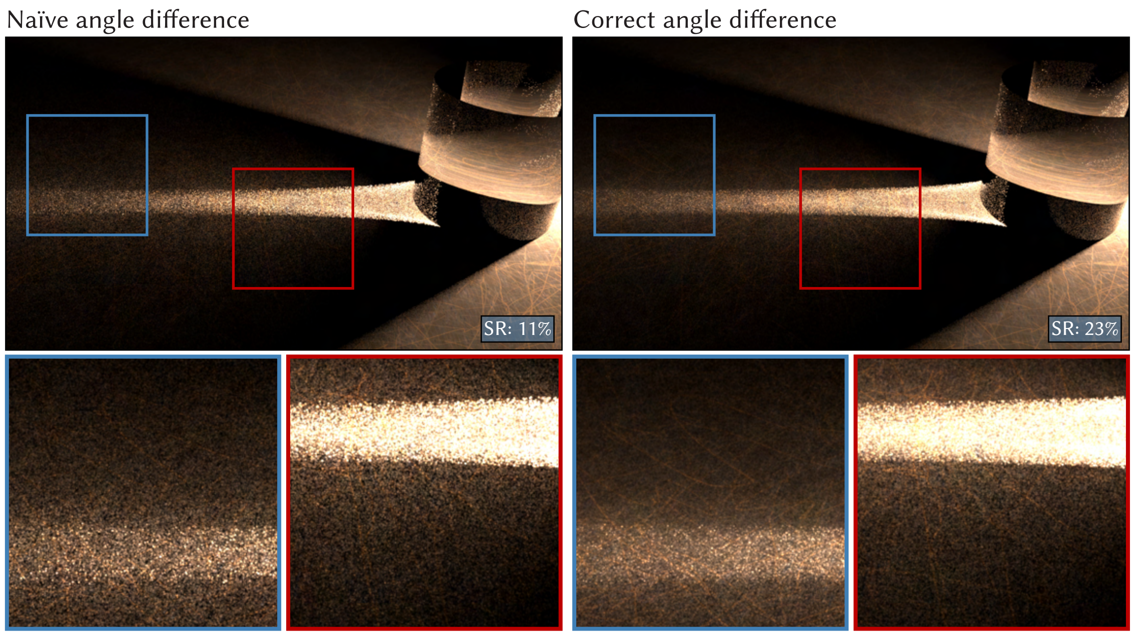


Figure 3.18: Equal time comparison on a caustic from a two-bounce refraction through a dielectric cylinder. We compare naïve (**left**) and correct (**right**) handling of the azimuth angle difference in the specular constraint from Equation (3.19) which causes the manifold walk success rate (SR) to roughly double. The brightness of the two insets below is scaled in order to make the noise reduction outside of the main caustic more visible.

1. The scattering operation $S(\cdot)$ can fail in configurations with total internal reflection. In such cases we are still able to evaluate the constraint for the opposite ray direction by taking the difference between $S(\boldsymbol{\omega}_o, \mathbf{n}, \eta^{-1})$ and $\boldsymbol{\omega}_i$.

2. The subtraction of azimuth angles requires special handling due to their periodicity. For instance, the Newton solver does not consider a constraint value, i.e. angle difference, of roughly 2π to be close to the root and might therefore not converge. Instead, the result must be mapped onto $[-\pi, \pi]$ using a floating point modulo operation.

   While the naïve angle difference implementation only affects the convergence rate of the Newton solver and not the correctness of the method, this is of course detrimental for the overall sampling efficiency. Figure 3.18 demonstrates an example of a two-bounce refraction where this small change roughly doubles the solver convergence rate, and thus significantly reduces variance.

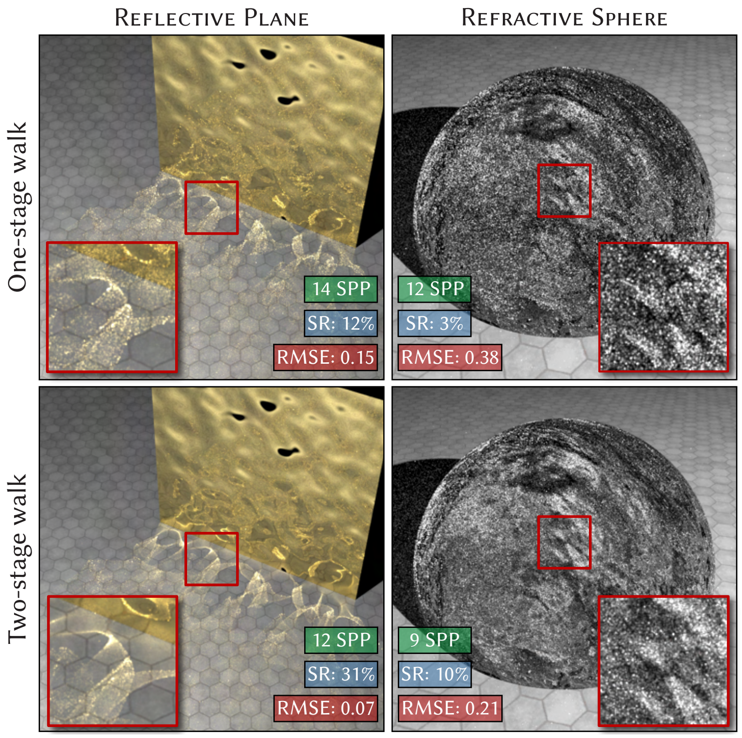


Figure 3.19: Two scenes showing specular reflection and refraction on normal-mapped surfaces, rendered with unbiased SMS. We show equal-time comparison (1 minute) between starting points distributed uniformly over the shapes (**top**) and our two-pass manifold walks with improved starting points (**bottom**). Insets show computed number of samples per pixel (SPP), root mean square error (RMSE) compared to a converged reference, and manifold walk success rate (SR) measured over all pixels. Note how the cost of the additional manifold walk is amortized by a faster probability estimate resulting in a similar number of computed samples between the two methods.

3.3.2 Two-stage manifold walks

Comparing the nature of specular paths in a simple cardioid caustic (Figure 3.14) to a reflection from normal-mapped geometry (Figure 3.19, left column), we observe that the more complex caustic is a superposition of many different solutions with a fairly localized effect. Figure 3.20 (a) visualizes the corresponding space $\mathcal{U}$ that is largely empty, and only contains a few small convergence basins that are clustered together. The probability of finding a valid solution is therefore low. Furthermore, estimating the reciprocal probability depends on our ability to *find the same solution once more*, which is even less likely. Unbiased SMS estimates therefore tend to be slow and noisy, while biased SMS loses a considerable amount of energy.

To address these problems, we will exploit prior knowledge to devise an improved strategy for generating initial guesses. To motivate our approach, consider a caustic

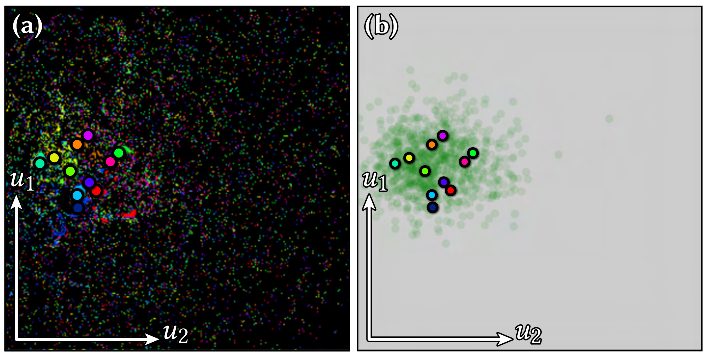


Figure 3.20: **(a)** Convergence basins of the Reflective Plane scene in Figure 3.19 (a), for one shading point on the ground inside the caustic region. **(b)** Sampling density (green) of intermediate points after the first manifold walk in our two-pass approach, which focuses on the cluster of solutions.

generated by a smooth planar surface (e.g. a flat version of the metal surface shown in Figure 3.19). Points on this caustic all correspond to a single solution with a large convergence basin in $\mathcal{U}$. If we begin to make the surface more complex, e.g., by perturbing positions or shading normals, this single solution splits into multiple nearby solutions with disproportionately smaller convergence basins. Stronger geometric perturbations tend to produce more solutions that are spread further apart. The aforementioned issues with SMS could be addressed if there was a way to predict the locations of these solutions and construct starting points in their proximity in a more targeted manner. We propose a simple technique to create such a targeted initial guess for the special case of normal-mapped surfaces[5].

Our *two-stage* sampling approach performs two manifold walks: the first stage ignores the specified normal map and finds specular paths on the original smooth surface. Such a smooth manifold walk will converge to a point that roughly lies at the center of the cluster of solutions in Figure 3.20 (a). However, instead of an ordinary manifold walk, our first stage relies on an *offset manifold walk* from Section 3.1.4, whose manifold constraint tilts the normal of the underlying surface. The normal perturbation is randomly chosen from the distribution of normals that are present in the normal map. Cheaper approximations are also usable—we use a Gaussian approximation of the entire normal map obtained from the lowest MIP level of a LEAN map [154]. Importantly, this offset normal is chosen before each manifold walk and does not change during the iteration which converges rapidly and with high probability (i.e. the convergence basin is large). This initial randomized manifold walk brings us into the proximity of the various solu-

[5]The high-level idea is more general and can likely be extended to other approaches for introducing surface detail, such as displacement maps.

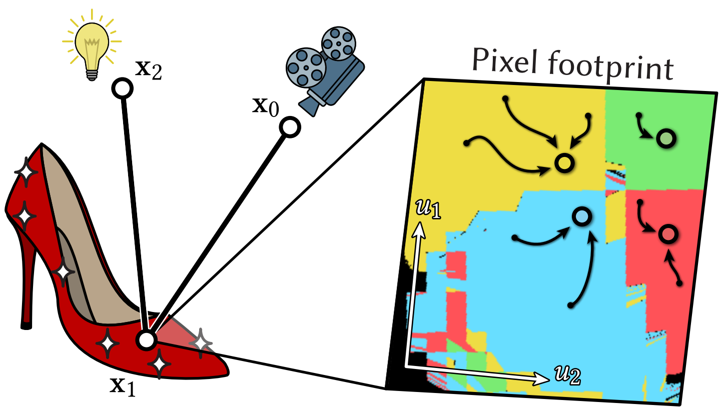


Figure 3.21: Specular manifold sampling can also find connections within a pixel footprint to render glints that arise due to specular microstructure.

tions (green density in Figure 3.20 (b)); a second manifold walk, on the bumpy surface, started from there takes us all the way to a solution.

Estimation of the reciprocal probability of this adapted sampling strategy is surprisingly easy: the only requirement is that two-stage sampling is consistently used in both Lines 3–6 and Lines 9–15 of Algorithm 3.2. It would generally appear that the two-stage sampling should be more costly, but in our experiments (e.g. Figure 3.19) we observed roughly equal performance. The reason for this is that the reciprocal probability estimator requires fewer iterations to rediscover a solution. At the same time, variance is reduced noticeably.

3.3.3 Glints

Our method generalizes straightforwardly to the problem of rendering *glints*, which are minuscule subpixel reflections of a light source with narrow angular support (e.g. the sun) on high-frequency specular microgeometry.

Rendering glints using standard Monte Carlo techniques tends to be prohibitively expensive, since millions of samples per pixel may be needed to obtain an acceptable result.

Interestingly, the problem of finding glints is almost identical to the caustic case, the main exception being that the search is now constrained to small surface regions observed by individual pixels and their reconstruction filters. Previous general glint rendering techniques [136, 137] evaluate an effective integrated BRDF over the entire pixel footprint and produce very high quality estimates at the cost of burdensome 4D spatio-directional hierarchies that are necessary to organize and query the distribution of

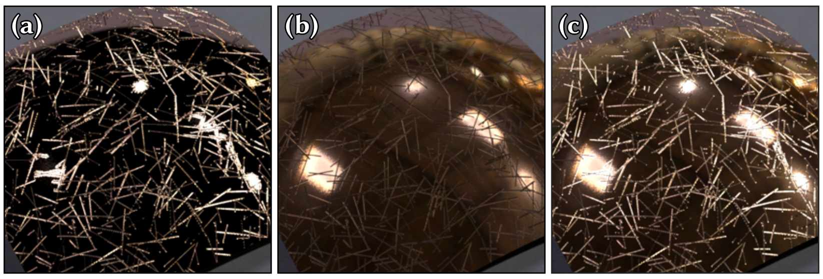


Figure 3.22: Curved metal surface with small scratches lit by a high-frequency environment map containing multiple light sources. Weighting using MIS effectively uses each technique where it performs best: **(a)** Our method (unbiased SMS) captures the direct illumination from the bright light sources inside the scratches. **(b)** Standard BSDF sampling focuses on reflections from flat regions. **(c)** Combined result.

normals. Our goal is to implement an unbiased estimate of such a query using manifold walks.

Our method generates random starting points within the pixel and then runs unbiased or biased SMS to find solutions (Figure 3.21). In contrast to prior work, which introduced a small amount of *intrinsic roughness* to relax the problem, we solve the unmodified problem with a discrete set of solutions to find a path $\bar{\mathbf{x}} = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2$ connecting the camera to a sampled emitter position via a single[6] specular reflection. After tracing the initial camera ray, we approximate the projected pixel footprint with a small parallelogram based on ray differentials [126]. We perform local manifold walks in UV space to refine the initial guess. This leads to much higher performance compared to the caustic case as no costly ray tracing operation is needed to re-project onto the surface. We terminate paths that step outside the parallelogram since they are unlikely to find usable solutions. We assume that the endpoints $\mathbf{x}_0$ and $\mathbf{x}_2$ are distant, in which case changes in the half-vector across the parallelogram are minimal and it can thus be approximated by a constant vector. This *far-field approximation* is shared by previous work, and technically biases all these techniques[7]. At the same time, this considerably simplifies the search for solutions: in particular, the specular manifold constraint can be simplified to a function that attempts to equate this fixed half-vector and the local shading normal, both expressed in slope space:

$$\mathbf{c} = \tilde{\omega}_{\mathrm{h}} - \tilde{\mathbf{n}}. \tag{3.20}$$

Here, the conversion between 3D vectors $\mathbf{v} = \left(v_x, v_y, v_z\right)^T$ and their corresponding 2D

[6]Glints involving multiple specular reflections on displaced geometry could in principle also be found using manifold walks, which could be an interesting topic for future work.

[7]We already consider the use of a normal map instead of an actual surface to be an approximation to some degree, and are not too concerned with this additional detail.

slope $\tilde{\mathbf{v}} = \left(\tilde{v}_x, \tilde{v}_y\right)^T$ is [154]:

$$\tilde{\mathbf{v}} = \begin{pmatrix} -v_x/v_z \\ -v_y/v_z \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \frac{1}{\sqrt{\tilde{v}_x^2 + \tilde{v}_y^2 + 1}} \begin{pmatrix} -\tilde{v}_x \\ -\tilde{v}_y \\ 1 \end{pmatrix}. \tag{3.21}$$

Like previous work, we use high-resolution normal maps to encode the subpixel surface details in our scenes. In principle, the method could be extended to other types of normal variation or actual geometric displacement.

To robustly apply our glint rendering technique in scenes with complex lighting (e.g. high-frequency environment maps), we further combine our SMS strategy with standard BSDF sampling using MIS (Section 2.4.4). We found it very effective to use approximate MIS weights based on the directional distributions from the light source and the effective BSDF of the pixel footprint provided by a LEAN map [154]. As shown in Figure 3.22 this approach successfully separates the regions of the integrands where one sampling strategy is preferable over the other. While precomputing a LEAN map adds some storage overhead, it is orders of magnitude lower than Yan et al.'s spatio-directional trees [137]. We also note that our method always renders the non-smoothed input normal map without geometric approximations—the LEAN map is only used as a proxy to compute an effective BSDF sampling density over a pixel footprint, enabling unbiased combination of our sampling strategy with other standard techniques via MIS.

Our method can become impractical when the pixel footprint under consideration covers a substantial region in texture space, e.g., when the camera observes the surface from far away or the surface consists of extremely small detail. In that case, the large number of solutions within the footprint that we need to stochastically enumerate becomes troublesome: the unbiased probability estimate will have high runtime and variance, whereas the biased variant will usually suffer from high energy loss. Future improvements to this, e.g. in the form of some prefiltering method [138, 154], would be desirable.

Nevertheless, our SMS glint rendering approach shows competitive performance against prior work in common scenarios with moderately sized footprints and vast improvements compared to a brute force approach—all while keeping memory requirements low. Example results and comparisons are shown later in Section 3.4.8.

3.4 Results

We now present several results rendered with our method as well as comparisons to relevant prior work. We based our implementation on the Mitsuba 2 renderer [4]. All time measurements were recorded on a compute node with 2 Xeon 6132 processors, each with 14 2.6 GHz cores. Many of the rendered results contain small insets with the manifold walk success rate (SR) for the corresponding image. These numbers may seem low at first, but are averaged over the full image—including pixels that do not contain any caustics for SMS to find.

3.4.1 Integration into rendering algorithms

Specular manifold sampling is a general building block in the design of rendering algorithms, and there is considerable flexibility in its usage. We experimented with different ways of incorporating it into a unidirectional path tracer and ultimately opted for uniform seed-point generation on specially marked "caustic caster" shapes. For two-bounce caustics, shown in Section 3.4.7, the seed point on the second interface is found by tracing a ray through the first vertex and performing either reflection or refraction depending on the type of surface. SMS is very effective for sampling high-frequency caustic paths from (near) specular surfaces and often greatly outperforms standard emitter or BSDF sampling strategies. However, in the case of significant surface roughness or smooth illumination, standard strategies remain superior. Ideally, all of these strategies would be combined via MIS to improve their robustness, as we do in Section 3.3.3 for the special case of glints. However, determination of suitable probabilities in the more general caustic setting remains an open problem. SMS iterations are also relatively expensive, and we use them even at shading points where no possible connections exist. Conservative criteria that specify when SMS should be used are another interesting avenue for future work.

3.4.2 Shop window scene

Figure 3.1 showcases our method in an example involving glints and caustics on complex geometry. It features several different caustics rendered using unbiased SMS: reflective caustics caused by a spotlight shining on the gold pedestals, and a two-bounce refractive caustic from a point light hidden inside the middle pedestal. The shoes have a glittery appearance that is rendered using biased SMS for glints. We disable caustics from higher

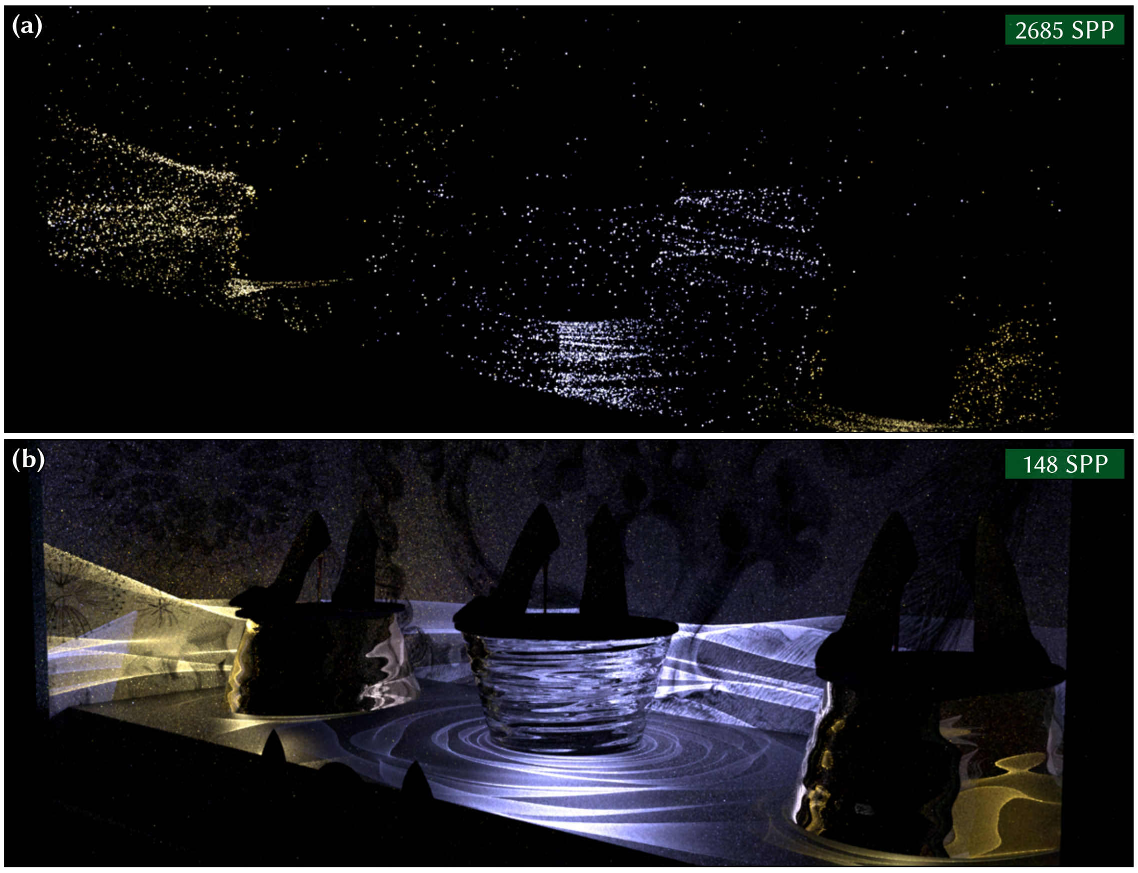


Figure 3.23: Modified version of the Shop Window scene from Figure 3.1 where a small amount of surface roughness is added to the previously purely specular surfaces, enabling a standard path tracer **(a)** to find the same light transport paths as our proposed method **(b)**. Both renderings are computed in equal time (20 minutes), showing only contributions due to caustics.

order scattering (e.g. two or more bounces between the pedestals). Many of the caustics in this scene are SDS paths that a regular path tracer cannot find. We can still compare to a path tracer by adding some surface roughness (Beckmann NDF with $\alpha = 0.005$) to all specular shapes. Figure 3.23 shows such a comparison, where we only visualize the contributions sampled by SMS. Given equal render time, our method outperforms regular path tracing by a significant margin.

The scene also illustrates a current limitation of our method: even though SMS can effectively sample light connections through specular interfaces, there is still significant variance in the scene coming from other challenging paths, mainly the indirect lighting caused by the caustics; see Figure 3.23 (b) and the remaining noise in Figure 3.1. Our method cannot sample these paths explicitly; the issue could be addressed by path guiding methods.

3.4.3 Specular manifold sampling

In Figures 3.24 and 3.25 we compare the effectiveness of SMS to brute-force path tracing and the previous state-of-the-art method MNEE of Hanika et al. [132]. We examine three scenes with challenging single-bounce caustics due to normal-mapped surfaces. The Swimming Pool and Refractive caustic scenes are a famous examples where both uni- and bidirectional path tracing techniques fail to discover the prominent SDS paths that generate intricate patterns on the ground plane. MNEE improves on this but misses all but one light connection. To stay unbiased, it has to fall back to the brute-force strategy for the remaining paths which still causes significant variance. Alternatively, a biased version of MNEE can simply omit such light connections at the cost of severe energy loss. Our method finds all paths and significantly outperforms the others in equal time.

The Reflective Plane scene is an example where MNEE was previously not applicable. For a clearer comparison, we added a variation (called "MNEE" in quotes) that constructs a deterministic seed path by tracing a ray from the shading point towards the center of the object's bounding box. Since the caustic is the superposition of many individual solutions, this is clearly not sufficient and "MNEE" ends up finding only a very small part of the caustic. Biased versions of SMS can optionally be applied and reduce high-frequency noise caused by the unbiased probability estimate. This comes at a higher cost per individual sample and some energy loss, the extent of which can be controlled by the trial set size M. We want to highlight that the biased method is free of artifacts and generates temporally coherent results despite its limited exploration of a random subset of the path space.

3.4.4 Biased SMS

Our biased SMS approach involves an interesting tradeoff between sample variance and energy loss. The trial set size parameter M plays an important role here: it directly relates to how much effort the sampler spends per radiance estimate from caustics, and indirectly controls how much of the caustic is found. We explore this effect in Figure 3.26 on the same scenes as used above. Choosing the optimal M is currently a user choice and could be investigated more in the future.

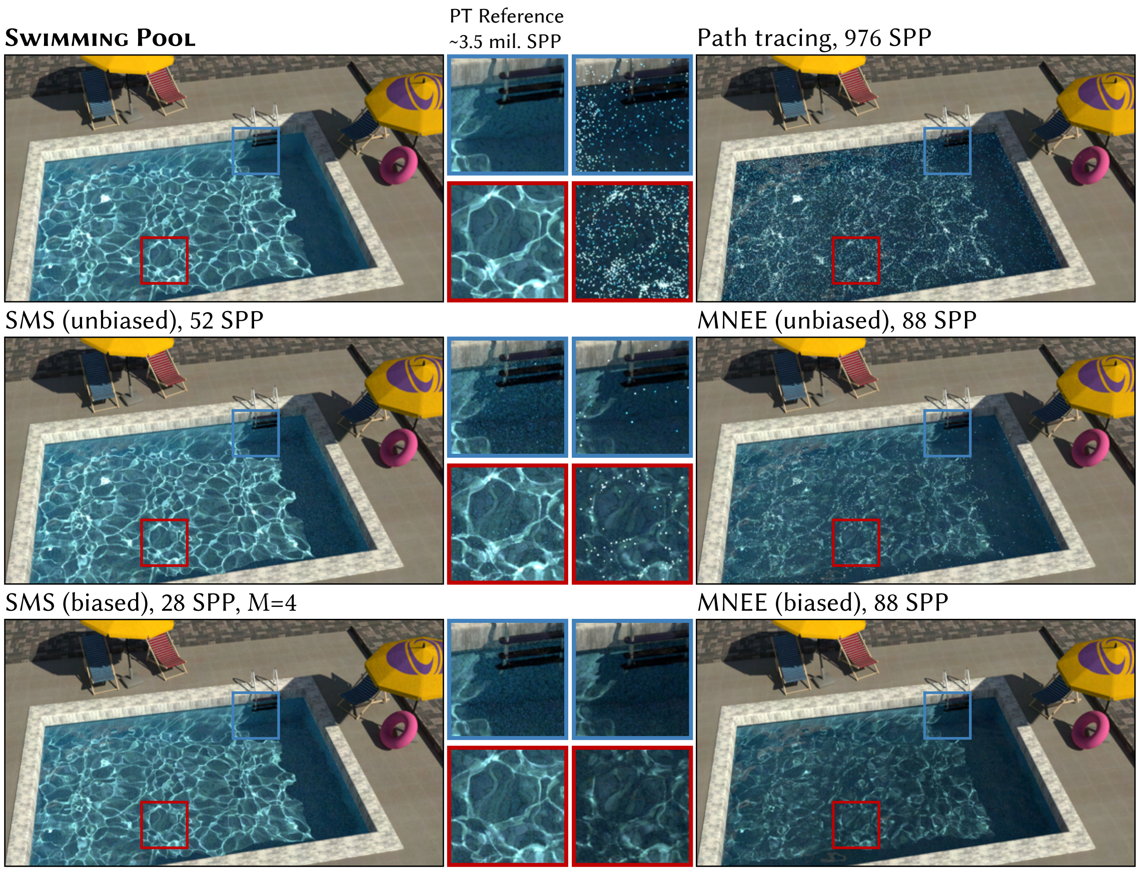


Figure 3.24: Equal-time comparison (5 minutes) between path tracing, prior-work MNEE [132], and both unbiased and biased versions of our proposed SMS method. Previous work can only produce (at most) one connection to the light source via the specular interface, whereas our techniques can sample the full range of light paths either in an unbiased or biased way. The latter removes some of the high-frequency noise introduced by the unbiased probability estimate and trades it for energy loss based on the trial-set size parameter M. We report samples per pixel (SPP) computed by each method, as well as the chosen M in the biased case. Due to the challenging light transport, the converged image (top left) is rendered using (unbiased) SMS and only the corresponding insets are path traced references with an extremely high sample count.

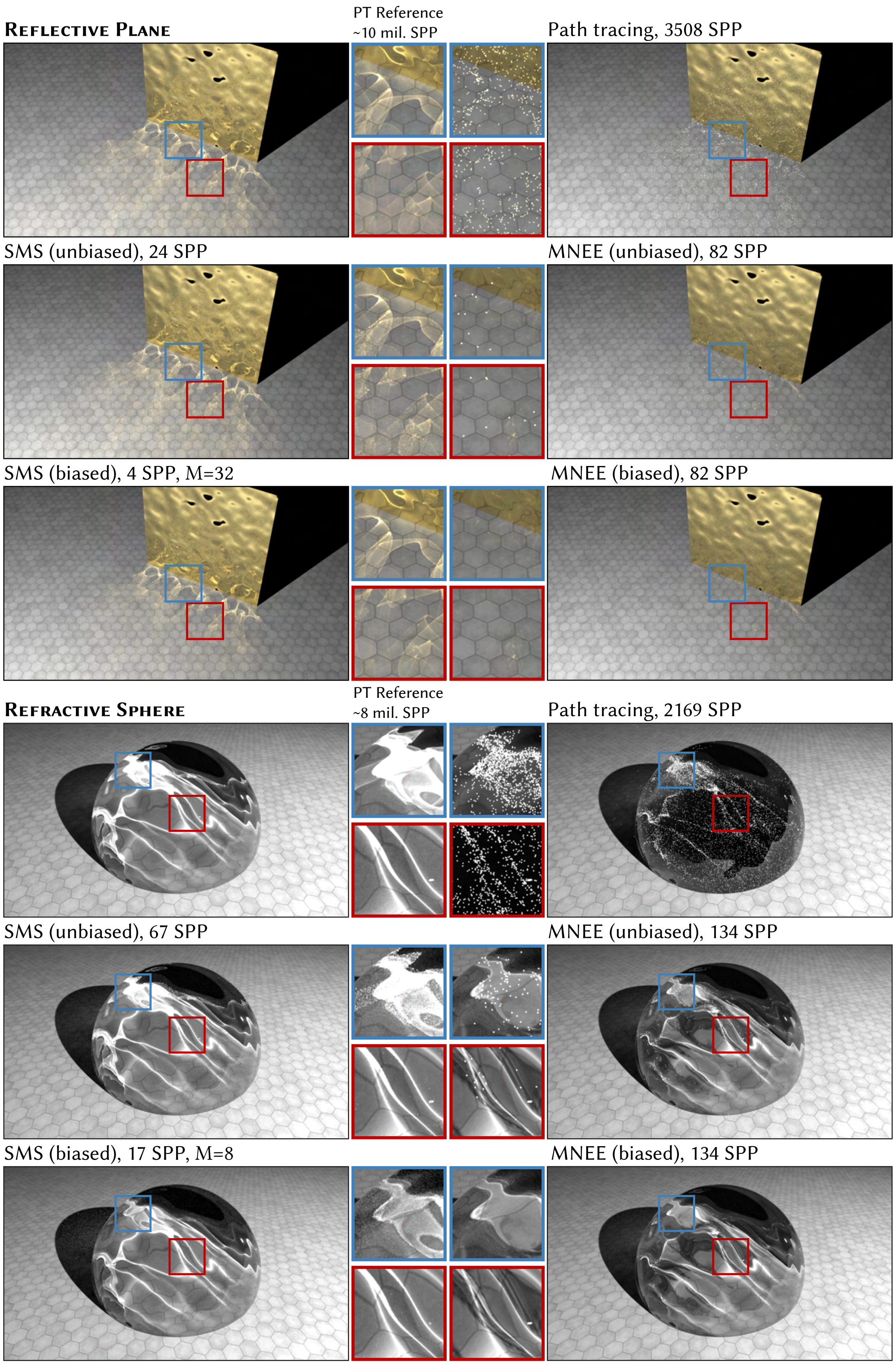


Figure 3.25: Same equal time comparison as in Figure 3.24 but on two additional test scenes.

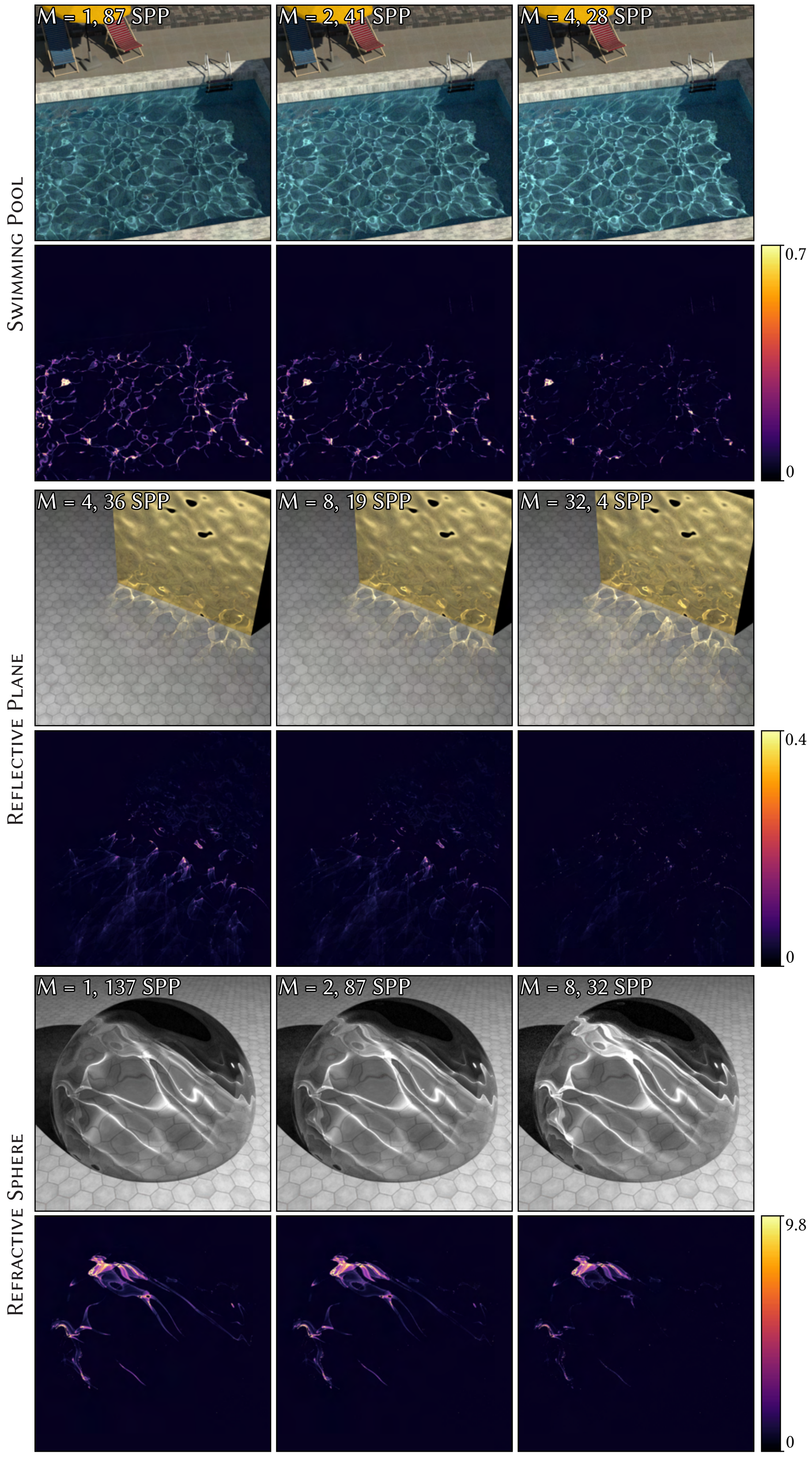


Figure 3.26: Comparing our biased method with varying trial set sizes M, at equal render time of 5 minutes. As M increases, each individual radiance estimate becomes more expensive but less energy of the caustics is lost. Overlays show pixel-wise squared error compared to unbiased SMS.

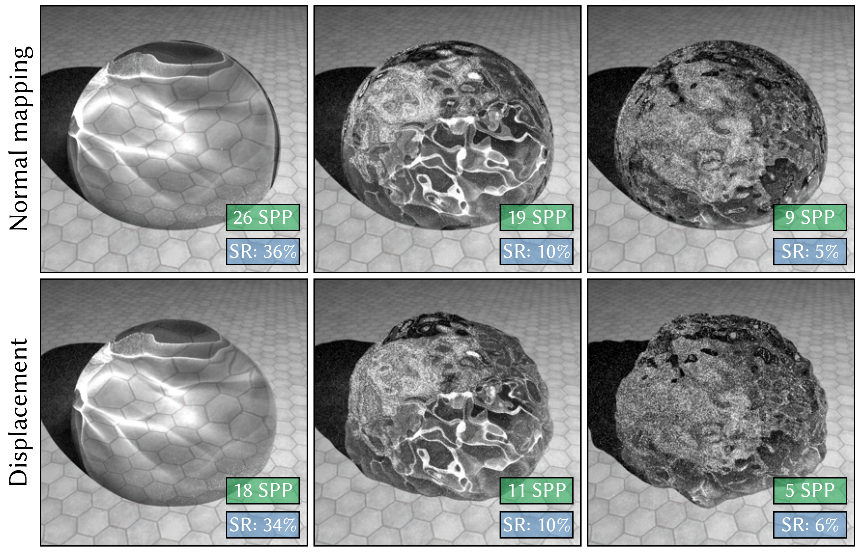


Figure 3.27: **Top**: Sequence of refractive spheres with increasingly complex normal maps. **Bottom**: The same setup, but this time with actual displaced geometry. Insets show computed number of samples per pixel (SPP) and the manifold walk success rate (SR) measured over all pixels in the image. The manifold walks perform about equally on both geometry types.

3.4.5 Geometric displacement

Although many of the shown results involve caustics from specular surfaces with normal maps, our method is more general, and we also found it to be effective on surfaces with true geometric detail (e.g. from a displacement map), which we illustrate in Figure 3.27. When not using our two-stage sampling method, which is currently limited to normal maps, manifold sampling performs equally well on both types of surfaces as confirmed by the similar success rates. This seems counter-intuitive at first, since smoother geometry should also result in a specular manifold that is easier to navigate. However, the disagreement between actual geometry and the “fake” surface variation from normal maps can also limit the Newton solver’s efficiency. The ray-tracing-heavy nature of the manifold walks and the underlying re-projection steps lead to slightly reduced performance in scenes with dense geometric tessellation.

3.4.6 Surface roughness

Like prior work on specular manifolds, our method generalizes to near-specular or rough surfaces by performing offset manifold walks previously described in Section 3.3.2. Here, we sample an *offset normal* from the material’s microfacet distribution before the SMS step. The manifold walk then searches for specular connections involving that offset

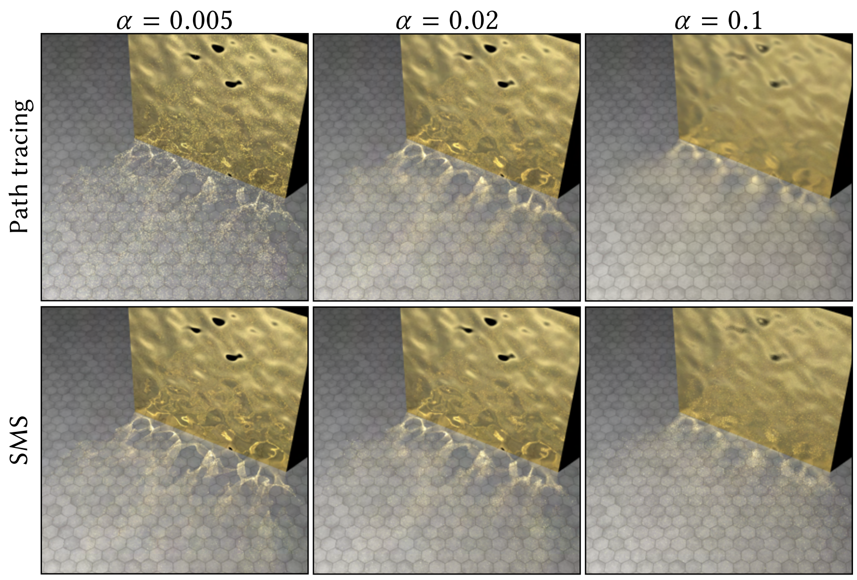


Figure 3.28: Rough reflective plane with a rough metallic surface with Beckmann normal distribution. As the roughness α increases from left to right, the path tracer becomes more capable of performing the connection to the light source on specular plane directly, whereas our proposed sampling strategy loses its efficiency. Both methods use equal render time of 5 minutes.

normal instead of the true shading normal. Only a finite number of solutions exists for this specific offset normal, and the probability estimate of therefore remains unchanged. Averaging over paths sampled in this manner converges to the correct solution [132].

Figure 3.28 shows a sequence of increasingly rough reflective surfaces. Note how the caustic is at first sharp and full of high-frequency details, and becomes progressively blurry from left to right. This blur also enables a unidirectional path tracer to find valid light connections more often: in the limit case, every point on the surface is contributing to the shading point. As our method handles roughness by integrating over many perfectly specular light paths with randomized offset normals, the opposite is true for our method and its variance increases. We only recommend using our technique for specular or near-specular surfaces and switching over to conventional path sampling techniques in other cases. In the future, it would be interesting to incorporate a form of multiple importance sampling to robustly handle both extremes in addition to intermediate cases. Note that the same argument applies also to scenes with low-frequency lighting, e.g. largely constant environment maps. There, standard BSDF sampling techniques become viable options of randomly intersecting the light sources.

3.4.7 Multiple specular interactions

Like MNEE, the principle behind our method generalizes to longer chains with multiple specular interactions. In Figure 3.29, we show an intricate caustic pattern caused by double refraction through a solid displaced piece of glass. We again compare our unbiased and biased SMS variants to a standard path tracer and MNEE. Multiple interactions increase the dimension of the space of initial configurations that SMS must generate: we could generate initial rays to start the manifold walks in the same way as before, but at each interaction we additionally decide between reflection or refraction (if applicable), and whether or not to terminate the chain and attempt to connect to a light position. To keep variance manageable we found it best to limit SMS to a single family of light paths in this setting (e.g. paths of fixed length with only refractive events). In cases where this increased variance is not acceptable, the biased technique can still be applied—but for the same reasons there will be a potentially significant energy loss. As shown in the Double-Refractive Slab scene, SMS can still produce good results, but better strategies for sampling initial configurations will be required to turn SMS into a fully general sampling strategy that can efficiently find *all possible* chains of specular interactions.

3.4.8 Glints

Figure 3.30 examines the performance of our glint rendering technique on two scenes with complex microstructure specified using normal maps. We compare our SMS to the state-of-the-art method of Yan et al. [137]. Both methods make use of MIS in this comparison. The Shoes scene features a glittery pair of shoes with procedural normal displacement by a Gaussian height field and is lit by a sky with an almost purely directional sun. The Kettle scene involves brushed metal with very strong anisotropy. It is illuminated by the *Grace Cathedral* environment map, which includes several bright and narrow light sources. Stochastic sampling of glint solutions is beneficial in these cases, since several complementary sources of variance can be reduced at the same time.

We found the biased SMS variant to generally be more practical for glint rendering compared to its unbiased counterpart[8]. The potentially unbounded number of iterations in the recursive unbiased probability estimator, combined with extremely high-frequency normal map detail, occasionally produces acute outliers in the pixel estimate that lead to poor convergence, as seen in the plots.

[8]The bias here only involves the probability estimate. In practice, even the “unbiased” version will not match a brute-force result perfectly due to the far-field approximation.

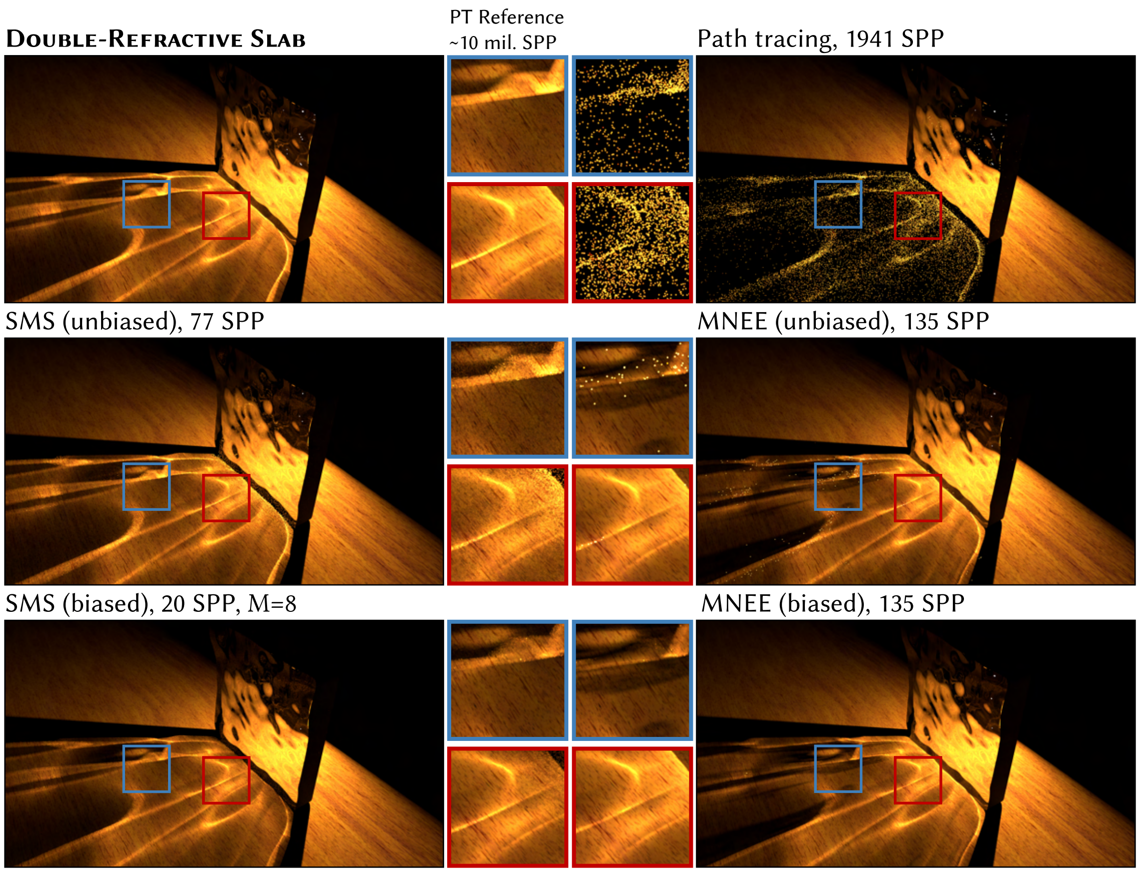


Figure 3.29: Equal-time comparison (5 minutes) of our method on a challenging scene where SMS samples a light connection involving two consecutive specular refractions. The two-sided solid piece of glass is modeled with geometric displacement.

Note that all methods in Figure 3.30 converge to slightly different results, but they all find the same individual glints and have very similar appearance overall. Our method uses 100–300× less memory compared to previous work (e.g. 110 MiB vs. 11 GiB). At the same time, it converges in an equal or shorter amount of time and still generates temporally coherent animations.

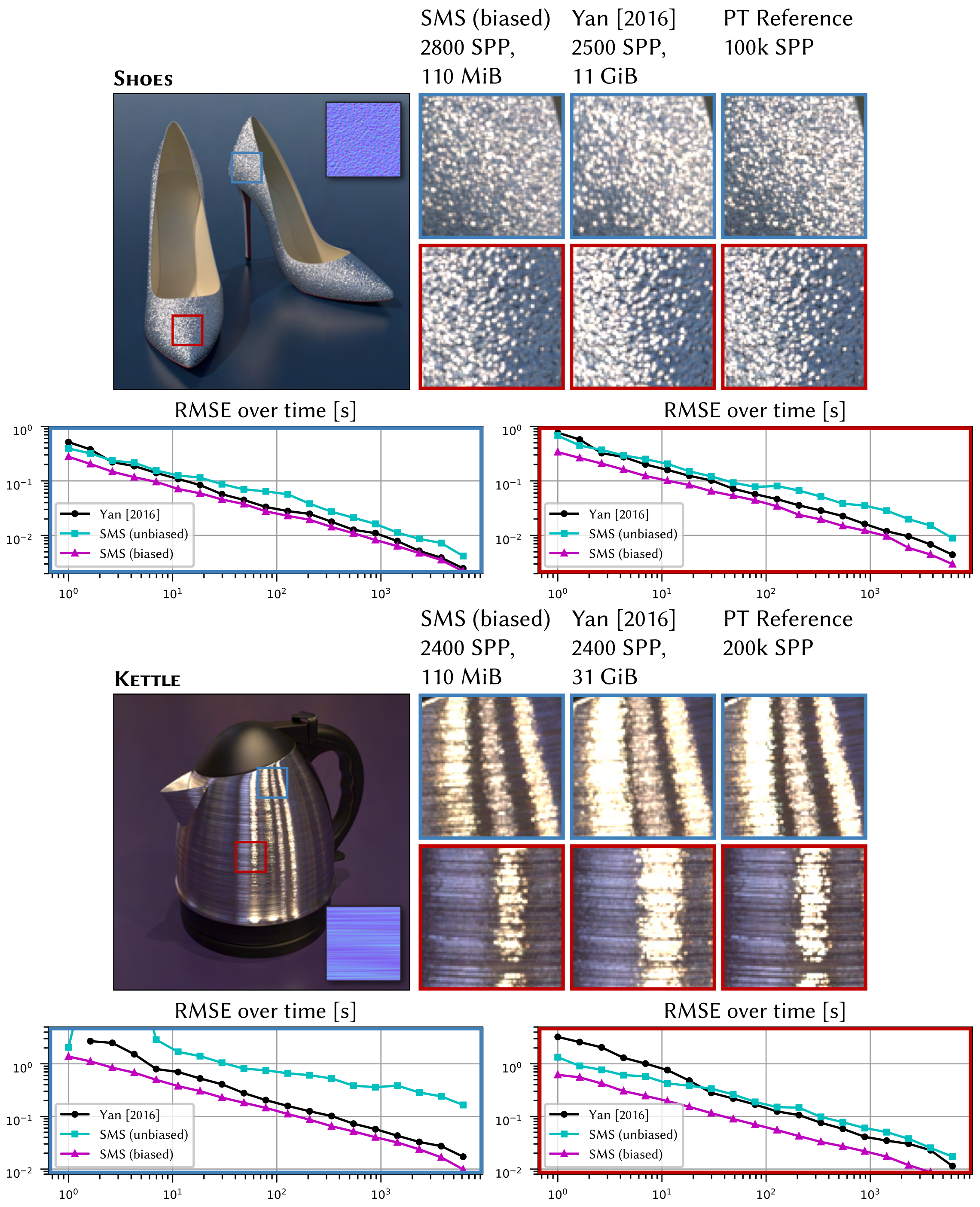


Figure 3.30: Equal-time comparison (9 minutes) of our glint rendering method to prior work specific to this problem [137]. Shoes scene: highly directional illumination from the sun, Kettle scene: *Grace Cathedral* environment map where integration over multiple sources of variance (i.e. all the lights) is critical. Our method yields comparable results in the first scenario and is superior in the second. At the same time, our method requires 100–300× less memory. The insets and corresponding convergence plots focus on different parts of the glinty appearance.

3.5 Summary and future work

We introduced a simple and powerful specular path sampling technique that combines deterministic root-finding with stochastic sampling in a pure Monte Carlo setting. The basic method can be used in a variety of different ways, and we demonstrated example applications in the context of efficient path tracing of glints and caustics. Our approach is not restricted to unidirectional path tracing, and we contemplate its utility in bidirectional and even MCMC methods, where manifold walks were originally proposed.

Building on a simple unbiased algorithm, we presented several complementary extensions and improvements. For example, better strategies for sampling seeds paths can further improve convergence, and such heuristics are easy to integrate into our method without introducing bias. Improved manifold constraints expand the size of the convergence basins in primary sample space. A further change yields an intentionally biased estimator with desirable properties for production usage. Far-field approximations in the context of specular glints lead to a particularly simple iterative algorithm, whose steps no longer require the use of ray tracing operations. In the future, we would like to explore further acceleration of this variant leveraging vectorized execution.

Determining *when* to use our method is another important aspect for future investigation. Attempting many connections that are ultimately unsuccessful can consume a large amount of computation. While glints would benefit from simple culling heuristics, e.g. based on cones bounding the normal variation inside the pixel, the general case of caustics from arbitrary specular geometry is significantly more challenging. Combining our techniques with others via multiple importance sampling in this general setting is another pertinent problem.

Our discussion focuses mainly on the generation of subpaths with a single specular vertex. While our method in principle also generalizes to more complex path classes with multiple specular reflection, performance using our current strategy for choosing starting points remains suboptimal and could be an interesting topic for future work. We wish to pursue these and related improvements, and envision a unified path sampling strategy that elevates stochastic manifold walks to a standard building block in the design of Monte Carlo rendering methods.

4 | Monte Carlo Estimators for Differential Light Transport

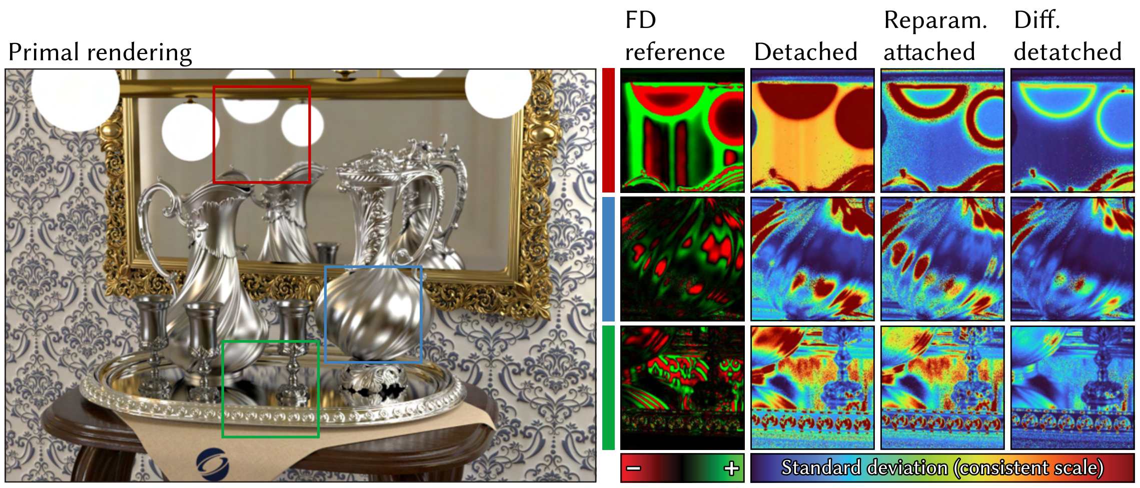


Figure 4.1: Differentiable rendering of a scene featuring specular interreflection between metallic surfaces of varying roughness. We differentiate the image with respect to the combined roughness of all objects, which produces the gradients shown in the first column with insets (computed using finite differences and an impractically high sample count). A disconcertingly large number of differential estimators can solve this problem, albeit with drastically different statistical efficiency: the following columns highlight the standard deviation of MIS estimators involving emitter sampling and three different strategies based on BSDF sampling. An overview of the exhaustive set of combinations (21 methods) and results for an additional five estimators are provided later in Section 4.5, which also contains uncropped images. The objective of our work is to provide intuition on how to navigate the large design space of differential Monte Carlo estimators.

We now transition from the discussion of specular light paths in the context of *forward* rendering to the *inverse* rendering problem.

As extensively discussed in Chapter 2, image formation is generally the result of the complex interplay of shape, illumination, and materials, in which indirect effects like shadowing and interreflection couple distant parts of the scene: a bright spot on a surface could, e.g., be explained by texture or shape variation, illumination from a light source, or focused reflection from another object. Resolving this ambiguity requires multiple observations and reconstruction techniques that account for the interconnected nature of light transport and scattering.

In this chapter, we study the mathematical principles of *differentiable rendering*, which formulates the inversion process as a gradient-based optimization task defined on a high-dimensional domain with millions of scene parameters specifying illumination, shapes, and materials. Scene parameter derivatives of a rendered image encode important cues that can be used to unravel this radiative coupling.

So far, the creation of differentiable rendering algorithms has followed a fairly rigid sequence of steps: derivatives are first moved into light transport integrals solved by a standard method (e.g. path tracing), possibly with extra steps to handle visibility-related discontinuities. Subsequent differentiation of the integrand involves the standard rules of calculus and can be performed by hand, or using software-based techniques for *automatic differentiation (AD).*

While this approach generally works, we observe that differentiation fundamentally changes the nature of the underlying integrals. A scene parameter can be *sensitive* in the sense that a small perturbation of its value would lead to a significant positive or negative change in the value of the integrand that affects the rendered image and optimization objective. Monte Carlo theory then tells us that a low-variance gradient estimator should place a proportional number of samples into this region. However, this type of adaptation is simply impossible when the differential rendering algorithm is rigidly created from its primal counterpart. In the worst case, the sensitive region could even be zero-valued and discarded during primal integration, in which case the differential algorithm is biased.

Recent work by Nimier-David et al. [155] proposed a method termed *radiative backpropagation (RB)*, which casts differentiable rendering into the form of an *adjoint* (i.e. reversed) transport problem that propagates derivative "radiation" from sensors towards objects with differentiable parameters. Their formulation decouples the primal and differential estimators and provides the starting point for our investigation. This decoupling brings considerable additional freedom but also reveals that elementary aspects of differentiable rendering remain poorly understood. In this chapter, we investigate the following choices that guide the design of differential transport estimators:

1. Estimators that apply importance sampling often do so using the inversion method, which involves a mapping to transform uniform variates to the target distribution. When creating the differential estimator, this mapping could remain unchanged, or it could be differentiated along with the integrand. We refer to these respectively as *detached* and *attached* strategies. The former produce static samples, while the latter capture the infinitesimal motion of samples with respect to parameter changes.

2. Sampling strategies are almost never used alone, but in combination with others via the framework of multiple importance sampling (MIS). Once more, the primal MIS weights could be used as-is or differentiated to track infinitesimal changes.

3. Sampling strategies are designed to approximate the shape of the associated integrand, but this property may no longer hold following differentiation regardless of whether attached or detached strategies are used. In such cases, it may be possible to design tailored strategies that match this new integrand, which we refer to as *differential strategies.* We propose one such strategy for the commonly used family of microfacet models. Our analysis demonstrates clear benefits of specialized differential strategies, while the trade-offs between detached and attached sampling remain more nuanced and problem-dependent.

4. Visibility-related discontinuities require careful treatment to avoid bias in computed gradients. We explain how recent techniques that are designed to sidestep this issue [73, 156] can be adapted to the adjoint framework of RB, enabling efficient and unbiased geometric optimization.

5. Not all options are compatible with each other: some estimator combinations of attached/detached MIS weights and attached/detached sampling strategies yield biased results, and attached sampling strategies can interfere with techniques to handle discontinuous integrals. We show how sampling strategies can simultaneously be attached yet behave correctly in the presence of discontinuities.

6. Finally, rendering algorithms frequently take discrete random decisions including path termination via Russian roulette and sampling of multi-lobed BSDFs. We show that these steps should *never* be differentiated, as this would severely bias their result.

After discussing the relevant previous work in Section 4.1, the remainder of this chapter provides a taxonomy of differential Monte Carlo estimators based on this bewilderingly large set of possibilities. Figure 4.1 shows a preview of three such estimators with very different statistical efficiency.

4.1 Background

Inverse rendering is a standard problem in computer vision, where a considerable body of prior work has investigated ways of differentiating the process of image formation. Effects like shadows, interreflections, depth of field, or global illumination in general have historically played a lesser role during this process, and related works investigating differentiable rasterization of meshes and volumes thus mainly focus on primary visibility [157, 158, 159, 160, 161, 162].

As we are interested in differentiating physically based rendering algorithms, including support for such indirect effects, we consider Monte Carlo sampling of integrals of the form

$$I(\boldsymbol{\pi}) = \int_{\mathcal{X}} f(\mathbf{x}, \boldsymbol{\pi})\, \mathrm{d}\mu(\mathbf{x}), \tag{4.1}$$

where $\boldsymbol{\pi}$ refers to a set of scene parameters. The domain $\mathcal{X}$ typically consists of light paths connecting a light source to a sensor via a number of intermediate scattering events. In this work, we are concerned with individual (hemi-)spherical integrals that may reference nested integrals, hence we set $\mathcal{X} = \mathcal{S}^2$. We have not investigated path-space methods [163, 164], though we suspect that many of our observations will generalize.

Physically based differentiable rendering algorithms [4, 165, 166] estimate the partial derivative of the above integral with respect to $\boldsymbol{\pi}$:

$$\partial_{\boldsymbol{\pi}} I(\boldsymbol{\pi}) = \partial_{\boldsymbol{\pi}} \left[\int_{\mathcal{X}} f(\mathbf{x}, \boldsymbol{\pi})\, \mathrm{d}\mu(\mathbf{x}) \right], \tag{4.2}$$

where we use a shorthand notation $\partial_{\boldsymbol{\pi}} := {}^{\partial}/_{\partial_{\boldsymbol{\pi}}}$ from now on.

Indirect effects are especially important when optimizing materials like participating media that are characterized by significant multiple scattering [164, 167, 168, 169, 170, 171]. Initial work on physically based differentiable rendering relied on forward mode differentiation to propagate an infinitesimal perturbation through the simulation, requiring a separate run for each parameter of interest. Later techniques applied reverse mode differentiation [4, 165] to compute derivatives with respect to all scene parameters at once.

Reverse mode differentiation is a widely used tool [115] that greatly improves the efficiency when many derivatives are desired, but it also introduces its own set of problems: derivative evaluation now requires access to intermediate steps of the primal computation, and this sequence of accesses furthermore occurs in reverse order compared to the original program execution. Program reversal is impractical without at least some temporary storage of primal variables, and the size of this scratch space tends to be overwhelming in the context of rendering.

The *radiative backpropagation (RB)* method [155] addresses this issue by observing that the derivative program effectively solves a separate type of transport problem where derivative "radiation" that corresponds to the derivative of the objective in pixel space is "emitted" from the camera, "scatters" from scene objects, and is eventually "received" by differentiable scene objects that now take the role of the sensor. Instead of being constrained by the inflexibility and memory overheads of automatically differentiating a primal algorithm in reverse mode, one can thus create differential algorithms that directly solve this modified transport problem. Our work builds on this idea and leverages the decoupled nature of primal and differential phases.

Naïvely differentiating light transport generally leads to severely biased gradients due to discontinuities inside the associated integrals, e.g. due to visibility changes at object boundaries in the common case of differentiating scene geometry parameters. One solution to this is explicit sampling of a *boundary integral* over silhouette edges in the scene, as originally proposed by Li et al. [165] and later extended by Zhang et al. [163]. Alternatively, Loubet et al. [156] show how to solve the same problem purely with interior integrals by applying a change of variables. Their approach introduces bias, but this limitation was later overcome by a more general approach by Bangaru et al. [73]. Recently, Zhou et al. [172] also proposed an analytical and differentiable visibility computation approach based on beam tracing [173]. However, none of these techniques are readily usable in the framework of RB. We show how the method of Bangaru et al. can be used as a reparameterization that is queried as part of a memory-less reverse mode differentiation procedure.

Interestingly, bias due to discontinuities can also arise from attached sampling strategies, which happens even when geometry is not part of the optimization process! We introduce a modified parameterization that addresses this problem.

Concurrently to our analysis, Vicini et al. [174] propose algorithms that can evaluate attached and detached differential estimators using linear time and constant space complexity. These are mostly orthogonal to the topics discussed in this chapter, and both works can be combined.

We want to point out that a differentiable renderer is useful beyond mere optimization. It is also useful when rendering occurs as a part of a larger differentiable calculation such as a neural autoencoder or generative adversarial network. However we do not pursue this direction in this thesis.

Derivatives of Monte Carlo simulations have also been used outside of rendering, to model the criticality of nuclear reactors [175], and to perform inverse modeling of tissue [176]. These types of methods are named *Perturbation Monte Carlo* or *Differential Monte Carlo.*

We now look at the mathematical background of inverse and differential light transport more closely. First, we look at how differentiation is used to solve inverse rendering problems via gradient-based optimization (Section 4.1.1). We also review the radiative backpropagation algorithm by Nimier-David et al. [155] (Section 4.1.2) and discuss the problem of discontinuities in much more detail (Section 4.1.3).

4.1.1 Inverse rendering overview

While *forward* rendering generates an image $\mathbf{I}(\boldsymbol{\pi})$ from a number of scene input parameters $\boldsymbol{\pi} \in \Pi$, *inverse* rendering problems attempt to recover parameters that produce an image which minimizes a given loss or objective function $\ell(\cdot)$:

$$\boldsymbol{\pi}^* = \arg\min_{\boldsymbol{\pi}} \; \ell(\mathbf{I}(\boldsymbol{\pi})) \tag{4.3}$$

Mathematically, the output of image formation is simply a vector that collects the measurements (2.47) of all image pixels $\mathbf{I}(\boldsymbol{\pi}) = (I_1(\boldsymbol{\pi}), \ldots, I_n(\boldsymbol{\pi}))^T$, here written with an explicit dependence on a vector of scene parameters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_p)^T$.

The loss function is problem specific; simple examples could be the standard L1 or L2 loss based on one or multiple target images. However, more complex objectives, e.g. based on a black-box neural network, are also conceivable.

Due to the complex nature of light transport (Section 2.3), direct solutions to Equation (4.3) are usually out of the question and we instead use nonlinear optimization techniques such as *gradient descent.* Starting from an initial guess $\boldsymbol{\pi}_{(0)}$, we then follow the direction of steepest descent on the loss landscape as given by the parameter gradients. This process potentially repeats many times. The update equation of the most basic descent scheme reads

$$\boldsymbol{\pi}_{(i+1)} = \boldsymbol{\pi}_{(i)} - \gamma \cdot \nabla \ell(\mathbf{I}(\boldsymbol{\pi})) = \boldsymbol{\pi}_{(i)} - \gamma \cdot \left(\frac{\partial \ell}{\partial \boldsymbol{\pi}}\right)^T, \tag{4.4}$$

where $\boldsymbol{\pi}_{(i)}$ is the parameter state at iteration i and γ is the *learning rate* or *step size.* Note that we are technically performing a form of *stochastic* gradient descent in practice, as the computed parameter gradients are usually contaminated by Monte Carlo noise. An

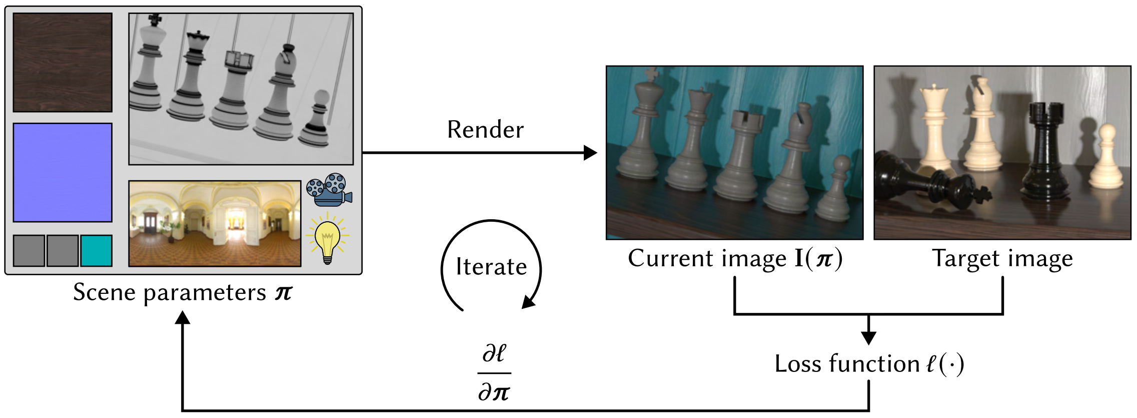

Render
Iterate
Current image $\mathbf{I}(\boldsymbol{\pi})$
Target image
Scene parameters $\boldsymbol{\pi}$
$\frac{\partial \ell}{\partial \boldsymbol{\pi}}$
Loss function $\ell(\cdot)$

Figure 4.2: The typical optimization loop found in inverse rendering applications. A set of (initial) scene parameters $\boldsymbol{\pi}$ passes through a rendering system, producing an image $\mathbf{I}(\boldsymbol{\pi})$. Following comparison to a reference (using a loss function ℓ), the gradient $\partial\ell/\partial\boldsymbol{\pi}$ is computed and used to update the parameter state. Under reasonable assumptions on the step size, repeated iterations of this process will converge to a (local) optimum according to ℓ.

alternate option is the *ADAM* optimizer [177] that additionally applies momentum and scales the step size based on each parameter's variance.

If the step size is chosen sufficiently small, either of these approaches usually converge to a *local* optimum. Unfortunately, the loss landscapes of inverse rendering problems are rarely convex, and thus, the procedure does often not converge to a *global* optimum. It is advantageous to either add regularization to this process (e.g. by adding terms to the loss function that penalize undesired characteristics of a solution) or to start with a good initial guess to avoid local minima[1].

An overview of the standard optimization loop for inverse rendering is shown in Figure 4.2. We now turn to the actual computation of the required parameter gradients

$$\frac{\partial \ell}{\partial \boldsymbol{\pi}} = \frac{\partial \ell}{\partial \mathbf{I}} \cdot \frac{\partial \mathbf{I}}{\partial \boldsymbol{\pi}} \tag{4.5}$$

from Equation (4.4), which can intuitively be understood as the sensitivity of the input parameters due to a change in the loss. The first term in the product is relatively simple: it is a row vector of dimension n, i.e. the number of image pixels. The second term is a Jacobian matrix of size $n \times p$. Recall that we would like to jointly optimize a large number (e.g. millions) of parameters and that, in principle, each parameter can affect each image pixel due to global illumination. This means, the required matrix is both dense and large

[1]We note that, in principle, it would be desirable to use more sophisticated optimization approaches (e.g. Newton's method) that have better convergence rates. However, they require higher order derivative terms which are prohibitively expensive to compute.

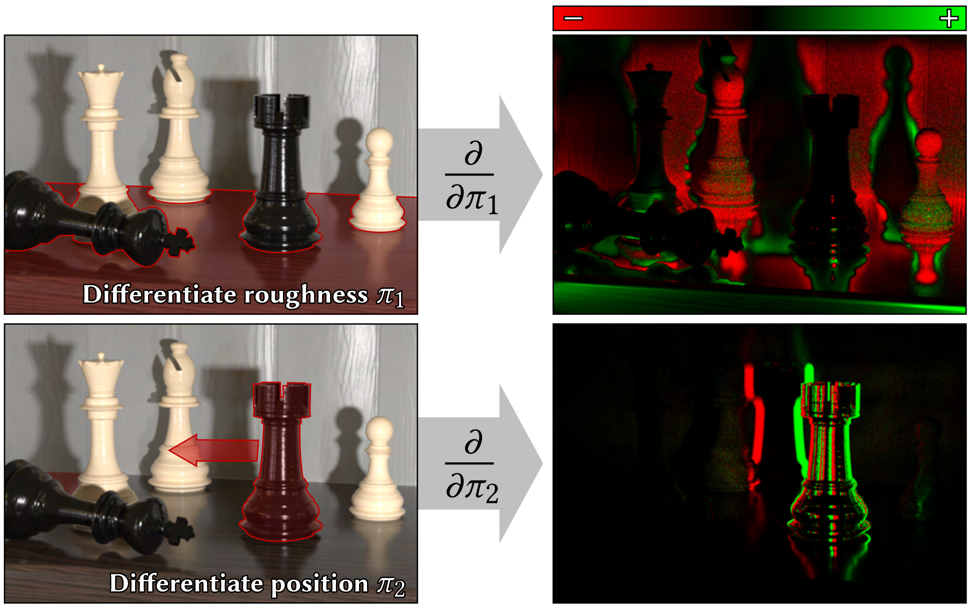

−
+
Differentiate roughness π_1
$\frac{\partial}{\partial \pi_1}$
Differentiate position π_2
$\frac{\partial}{\partial \pi_2}$

Figure 4.3: Two examples of (false color) gradient images that show changes of pixel intensities due to infinitesimal changes to individual parameters. **Top**: A change of the table surface roughness parameter. Most of the image is affected by this as the table reflects large parts of the incident illumination into the rest of the scene. **Bottom**: Translation of the rook piece to the left. Even this causes a global effect due to its shadow and reflections.

in practice, which makes its computation troublesome. As outlined in Section 2.6 there are two main approaches to sidestep a full matrix computation.

Applying forward differentiation to $\partial \mathbf{I}/\partial \pi$ generally requires p separate rendering passes, i.e. one per scene parameter. For example, each pass could compute a column of this matrix in the form of a *gradient image*. This is conceptually simple and could even be accomplished with finite differences (in that case requiring $2p$ passes), but is clearly not suited for our optimization task where p is commonly very large. However, gradient images are valuable tools during the development of differentiable rendering algorithms as they can help to better understand and visualize the sensitivity of the pixel intensities based on individual inputs. We show two such examples in Figure 4.3 which also illustrate how potentially the whole image can be affected by a change to a single input.

In contrast, reverse mode differentiation is much better suited for this problem: starting from the loss output, a single backward pass is sufficient to compute all required parameter gradients. This however comes with the downside of requiring storage of all intermediate results of the program in form of a computation graph.

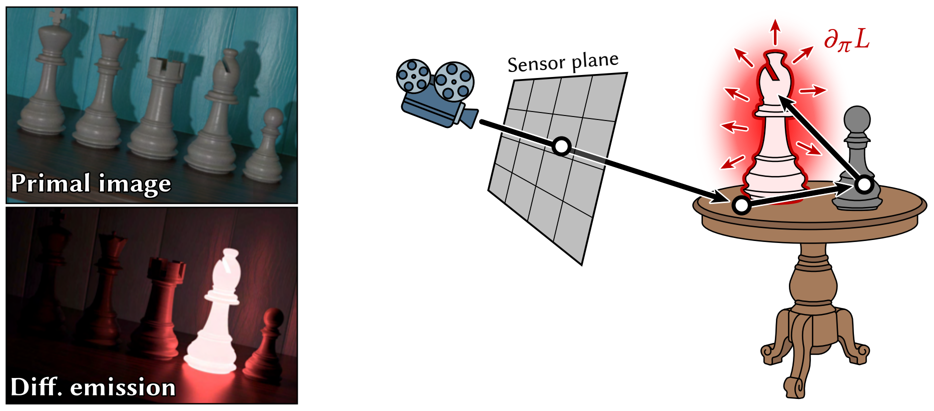


Figure 4.4: **Left**: An illustration of differentiating light transport based on a single scene parameter. In this specific case, we perform an infinitesimal change to the color of the bishop piece. This causes it to "emit" differential radiance (visualized in red) into the scene which then scatters like normal radiance. **Right**: A forward mode rendering approach that generates a corresponding gradient image by measuring differential radiance arriving at the pixels of the sensor.

There is an important detail here that was noted by Gkioulekas et al. [168] and Azinović et al. [166]: both the forward rendering (used during the computation of the loss) and the reverse pass through the program are usually a form of Monte Carlo integration that use a shared random number state. But this introduces undesirable correlation between the two terms in Equation (4.5). This means the expectation of their product is no longer equal to the product of their individual estimates ($\mathrm{E}[X \cdot Y] \neq \mathrm{E}[X] \cdot \mathrm{E}[Y]$, see Section 2.4.1). We therefore need to compute the rendered image *twice* using different random number generator seeds, where the reverse pass that propagates gradients is only performed after running the forward rendering process for the second time.

4.1.2 Radiative backpropagation

Unfortunately, naïve application of reverse mode differentiation to a full light transport algorithm (e.g. path tracing) leads to serious scalability issues. The bookkeeping and traversal of the computation graph incur significant computational overhead that would dominate the computation if done separately for each of the millions of light paths that must be traced to generate a typical image.

This type of overhead can, however, be reduce by batching Monte Carlo samples using a variant of AD termed *vector mode* [115], in which the additional bookkeeping cost can be effectively amortized. In the context of rendering, this entails tracing *wavefronts* of light paths [178].

Unfortunately, this turns out to be extremely memory intensive. The memory footprint grows with the image resolution, the number of Monte Carlo samples, and the length of the simulated paths. Relatively modest settings for these parameters can easily cause the computation to exceed the available memory, even on high-end GPUs [4, 155], which hinders scalability beyond small toy problems. Recently, Nimier-David et al. [155] introduced the radiative backpropagation (RB) algorithm, a more practical approach for reverse mode gradient propagation that is inspired by the adjoint sensitivity method from the area of optimal control [179, 180]. The authors investigate what differentiation actually does to the underlying light transport problem and propose a specialized backpropagation technique that avoids the costly storage overheads.

Consider again the three fundamental light transport equations previously discussed in Section 2.3.1:

1. **Measurement**:

$$I_k = \int_{\mathcal{M}} \int_{\mathcal{S}^2} W_{\mathrm{e}}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})\, \mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}) \tag{4.6}$$

2. **Transport**:

$$L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}) = L_{\mathrm{o}}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega}) \tag{4.7}$$

3. **Scattering**:

$$L_{\mathrm{o}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}} \to \boldsymbol{\omega}_{\mathrm{o}})\, L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{i}})\, \mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega}_{\mathrm{i}}) \tag{4.8}$$

Nimier-David et al. transform these equations by applying derivatives on both the left and right hand sides. This yields a new set of "physical" statements that will be instructive to find methods for more efficient gradient propagation. To be more precise, the next three equations can be interpreted as describing the transport of an alternate, hypothetical quantity called *differential radiance* $\partial_{\boldsymbol{\pi}} L$. Like its ordinary counterpart, it also exists in incident, outgoing, and emitted variants $\partial_{\boldsymbol{\pi}} L_{\mathrm{i}}$, $\partial_{\boldsymbol{\pi}} L_{\mathrm{o}}$, and $\partial_{\boldsymbol{\pi}} L_{\mathrm{e}}$. We now discuss these three new energy balance equations in turn:

1. **Differential measurement**: Measurement of differential radiance is equivalent to the case of normal radiance, assuming the sensor itself is not being differentiated, i.e. $\partial_{\boldsymbol{\pi}} W_{\mathrm{e}}^{(k)}(\mathbf{x}, \boldsymbol{\omega}) = 0$:

$$\partial_{\boldsymbol{\pi}} I_k = \int_{\mathcal{M}} \int_{\mathcal{S}^2} W_{\mathrm{e}}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, \partial_{\boldsymbol{\pi}} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega})\, \mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}). \tag{4.9}$$

2. **Differential transport**: Differential radiance also propagates normally, i.e. its value remains unchanged between surface points along a ray:

$$\partial_\pi L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}) = \partial_\pi L_\mathrm{o}(\mathbf{r}(\mathbf{x}, \boldsymbol{\omega}), -\boldsymbol{\omega}). \tag{4.10}$$

3. **Differential scattering**: This is the most interesting part, as applying the product rule to the ordinary rendering equation creates three new terms

$$\begin{aligned}\partial_\pi L_\mathrm{o}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) = \partial_\pi L_\mathrm{e}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{o}) & \qquad \text{(T1)} \\ + \int_{\mathcal{S}^2} \partial_\pi f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}) & \qquad \text{(T2)} \\ + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o})\, \partial_\pi L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}) & \qquad \text{(T3)}\end{aligned} \tag{4.11}$$

with the following interpretations:

T1. Differential radiance $\partial_\pi L_\mathrm{o}$ is "emitted" not from the usual emission term L_e, but its differential version. This is non-zero whenever emission changes as a result of a differentiated scene parameter, e.g. a light source changes its intensity.

T2. Differential radiance is also "emitted" due to changing BSDF parameters. This depends additionally on the (ordinary) incident radiance L_i. This is intuitive as changing material parameters of a brightly lit object has a greater influence on the final image compared to one that sits in a dark corner of the scene.

T3. Differential radiance scatters just like ordinary radiance based on the normal BSDF f_s.

These similarities to forward rendering suggest that we can perform forward mode differentiation by running another light transport simulation with an algorithm analogous to path tracing. Paths are generated from the sensor side and we use Monte Carlo integration to approximate the integral terms above. This ultimately generates a gradient image, as long as we can evaluate the (local) derivative terms $\partial_\pi L_\mathrm{e}$ and $\partial_\pi f_\mathrm{s}$, e.g. via AD. An abstract illustration is shown in Figure 4.4.

As discussed, forward mode differentiation is not well-suited for the type of optimization tasks we aim to solve—but the remarkable insight behind RB is that we can also cast the reverse mode propagation of gradients into a similar light transport formulation.

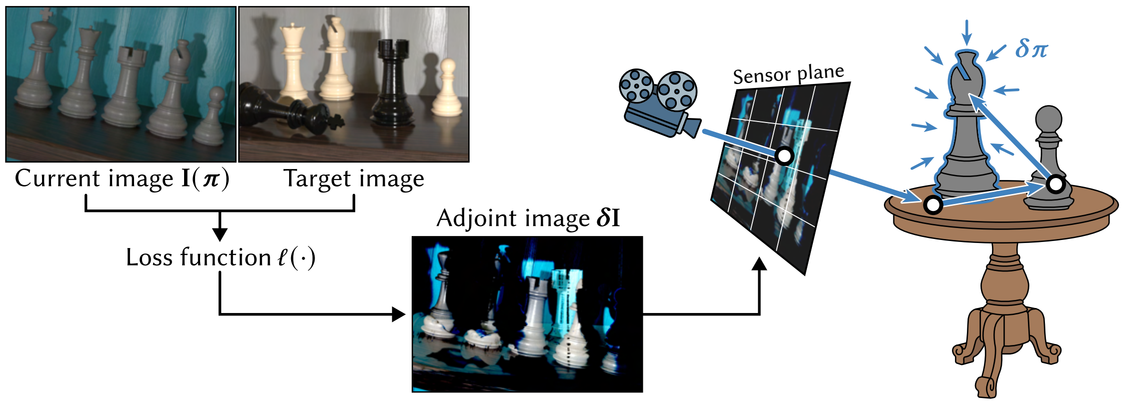

Current image $\mathbf{I}(\boldsymbol{\pi})$
Target image
Loss function $\ell(\cdot)$
Adjoint image $\boldsymbol{\delta}\mathbf{I}$
Sensor plane
$\delta\pi$

Figure 4.5: A loss function ℓ is used to compare a current state of the optimization against a given target image. A simple backpropagation through ℓ gives the adjoint rendering $\boldsymbol{\delta}\mathbf{I}$, which encodes how each pixel should change to improve the loss. These values are then "emitted" into the scene from the camera. Finally, when adjoint radiance interacts with scene objects, its value is backpropagated to the corresponding (differentiable) scene parameters.

Recall Equation (4.5) from earlier. In reverse mode, we want to propagate a perturbation of the scalar image loss $\boldsymbol{\delta}\ell$ first to an intermediate result

$$\boldsymbol{\delta}\mathbf{I} = \left(\frac{\partial \ell}{\partial \mathbf{I}}\right)^T \cdot \boldsymbol{\delta}\ell \tag{4.12}$$

called the *adjoint rendering*. This is a straightforward application of the chain rule to the loss function, see Figure 4.5. The subsequent step is much harder as it needs to backpropagate to the scene parameters through the rendering process:

$$\boldsymbol{\delta\pi} = \left(\frac{\partial \mathbf{I}}{\partial \boldsymbol{\pi}}\right)^T \cdot \boldsymbol{\delta}\mathbf{I} = \begin{bmatrix} \frac{\partial I_1}{\partial \pi_1} & \cdots & \frac{\partial I_n}{\partial \pi_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial I_1}{\partial \pi_p} & \cdots & \frac{\partial I_n}{\partial \pi_p} \end{bmatrix} \cdot \boldsymbol{\delta}\mathbf{I}. \tag{4.13}$$

As mentioned previously, the Jacobian matrix here is generally dense and a full computation is infeasible. We can however substitute the differentiated measurement equation (4.9) which gives us

$$\begin{aligned}\left(\frac{\partial \mathbf{I}}{\partial \boldsymbol{\pi}}\right)^T \cdot \boldsymbol{\delta}\mathbf{I} &= \sum_{k=1}^{n} \boldsymbol{\delta}\mathbf{I}_k \cdot \partial_{\boldsymbol{\pi}} I_k \\ &= \int_{\mathcal{M}} \int_{\mathcal{S}^2} \left[\sum_{k=1}^{n} \boldsymbol{\delta}\mathbf{I}_k \cdot W_{\mathrm{e}}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\right] \partial_{\boldsymbol{\pi}} L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}) \,\mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega}) \,\mathrm{d}A(\mathbf{x}),\end{aligned} \tag{4.14}$$

where $\boldsymbol{\delta}\mathbf{I}_k$ represents the k-th pixel of the adjoint rendering.

Nimier-David et al. observed an interesting symmetry relationship here between differential radiance $\partial_{\pi} L$ and yet another new quantity which they call *adjoint radiance*[2]:

$$A_e(\boldsymbol{\delta}\mathbf{I}, \mathbf{x}, \boldsymbol{\omega}) := \sum_{k=1}^{n} \boldsymbol{\delta}\mathbf{I}_k \cdot W_{\mathrm{e}}^{(k)}(\mathbf{x}, \boldsymbol{\omega}). \tag{4.15}$$

This quantity can be understood as being "emitted" into the scene from the side of the sensor. It then scatters through the scene until it is eventually "received" by objects whose parameters are being differentiated. In practice, this is a backpropagation step through either the $\partial_{\pi} L_{\mathrm{e}}$ or $\partial_{\pi} f_s$ terms in Equation (4.11). In other words, this performs a (transposed) Jacobian-vector product—but this time with a very sparse matrix where only the local emitter or BSDF parameters are involved, e.g. nearby texels involved in a query of a textured BSDF. Figure 4.5 illustrates the high-level idea.

An implementation of RB is strikingly similar to a standard unidirectional path tracer. We show pseudocode of a basic version of the algorithm in Algorithms 4.1 and 4.2. The notation `adjoint([[ <expr> ]], grad_out)` is reused from Nimier-David et al. [155] and refers to the reverse mode derivative of an expression `<expr>` that backpropagates a gradient with respect to its output (`grad_out`) towards the scene parameters $\boldsymbol{\pi}$, while returning another gradient that results from this step. Please refer to the original publication for more details about the derivation and implementation of RB.

Radiative backpropagation enjoys a number of advantages compared to standard reverse mode differentiation of a rendering algorithm:

1. The method does not require memorization of all intermediate variable values computed during the primal rendering phase.

2. As a consequence, we also do not need to fall back to wavefront rendering in order to amortize costly graph creation and traversal. Instead, RB can be implemented in a *megakernel* style that keeps memory usage extremely low, see Nimier-David et al. [155].

3. Because the gradient propagation step is now decoupled from the primal program execution that renders an image, we can be more conscious about how we solve the equations above. Many techniques known from forward rendering could now be repurposed for the differentiated integrals. Our analysis in this chapter is one of the first instances where we explore this direction and we will discuss some of these new possibilities in detail later in Sections 4.2 and 4.3.

[2]This is closely related to the relation between radiance and importance in ordinary light transport, see Section 2.3.4.

```
def radiative_backprop($\boldsymbol{\pi}$, $\boldsymbol{\delta}\mathbf{I}$):
    # Initialize parameter gradient(s) to zero
    $\boldsymbol{\delta\pi} = 0$
    for _ in range(num_samples):
        # Sample ray proportional to sensor response and pixel filter
        ($\mathbf{x}$, $\boldsymbol{\omega}$), s_weight = sensor.sample_ray()
        # Query adjoint emitted radiance associated with current ray
        A_e = $A_{\mathrm{e}}(\boldsymbol{\delta}\mathbf{I}, \mathbf{x}, \boldsymbol{\omega})$ / num_samples
        # Propagate adjoint radiance into the scene
        $\boldsymbol{\delta\pi}$ += radiative_backprop_Li($\boldsymbol{\pi}$, x, $\boldsymbol{\omega}$, A_e * s_weight)
    # Finished, return gradients
    return $\boldsymbol{\delta\pi}$
```

Algorithm 4.1: Radiative backpropagation takes scene parameters $\boldsymbol{\pi}$ and an adjoint rendering $\boldsymbol{\delta}\mathbf{I}$ as input. This pseudocode fragment is responsible for the measurement integral. It samples a set of sensor rays, queries the associated emitted adjoint radiance A_e and propagates these gradients into the scene.

```
def radiative_backprop_Li($\boldsymbol{\pi}$, x, $\boldsymbol{\omega}$, $\boldsymbol{\delta}L$):
    # Find an intersection with the scene geometry
    x′ = $\mathbf{r}(\mathbf{x}, \boldsymbol{\omega})$
    # T1: Backpropagate to emitter parameters, if any
    $\boldsymbol{\delta\pi}$ = adjoint([[ $L_{\mathrm{e}}(\mathbf{x}', -\boldsymbol{\omega})$ ]], $\boldsymbol{\delta}L$)
    # Sample a direction from the BSDF
    $\boldsymbol{\omega}'$, b_val, b_pdf = sample $f_{\mathrm{s}}(\mathbf{x}', \boldsymbol{\omega}' \to -\boldsymbol{\omega})$
    # T2: Backpropagate to parameters of BSDF, if any
    $\boldsymbol{\delta\pi}$ += adjoint([[ $f_{\mathrm{s}}(\mathbf{x}', \boldsymbol{\omega}' \to -\boldsymbol{\omega})$ ]], $\boldsymbol{\delta}L$ * $L_{\mathrm{i}}(\mathbf{x}, \boldsymbol{\omega}')$ / b_pdf)
    # T3: Recurse to account for indirect differential radiance
    return $\boldsymbol{\delta\pi}$ + radiative_backprop_Li($\boldsymbol{\pi}$, x′, $\boldsymbol{\omega}'$, $\boldsymbol{\delta}L$ * b_val / b_pdf)
```

Algorithm 4.2: This pseudocode fragment provides `radiative_backprop_Li()` referenced in Algorithm 4.1. It implements the RB version of the transport and scattering equation that transports derivatives through the scene and backpropagates adjoint radiance $\boldsymbol{\delta}L$ to objects with differentiable parameters.

In its originally proposed form, RB also suffers from some limitations however. When backpropagating adjoint radiance to the BSDF parameters we need to evaluate the (ordinary) incident radiance at that location, see the L_i term as part of (T2) in Equation (4.11) and Line 9 of Algorithm 4.2. This requires that we perform another recursive Monte Carlo estimation at each path vertex, e.g. by referring to a normal path tracer. As a result, RB has quadratic run time complexity in the length of simulated light paths. This is especially concerning in scenes where many bounces of global illumination are required. This problem was recently addressed by Vicini et al. [174] with the *path replay backpropagation* algorithm. This manages to compute gradients in linear time by replaying light paths sampled during an additional primal rendering step in reverse.

So far we also neglected a very important corner case of differentiable rendering: the integrals we want to solve are generally riddled with discontinuities in the incident radiance (e.g. due to object silhouettes and visibility changes), and the position of these discontinuities furthermore depends on the scene parameters $\boldsymbol{\pi}$ that are to be differentiated. Differentiation under the integral sign is invalid under these conditions and actually produces severely biased results. We will now discuss this issue in-depth, including the approaches taken by previous work. Later in Section 4.4, we will combine these ideas with the RB algorithm to also enable support for discontinuities in this framework.

4.1.3 Geometric discontinuities

In order to study the issue of discontinuities, we return to a more general setting where we differentiate an integral of a function f over the domain $\mathcal{X}$:

$$\partial_\pi I = \partial_\pi \int_\mathcal{X} f(\mathbf{x}, \pi)\, \mathrm{d}\mu(\mathbf{x}). \tag{4.16}$$

For conceptual and notational simplicity, we will from now on simply look at the derivative with respect to a single parameter π, even though our final algorithms will evaluate *all* derivatives at once.

When computing gradients, we might be inclined to simply exchange the order of integration and differentiation

$$\partial_\pi \int_\mathcal{X} f(\mathbf{x}, \pi)\, \mathrm{d}\mu(\mathbf{x}) \overset{?}{=} \int_\mathcal{X} \partial_\pi f(\mathbf{x}, \pi)\, \mathrm{d}\mu(\mathbf{x}). \tag{4.17}$$

In case the integrand f is discontinuous, this only computes an incomplete version of the desired gradients. To be more precise, the presence of discontinuities is not problematic in itself: bias arises only due to the dependence of their position on the parameter π,

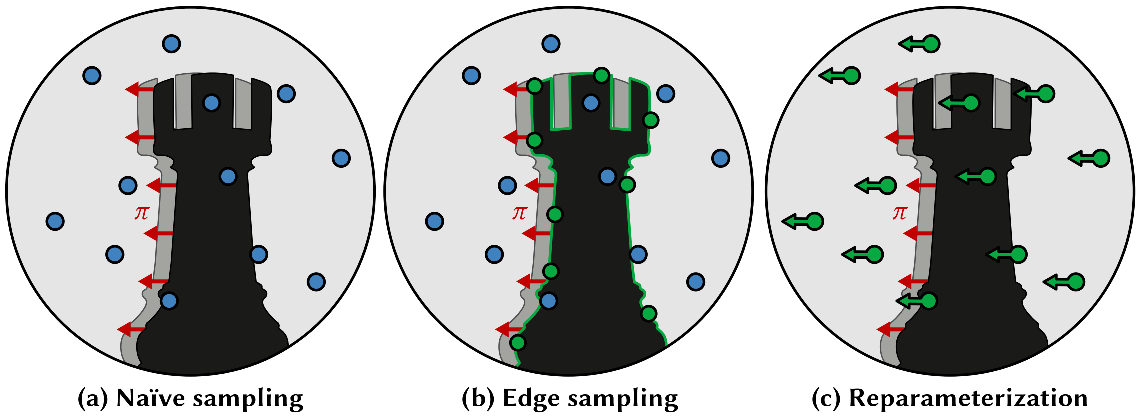


Figure 4.6: We consider integration of a function over $\mathcal{S}^2$ where infinitesimal movement of geometry causes a parameter dependent discontinuity. **(a)** The function value at sample locations is discontinuous and Monte Carlo integration does not capture the gradient due to the moving geometry. **(b)** Edge sampling [165] performs explicit integration over the discontinuity boundaries by placing samples on object silhouettes. **(c)** Reparameterization [156] re-writes the integral in order to freeze discontinuities in place. This can also be viewed as making the sample locations follow along with the discontinuity boundaries.

as pointed out by Loubet et al. [156]. In differentiable rendering, this case is extremely common however. Examples of such parameters include camera pose, object transformations, vertex positions, etc. The problem is also readily apparent when considering that we use Monte Carlo point samples in order to approximate these integrals. Discontinuities in the integrand lie on curves with probability measure zero and are therefore not sampled, see Figure 4.6 (a).

A full solution to Equation (4.16) is found by applying the *Reynolds transport theorem* [163, 171], which introduces an additional boundary correction term that integrates over the parameter dependent edges $\partial\mathcal{X}(\pi)$ explicitly:

$$\partial_\pi I = \int_{\mathcal{X}} \partial_\pi f(\mathbf{x}, \pi)\, \mathrm{d}\mu(\mathbf{x}) + \oint_{\partial\mathcal{X}(\pi)} f(\mathbf{x}, \pi)\, \langle \partial_\pi \mathbf{x}, \hat{\mathbf{n}} \rangle\, \mathrm{d}\mu(\mathbf{x}). \tag{4.18}$$

Here, $\hat{\mathbf{n}}$ denotes the normal direction of the edge at $\mathbf{x} \in \partial\mathcal{X}(\pi)$. Li et al. [165] were the first to propose a solution based on this expression by importance sampling the set of silhouette edges observed from a given scene location, see Figure 4.6 (b). However, existing data structures for this sampling step exhibit poor scaling as the geometric complexity of a scene grows. Zhang et al. [163] revisit this in a higher-dimensional path space, which provides access to additional edge sampling strategies.

An alternative approach was proposed by Loubet et al. [156]. They apply a *change of variables* based on a bijective *(re-)*parameterization $R : \mathcal{X} \times \Pi \to \mathcal{X}$ of the domain $\mathcal{X}$ that freezes discontinuities in place. Note that it must necessarily depend on the scene parameters to accomplish this task and it also introduces a Jacobian determinant $|J_R|$ that ensures the integral result is unchanged, by accounting for any expansion or contraction of the domain. Following this change, the partial derivative can then safely be moved into the integral:

$$\partial_\pi I = \int_{\mathcal{X}} \partial_\pi \big[f(R(\mathbf{x}, \pi), \pi)\ |J_R(\mathbf{x}, \pi)| \big] \,\mathrm{d}\mu(\mathbf{x}). \tag{4.19}$$

Since the boundary correction is no longer needed, this integration only involves standard interior estimators. An alternative interpretation of this is that Monte Carlo samples of Equation (4.19) now have a new dependency on π and follow the discontinuity boundaries with changes of π, see Figure 4.6 (c). The specific change of variables proposed by Loubet et al. is approximate, however: it only rotates the spherical domain and cannot counteract all silhouette motion unless it perfectly matches a spherical rotation as well.

Recently, Bangaru et al. [73] observed that the divergence theorem can be applied to Equation (4.18), turning the troublesome boundary integral into another interior integral

$$\partial_\pi I = \int_{\mathcal{X}} \partial_\pi f(\mathbf{x}, \pi) \,\mathrm{d}\mu(\mathbf{x}) + \int_{\mathcal{X}} \nabla_{\mathbf{x}} \cdot \big(f(\mathbf{x}, \pi)\ V(\mathbf{x}, \pi)\big) \,\mathrm{d}\mu(\mathbf{x}), \tag{4.20}$$

where the *warp field* $V(\mathbf{x}, \pi)$ smoothly interpolates the boundary velocity $\partial_\pi \mathbf{x}$ from Equation (4.18). This formulation is ultimately shown to be equivalent to the change of variables approach of Loubet et al. In particular, the new divergence term directly corresponds to the derivative of the Jacobian in Equation (4.19), and there is a one-to-one correspondence between warp fields and parameterizations of integrals. An important contribution of Bangaru et al. [73] is the construction of a warp field that smoothly tends to the correct velocity as one approaches a boundary. In the formulation using reparameterization, this can be interpreted as correctly counteracting all boundary motion.

In Section 4.4 we will show how to combine such an approach with the radiative backpropagation algorithm. For this reason, we will first review the relevant details more closely, in particular, how the improved reparameterization of Bangaru et al. is defined.

Spherical reparameterization. Both prior works [73, 156] consider integrals over the unit sphere $\mathcal{X} = \mathcal{S}^2$:

$$\partial_\pi \int_{\mathcal{S}^2} f(\boldsymbol{\omega}, \pi) \,\mathrm{d}\Omega(\boldsymbol{\omega}) = \partial_\pi \int_{\mathcal{S}^2} f(R(\boldsymbol{\omega}, \pi), \pi) \cdot |J_R(\boldsymbol{\omega}, \pi)| \,\mathrm{d}\Omega(\boldsymbol{\omega}). \tag{4.21}$$

In this case, $|J_R|$ denotes the Jacobian determinant that counteracts the change in spherical area due to the mapping R. Similarly to Loubet et al., R is chosen such that it satisfies

$R(\boldsymbol{\omega}, \pi_0) = \boldsymbol{\omega}$, where π_0 is a variable matching the (primal) value of π, but that does not participate in the differentiation. As a result, the change of variables does not affect the primal integral and only changes the gradient computation.

Furthermore, $\partial_\pi R(\boldsymbol{\omega}, \pi) = \partial_\pi P(\boldsymbol{\omega}, \pi)$ where $P : \mathcal{S}^2 \times \Pi \rightarrow \mathcal{S}^2$ returns a direction, whose *velocity* $\partial_\pi P$ must be carefully chosen such that the parameterization can accomplish its goal of freezing discontinuities in place. The equations above are easily satisfied if R is defined as[3]

$$R(\boldsymbol{\omega}, \pi) = \boldsymbol{\omega} + P(\boldsymbol{\omega}, \pi) - P(\boldsymbol{\omega}, \pi_0). \tag{4.22}$$

However, a suitable function P must still be chosen, following two critical requirements:

1. As $\boldsymbol{\omega}$ approaches another direction $\boldsymbol{\omega}_b(\pi)$ that is located on a π-dependent discontinuity, the velocity $\partial_\pi P(\boldsymbol{\omega}, \pi)$ *must* tend to $\partial_\pi \boldsymbol{\omega}_b(\pi)$.

2. P and its derivative $\partial_\pi P$ must themselves be smooth functions, i.e. they should not introduce any new discontinuities that would again cause problems when differentiating.

The first condition is fulfilled with a preliminary function $\tilde{P}(\boldsymbol{\omega}, \pi)$. Its primal evaluation is again not particularly interesting and simply returns the unaltered direction $\tilde{P}(\boldsymbol{\omega}, \pi_0) = \boldsymbol{\omega}$. However, the direction is computed as part of a modified ray tracing operation that returns a hit point $\tilde{\mathbf{x}}$ whose π-derivative captures the motion of the intersected object, see Figure 4.7.

Note that the derivatives of this suggested direction $\tilde{P}$ are only continuous on the *interior* of intersected shapes (Figure 4.8 (c)) and satisfying the smoothness condition is more challenging, especially given that we do not have knowledge about the position of the discontinuities in the integration domain $\mathcal{S}^2$. Bangaru et al. therefore define P in terms of a spherical convolution of the base direction $\tilde{P}$ and a weighting kernel w requiring normalization through an additional integral in the denominator:

$$P(\boldsymbol{\omega}, \pi) = \frac{\int_{\mathcal{S}^2} w(\boldsymbol{\omega}, \boldsymbol{\omega}')\, \tilde{P}(\boldsymbol{\omega}', \pi)\, \mathrm{d}\Omega(\boldsymbol{\omega}')}{\int_{\mathcal{S}^2} w(\boldsymbol{\omega}, \boldsymbol{\omega}')\, \mathrm{d}\Omega(\boldsymbol{\omega}')}. \tag{4.23}$$

The choice of the weights w is crucial here. For instance, a simple Gaussian kernel ensures smoothness, but the derivatives would no longer match the boundary motion

[3]Note that some operations involving the parameterization (like Equation (4.22) or evaluating Jacobian determinants) implicitly assume that the underlying spherical domain is accessed using suitable 2-dimensional coordinates (e.g. spherical coordinates), as the ambient 3D space is too high-dimensional.

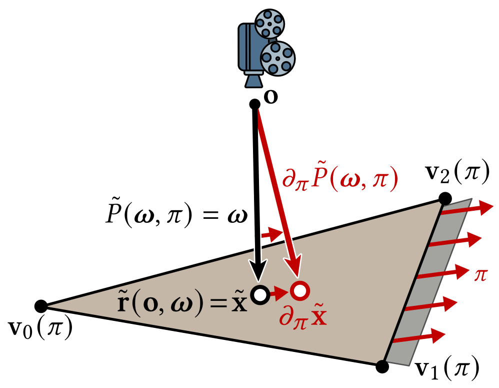

$\mathbf{o}$
$\partial_\pi \tilde{P}(\boldsymbol{\omega}, \pi)$
$\mathbf{v}_2(\pi)$
$\tilde{P}(\boldsymbol{\omega}, \pi) = \boldsymbol{\omega}$
π
$\tilde{\mathbf{r}}(\mathbf{o}, \boldsymbol{\omega}) = \tilde{\mathbf{x}}$
$\partial_\pi \tilde{\mathbf{x}}$
$\mathbf{v}_0(\pi)$
$\mathbf{v}_1(\pi)$

Figure 4.7: A specialized ray tracing routine $\tilde{\mathbf{r}}$ returns a hit point $\tilde{\mathbf{x}}$ whose derivative follows the movement of the intersected shape due to infinitesimal changes of π. For the case of a common ray-triangle intersection, this involves recomputing the position based on barycentric interpolation of the π-dependent vertices $\mathbf{v}_i(\pi)$, but where the barycentric coordinates are used as non-differentiated constants. $\tilde{P}$ is then the normalized direction computed between $\tilde{\mathbf{x}}$ and the ray origin $\mathbf{o}$.

(Figure 4.8 (d)). Bangaru et al. propose a new *harmonic* convolution weight

$$w(\boldsymbol{\omega}, \boldsymbol{\omega}') = \frac{1}{\tilde{B}(\boldsymbol{\omega}') + e^{\kappa(1-\boldsymbol{\omega}\cdot\boldsymbol{\omega}')} - 1} \tag{4.24}$$

involving a spherical von Mises-Fisher distance term with *concentration* κ and a *boundary test* $\tilde{B}$ that queries *approximate* distances to the visible edges. Specifically, $\tilde{B}(\boldsymbol{\omega})$ needs to approach zero as $\boldsymbol{\omega}$ moves towards a discontinuity $\boldsymbol{\omega}_b$. This can, e.g., be accomplished based on a dot product with the shading normal or the distance to triangle edges for flat shaded geometry [73].

The resulting weights become extremely large as $\boldsymbol{\omega}$ and $\boldsymbol{\omega}'$ approach a silhouette. This furthermore occurs in a “unidirectional” fashion to ensure that the final result $P(\boldsymbol{\omega}, \pi)$ follows the motion of the occluder as $\boldsymbol{\omega} \to \boldsymbol{\omega}_b$.

Both integrals in Equation (4.23), are evaluated using Monte Carlo sampling and must use the same set of random samples to reduce variance to an acceptable level. Bangaru et al. also employ antithetic sampling (Section 2.4.5) and a control variate approach (Section 2.4.6) to reduce variance further. Note that the division by an estimate here generally produces a consistent, but biased estimator. Bangaru et al. also describe a de-biasing scheme based on Russian Roulette.

A summary of this harmonic convolution procedure is illustrated in Figure 4.8 in a simplified 2D setting. We refer to the original paper [73] for further details.

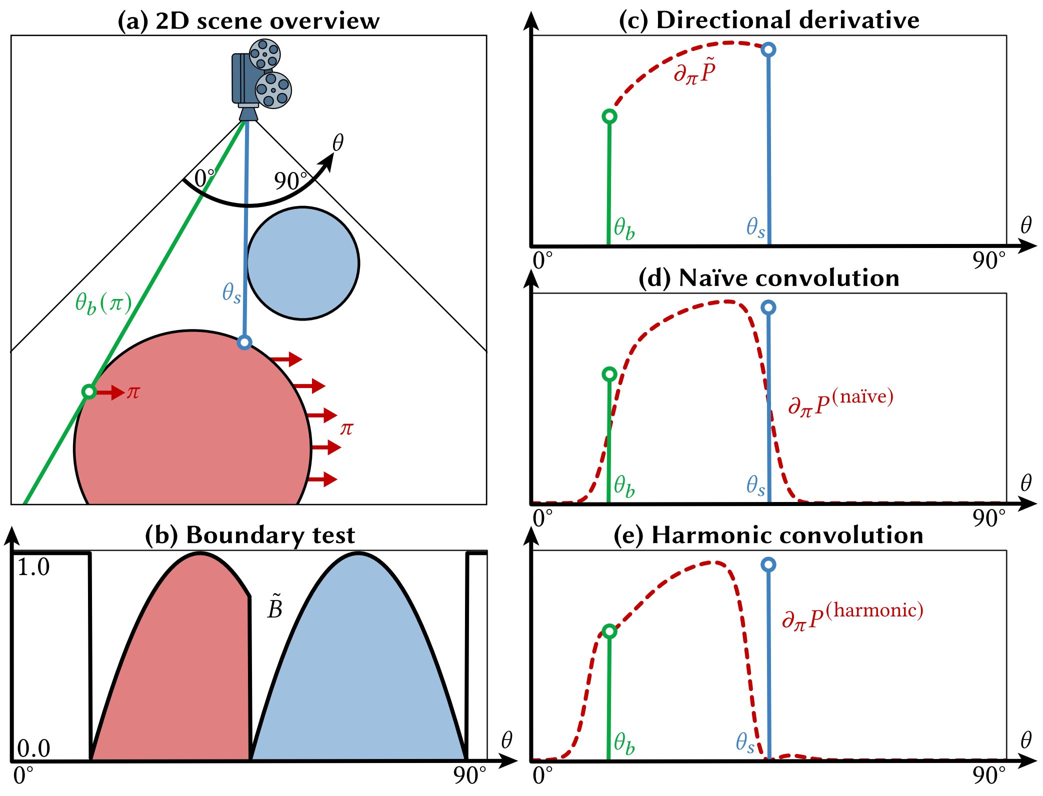


Figure 4.8: **(a)** The red circle in a 2D scene translates due to parameter π, causing a discontinuity $\theta_b(\pi)$. Subsequent plots show functions over the 2D angle θ of the camera's field of view. **(b)** The boundary test approaches zero at silhouettes of intersected geometry. **(c)** Direction $\tilde{P}$ captures the directional motion of rays hitting the geometry, but is discontinuous across silhouettes. **(d)** Naïve convolution with a Gaussian weight produces a smooth result but does not match the silhouette velocity. **(e)** The harmonic convolution satisfies both properties. Note that the discontinuity θ_s caused by the blue sphere is unproblematic as it remains *static* with respect to π.

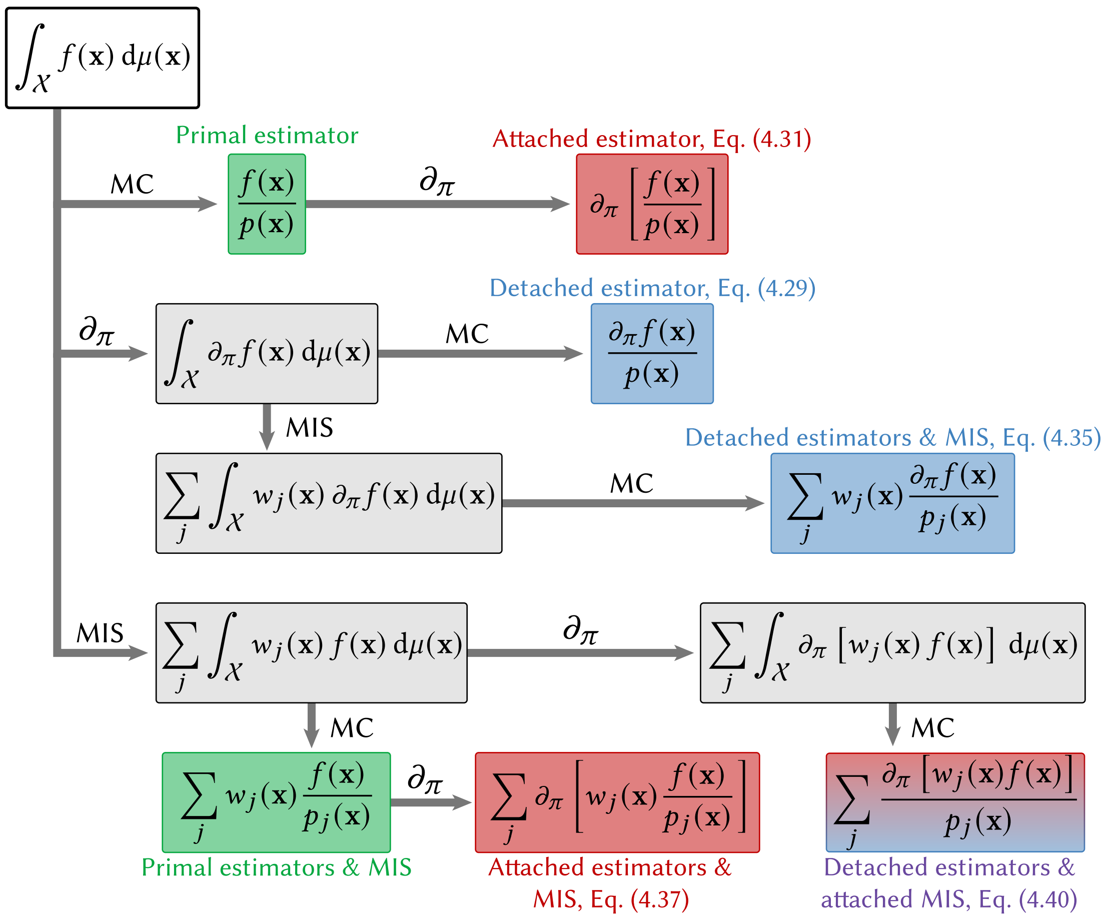

$\int_{\mathcal{X}} f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})$
MC
Primal estimator
$\frac{f(\mathbf{x})}{p(\mathbf{x})}$
∂_π
Attached estimator, Eq. (4.31)
$\partial_\pi\left[\frac{f(\mathbf{x})}{p(\mathbf{x})}\right]$
∂_π
$\int_{\mathcal{X}} \partial_\pi f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})$
MC
Detached estimator, Eq. (4.29)
$\frac{\partial_\pi f(\mathbf{x})}{p(\mathbf{x})}$
MIS
$\sum_j \int_{\mathcal{X}} w_j(\mathbf{x})\,\partial_\pi f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})$
MC
Detached estimators & MIS, Eq. (4.35)
$\sum_j w_j(\mathbf{x})\frac{\partial_\pi f(\mathbf{x})}{p_j(\mathbf{x})}$
MIS
$\sum_j \int_{\mathcal{X}} w_j(\mathbf{x})\,f(\mathbf{x})\,\mathrm{d}\mu(\mathbf{x})$
∂_π
$\sum_j \int_{\mathcal{X}} \partial_\pi\left[w_j(\mathbf{x})\,f(\mathbf{x})\right]\,\mathrm{d}\mu(\mathbf{x})$
MC
MC
$\sum_j w_j(\mathbf{x})\frac{f(\mathbf{x})}{p_j(\mathbf{x})}$
∂_π
$\sum_j \partial_\pi\left[w_j(\mathbf{x})\frac{f(\mathbf{x})}{p_j(\mathbf{x})}\right]$
$\sum_j \frac{\partial_\pi\left[w_j(\mathbf{x})f(\mathbf{x})\right]}{p_j(\mathbf{x})}$
Primal estimators & MIS
Attached estimators & MIS, Eq. (4.37)
Detached estimators & attached MIS, Eq. (4.40)

Figure 4.9: A taxonomy of differential estimators. We illustrate key operations that can be applied to a primal integral (top left). These include Monte Carlo importance sampling, multiple importance sampling, and differentiation. Non-commutativity of these operations leads to a plethora of differential estimators that we study in this chapter. We omit the explicit dependence of f and p on π for brevity. Equation numbers refer to the corresponding locations in the text.

4.2 Differential estimators

In this section, we build on top of the previous summary of Monte Carlo methods in Section 2.4 and investigate the case of differentiated integrals. We introduce several estimators illustrated by the taxonomy in Figure 4.9. We analyze their properties and correctness, assuming for now that the underlying integrals are free of discontinuities. Section 4.4 will revisit the discontinuous case and discuss interactions that arise due to the choices made here.

4.2.1 Detached sampling strategies

We begin with the most basic case that we refer to as the *detached strategy* for reasons that will become clear shortly when we contrast it to *attached* strategies. This approach corresponds to how one would ordinarily differentiate an integral with pencil and paper, i.e., without focusing on its eventual numerical evaluation. We simply move the partial derivative inside and differentiate under the integral sign:

$$\partial_\pi I = \partial_\pi \left[\int_{\mathcal{X}} f(\mathbf{x}, \pi) \, \mathrm{d}\mu(\mathbf{x}) \right] = \int_{\mathcal{X}} \partial_\pi f(\mathbf{x}, \pi) \, \mathrm{d}\mu(\mathbf{x}). \tag{4.25}$$

This transformation is legal if the integral is free of discontinuities. It also holds when any present discontinuities are *static*, i.e., independent of the parameter π being differentiated.

Importance sampling. As transport integrals in computer graphics are almost exclusively evaluated using Monte Carlo estimators based on importance sampling, we must therefore understand how this interacts with differentiation. We focus on the inverse transform sampling previously discussed in Section 2.4.3. This involves a diffeomorphism $T\colon \mathcal{U} \to \mathcal{X}$ that parameterizes the target domain $\mathcal{X}$ by the unit-hypercube of matching dimension. The mapping $\mathbf{x} = T(\mathbf{u})$ is constructed from a target density $p(\mathbf{x})$ so that its Jacobian determinant satisfies $|J_T(\mathbf{u})| = p(\mathbf{x})^{-1}$. The reparameterized primal integral then takes the form

$$I = \int_{\mathcal{U}} f(T(\mathbf{u})) \, |J_T(\mathbf{u})| \, \mathrm{d}\mu(\mathbf{u}) = \int_{\mathcal{U}} \frac{f(T(\mathbf{u}))}{p(T(\mathbf{u}))} \, \mathrm{d}\mu(\mathbf{u}). \tag{4.26}$$

If $p(\mathbf{x})$ is roughly proportional to $f(\mathbf{x})$, the new integrand is nearly constant, in which case its estimates are characterized by low variance. The integral can also be turned into a Monte Carlo estimator by sampling points $\mathbf{u}$ uniformly at random in $\mathcal{U}$, and dropping the integral.

With this notation established, let us now return to differential estimators. The change of variables in Equation (4.26) can be straightforwardly applied to a differential integral $\int_{\mathcal{X}} \partial_\pi f(\mathbf{x}) \, \mathrm{d}\mu(\mathbf{x})$:

$$\partial_\pi I = \int_{\mathcal{U}} \frac{\partial_\pi f(T(\mathbf{u}))}{p(T(\mathbf{u}))} \, \mathrm{d}\mu(\mathbf{u}). \tag{4.27}$$

In the case of rendering, the integrand f depends on the scene parameter $\pi \in \Pi$, and in primal estimators the density p and sampling technique T will generally also share this dependence to enable efficient scene-adaptive importance sampling. In the differential

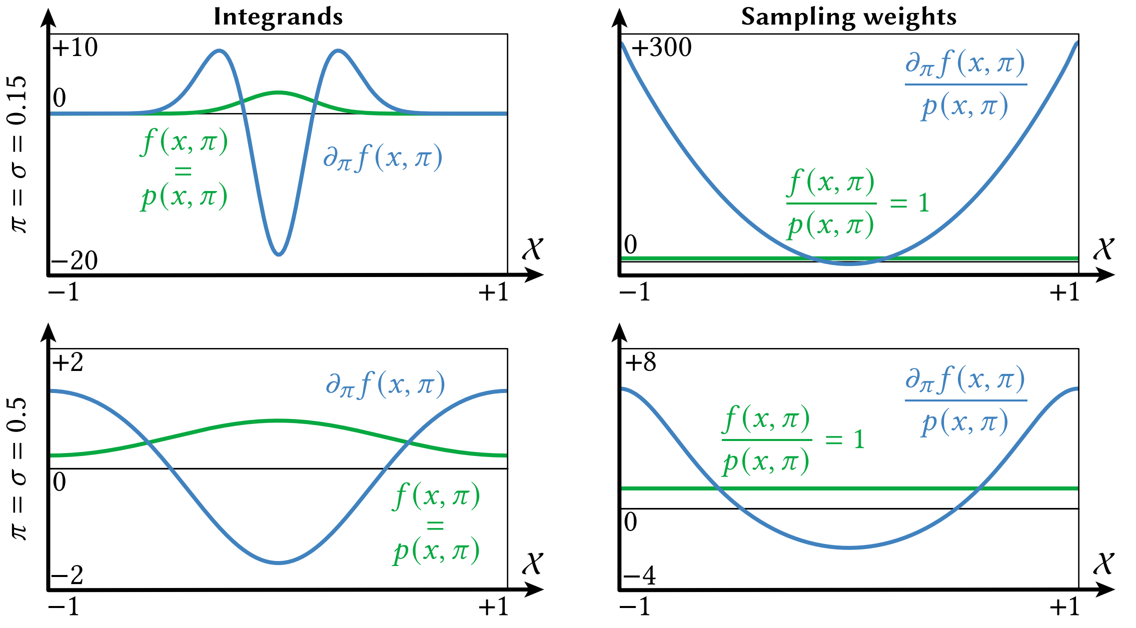


Figure 4.10: 1D examples of *detached* samplers. **Top left**: f (green) follows a wrapped normal distribution parameterized by standard deviation $\pi = \sigma$. Its derivative $\partial_\pi f$ (blue) has a markedly different shape. **Top right**: The sampling density $p = f$ yields a zero-variance primal estimator $f/p = 1$ (green), while the detached estimator of the derivative $\partial_\pi f/p$ (blue) produces large sample weights. **Bottom row**: The same experiment with a different value of π. The problem is less pronounced as $\partial_\pi f$ and p become more uniform.

setting, prior work [156] has handled this dependence by introducing another conceptual parameter variable π_0, whose value happens to match π, but is not part of the differentiation. In this case, both the transform and density can depend on π_0 to benefit from specialized primal sampling strategies:

$$\partial_\pi I = \int_{\mathcal{U}} \frac{\partial_\pi f(T(\mathbf{u}, \pi_0), \pi)}{p(T(\mathbf{u}, \pi_0), \pi_0)} \, \mathrm{d}\mu(\mathbf{u}). \tag{4.28}$$

This finally gives us the expression of the detached estimator

$$\langle \partial_\pi I \rangle^{(\text{detach})} = \frac{\partial_\pi f(\mathbf{x}, \pi)}{p(\mathbf{x}, \pi_0)}, \text{ with } \mathbf{x} = T(\mathbf{u}, \pi_0). \tag{4.29}$$

While not considering part of the expression during differentiation may intuitively appear incorrect, Equation (4.29) remains a valid estimator as long as the primal strategy samples all positions where $\partial_\pi f \neq 0$ with nonzero probability. This requirement may be violated in practice and requires special precautions in differential rendering algorithms, e.g., by ensuring a minimum density even in zero-valued regions of the integrand. An example where this would be necessary is a spatially varying emitter with zero-valued regions that can potentially be “turned on” by the optimization process.

It is important to realize that a high-quality primal sampling strategy with $p \approx f$ is not necessarily also a good choice for the differential estimator of $\partial_\pi f$, as illustrated in Figure 4.10. As with standard (i.e., non-differential) Monte Carlo estimators, the effectiveness of a strategy depends on how well its sampling density p matches the integrand $\partial_\pi f$. Additionally, there is a potential for high variance due to sign related changes in the estimate because $\partial_\pi f$ also contains negative regions.

Detached sampling strategies also cannot be used when the integrand contains a Dirac delta function, which collapses the integration domain.

4.2.2 Attached sampling strategies

Many widely used sampling strategies depend on scene parameters. Examples from the context of physically based rendering include:

1. Sampling of directionally peaked distributions like microfacet models that depend on a roughness parameter.
2. Directional sampling of environment maps proportionally to their textured intensity.
3. Any BSDF sampling method that has an implicit dependence on the local frame which is computed from the shading normals (and ultimately, the surface positions).

In this case, the generated samples conceptually move when we perturb the associated scene parameter π. The previously discussed strategy discarded these effects and was thus *detached* from this motion, motivating its name. We now turn to *attached* strategies that do account for the additional dependence. With this change, everything including the function f, the transformation T, and the division by the probability p are jointly differentiated

$$\partial_\pi I = \int_{\mathcal{U}} \partial_\pi \left[\frac{f(T(\mathbf{u}, \pi), \pi)}{p(T(\mathbf{u}, \pi), \pi)} \right] \mathrm{d}\mu(\mathbf{u}), \tag{4.30}$$

which leads to the attached estimator

$$\langle \partial_\pi I \rangle^{(\text{attach})} = \partial_\pi \left[\frac{f(T(\mathbf{u}, \pi), \pi)}{p(T(\mathbf{u}, \pi), \pi)} \right]. \tag{4.31}$$

Consequently, attached sampling is also possible across more than one scattering event. In that case, the integration domain $\mathcal{U}$ and its counterpart $\mathcal{X}$ simply have higher dimension.

It is interesting to note that the attached strategy will usually be produced *by default* when Monte Carlo sampling code is transparently differentiated using techniques for automatic differentiation.

Let us briefly consider the setting where $\dim \mathcal{U} = \dim \mathcal{X} = 1$ to better understand Equation (4.30). Applying the above derivatives then yields

$$\begin{aligned}\partial_\pi I = \int_{\mathcal{U}} \frac{1}{p(x,\pi)^2} \cdot \big[& p(x,\pi)\,(\partial_\pi f(x,\pi) + \partial_\pi T(u,\pi)\,\partial_x f(x,\pi)) \\ & - f(x,\pi)\,(\partial_\pi p(x,\pi) + \partial_\pi T(u,\pi)\,\partial_x p(x,\pi))\big]\,\mathrm{d}u. \end{aligned} \tag{4.32}$$

where $T(u,\pi)$ has been replaced by x for readability. What can we learn from this expression? When f and p are roughly proportional, then so are their derivatives $\partial_x f$, $\partial_\pi f$, $\partial_x p$, and $\partial_\pi p$. In this case, symmetries in the expression within square brackets cause it to be close to zero, which means that the differentiated sampling technique remains a good choice for the differential estimator.

Attached sampling naturally handles integrands containing delta functions, which are not supported by most previous work on differentiable rendering. This case arises, e.g., when computing gradients with respect to the surface normal of a mirror. In this situation, the product of (delta) BSDF and incident radiance simplifies to just the radiance term that will then be evaluated through the mapping T. This addresses a severe limitation of detached sampling techniques.

While that is all excellent news, there are also multiple potential pitfalls involving this type of estimator.

Product integrals. Things becomes more complicated when the integrand is actually a product of complex terms $f = g \cdot h$, of which only one is targeted by the sampling strategy. Suppose that the sampling density p is perfectly proportional to the first term, i.e., $p = C \cdot g$ for $C \in \mathbb{R}$. Then the attached differential estimator reduces to

$$\partial_\pi I = C^{-1} \int_{\mathcal{U}} \big[\partial_\pi h(x,\pi) + \partial_\pi T(u,\pi)\,\partial_x h(x,\pi)\big]\,\mathrm{d}u. \tag{4.33}$$

This expression indicates that two properties carry over from the primal case: g is handled perfectly in the sense that no variance will arise from this term, and integration of the second term h proceeds through the parameterization $T(u,\pi)$.

In contrast to the primal case, an additional term captures the differential change of the reparameterized function $h(T(..))$. It has the potential to introduce significant variance when the parameterization T rapidly distorts h for small perturbations of π. This additional complication does not exist in the primal case. It would be tempting to mix

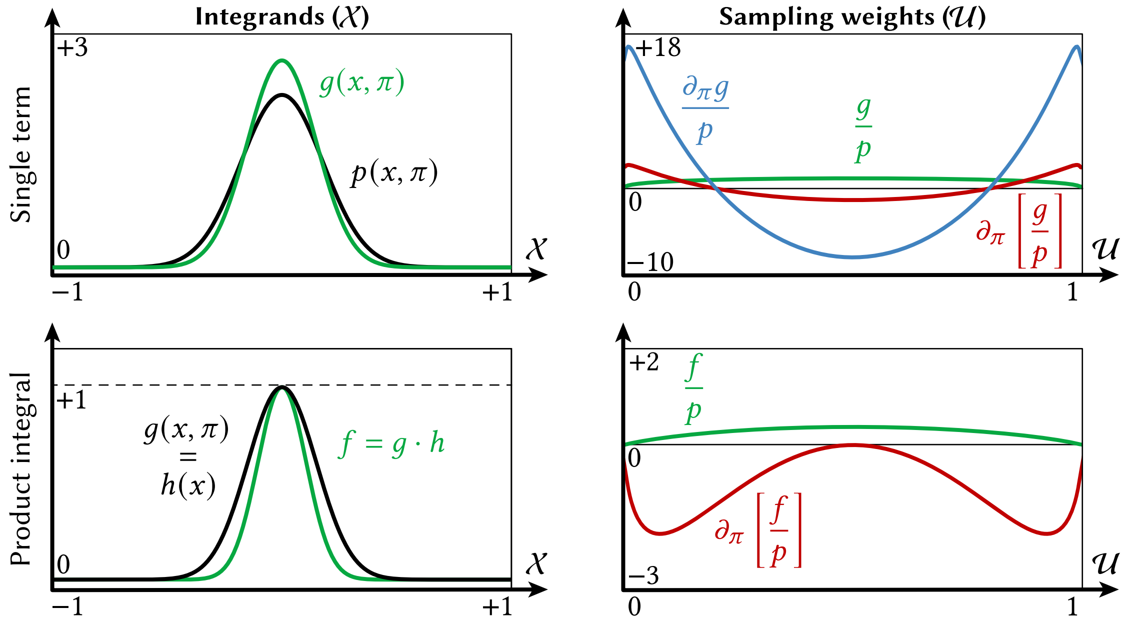


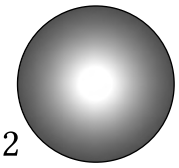
Figure 4.11: 1D Examples of *attached* samplers. **Top row**: g (green) follows a normal distribution parameterized by standard deviation $\pi = \sigma$ and is sampled with density p (black) approximating g. Variance due to attached sample weights (red) in $\mathcal{U}$ resembles the primal case (green), while a detached estimator (blue) performs substantially worse. **Bottom row**: Product integral of two scaled Gaussians $g = h$ sampled proportionally ($p \sim g$), where h is independent of π. Extra derivative terms involving the non-sampled factor in Equation (4.33) inject additional variance, causing the attached estimator (red) to perform poorly compared to an estimator of the primal product integral (green).

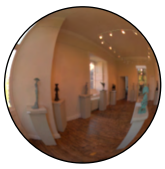
and match, i.e., to attach the factor being sampled and detach the other term to avoid this additional source of variance. However, this generally introduces bias[4]. Figure 4.11 illustrates the difference between complete sampling of an integral and partial sampling based on a factor.

Unfortunately, almost all integrals we want to solve in rendering are product integrals, e.g., the measurement (2.47) or rendering (2.50) equations. As a consequence, the effectiveness of attached strategies is often suboptimal. Let us also look at this for an actual example from a rendering context. We consider the product integral between a glossy microfacet BSDF f_s and the incident radiance L_i defined as an environment map texture. We assume that we can perfectly importance sample the BSDF term ($p(\boldsymbol{\omega},\pi) = C \cdot f_s(\boldsymbol{\omega},\pi)$) and want to estimate the derivative of the integral after differ-

[4]A product integral $\partial_\pi\left[\int_0^1 g(x,\pi)\cdot h(x,\pi)\,\mathrm{d}x\right]$ with $g(x,\pi) = h(x,\pi) = x^\pi$ and $p(x,\pi) \sim x^\pi$ provides a simple example of this: all possible ways of attaching and detaching the terms and reciprocal probability lead to different derivative estimates for $\pi_0 = 1$.

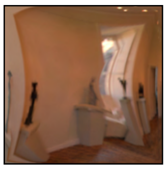
entiating the BSDF roughness parameter:





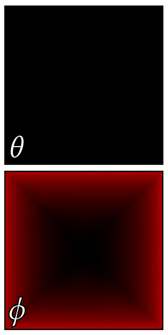
$$I = \int_{S^2} \underbrace{\qquad}_{f_s(\boldsymbol{\omega}, \pi)} \cdot \underbrace{\qquad}_{L_\mathrm{i}(\boldsymbol{\omega})} \mathrm{d}\Omega_{\mathbf{x}}^{\perp}(\boldsymbol{\omega})$$

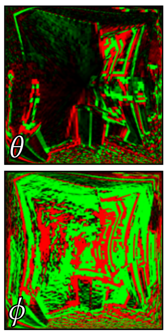
BSDF and probability density factors thus cancel out when we rewrite this as an integral over random numbers $\mathcal{U}$ that are used in the associated sampling routine $T(\mathbf{u}, \pi)$:



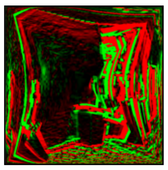
$$I = \frac{1}{C} \int_{\mathcal{U}} \underbrace{\qquad}_{L_\mathrm{i}(T(\mathbf{u}, \pi))} \mathrm{d}\mu(\mathbf{u})$$

Taking the derivative following the vector-valued version of Equation (4.33) introduces a dot product between the π-derivative of the sampled direction T, and a directional $\boldsymbol{\omega}$-derivative of the incident radiance[5]:





$$\partial_\pi I = \frac{1}{C} \int_{\mathcal{U}} \underbrace{\begin{pmatrix} \quad \\ \quad \end{pmatrix}}_{\partial_\pi T(\mathbf{u}, \pi)} \cdot \underbrace{\begin{pmatrix} \quad \\ \quad \end{pmatrix}}_{\nabla_\omega L_\mathrm{i}(\boldsymbol{\omega})} \mathrm{d}\mu(\mathbf{u})$$



$$= \frac{1}{C} \int_{\mathcal{U}} \underbrace{\qquad}_{\partial_\pi \, [L_\mathrm{i}(T(\mathbf{u}, \pi)]} \mathrm{d}\mu(\mathbf{u})$$

[5]For the illustration, we parameterize the directions $\boldsymbol{\omega}$ using 2D spherical coordinates (θ, ϕ). It is also valid to work in a 3D coordinate system with normalized vectors $\boldsymbol{\omega}$.

Note that an additional $\partial_\pi L_\mathrm{i}(\boldsymbol{\omega})$ term is omitted in the illustration as it evaluates to zero, i.e., the incident radiance is independent of π when not warped through $T(\mathbf{u}, \pi)$.

As in the 1D example (4.33) earlier, differentiating through the full sampling routine adds considerable complexity to the problem that was not present in the primal integral. The corresponding (attached) Monte Carlo estimator will now place samples randomly in $\mathcal{U}$ to estimate the derivative, causing high variance.

Discrete decisions. Sampling techniques often consume uniform variates to take discrete decisions like choosing the component of a multi-lobe BSDF. In contrast to the unified differentiation of integrand, density, and parameterization in Equation (4.30), the probabilities of such discrete decisions should never be handled in an attached manner, as doing so would introduce severe bias. For example, consider a path termination criterion such as *Russian Roulette* (Section 2.5.2), which only continues a random walk with probability P_R, while applying a scaling correction to account for this change:

$$\mathrm{RR}(u, P_R) = \begin{cases} 1/P_R, & u < P_R, \\ 0, & \text{otherwise.} \end{cases}$$

In practice, the probability P_R would be related to the reflectances of prior scattering interaction, which introduces a dependence on the scene parameters (in the extreme case, $P_R = \pi$). However, this expression then behaves like a parameter-dependent discontinuity that was explicitly forbidden at the beginning of this section. Computations of such probabilities can still depend on scene parameters, but to resolve the issue they should use only its value π_0 without being part of the differentiation process.

Creation of discontinuities. Techniques of the detached type can be used to compute unbiased estimates of discontinuous integrands if the positions of these discontinuities do not depend on scene parameters π. This common case arises, e.g., when optimizing materials on static geometry. The ability to easily solve such problems despite the omnipresent visibility-induced discontinuities is a welcome simplification.

The previous discussion has shown that attached sampling strategies can be superior to detached ones, particularly when the former are built from high-quality primal methods. However, attempting to differentiate discontinuous integrands using such attached strategies reveals a fundamental problem: these methods warp the integrand in a parameter-dependent fashion, and this transformation will naturally also affect discontinuities. Consequently, as shown in Figure 4.12, discontinuities that were previously static on $\mathcal{X}$ will lose this property on the reparameterized domain $\mathcal{U}$, introducing bias.

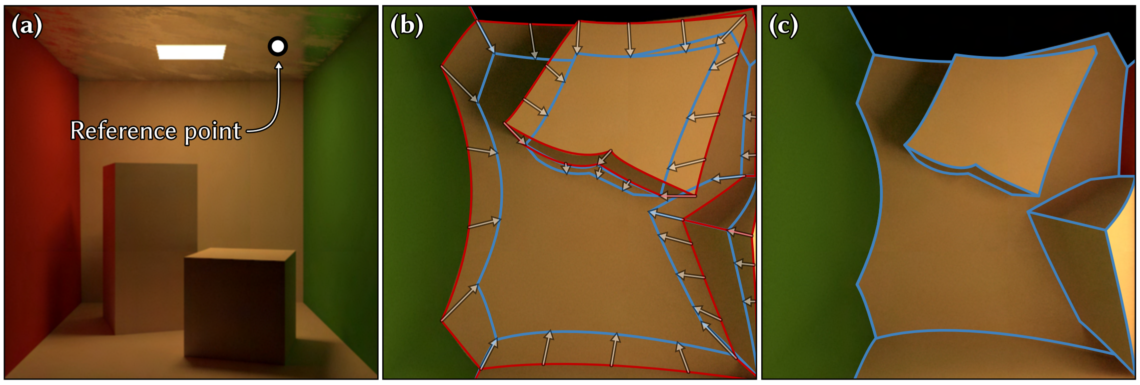


Figure 4.12: The visible hemisphere from a given reference point in a Cornell box scene **(a)** is mapped through a microfacet importance sampling transform onto the unit square $\mathcal{U}$ for two different roughness values $\alpha = 0.4$ **(b)** and $\alpha = 0.6$ **(c)**. Motion vectors (white arrows) indicate how static discontinuities (red and blue lines) become dependent on α through this parameterization. The sampling routine used in this visualization targets the Beckmann microfacet distribution through a concentric disk mapping.

This can also be interpreted as placing parameter dependent samples in $\mathcal{X}$ that will then cross discontinuity curves when applying infinitesimal changes to π.

As-is, such attached techniques simply cannot be used with discontinuous integrands, which rules out most rendering-related applications. Fortunately, it is possible to address this limitation using *reparameterized attached* sampling that we present in Section 4.4 for the important special case of directly visible discontinuities on the unit sphere $\mathcal{X} = \mathcal{S}^2$.

4.2.3 Multiple importance sampling

Given the essential role of multiple importance sampling for variance reduction in forward rendering, we will now also consider MIS in the context of gradient estimators. Analogously to the choice of attaching or detaching the estimators themselves, the same decision must now be taken for their MIS weights, resulting in a 2×2 matrix of possible combinations.

The main benefit of attached strategies lies in their ability to consider the dependence on π for variance reduction. This is ultimately not very useful for MIS weights, where a strong dependence on π represents an unusual situation: this would mean that a perturbation of a scene parameter rapidly changes the sampling technique of choice. While MIS continues to play an important role in combining several strategies, the choice of whether to attach or detach its weights is thus largely irrelevant from the viewpoint of variance reduction. However, not all possible combinations of attached estimators and attached MIS are useful or even correct, and we now review the various possibilities from this viewpoint.

Detached estimators, detached MIS. Suppose that we are already working with detached estimators: in this case, it would be natural to similarly neglect the π-dependence of MIS weights during differentiation:

$$\partial_\pi I = \sum_{j=1}^{M} \int_{\mathcal{U}} w_j(T_j(\mathbf{u}, \pi_0), \pi_0) \cdot \frac{\partial_\pi f(T_j(\mathbf{u}, \pi_0), \pi)}{p_i(T_j(\mathbf{u}, \pi_0), \pi_0)} \, \mathrm{d}\mu(\mathbf{u}), \tag{4.34}$$

where M techniques with sampling transforms T_j and PDFs p_j are combined. The resulting estimator is

$$\langle \partial_\pi I \rangle^{(\text{MIS d./d.})} = \sum_{j=1}^{M} w_j(\mathbf{x}_j, \pi_0) \cdot \frac{\partial_\pi f(\mathbf{x}_j, \pi)}{p_i(\mathbf{x}_j, \pi_0)}, \tag{4.35}$$

where a sample drawn from strategy j is denoted as $\mathbf{x}_j = T_j(\mathbf{u}, \pi_0)$.

This combination is a standard application of MIS to a particular function that happens to be a derivative, and its correctness thus follows from prior work, see Equation (2.98).

Attached estimators, attached MIS. Alternatively, both estimators and MIS weights can be parameterized through corresponding inverse-transform mappings $T_i : \mathcal{U} \to \mathcal{X}$ to track all parameter dependencies during the differentiation process. This case is correct by definition as we are now simply looking at the derivative of the entire expression:

$$\partial_\pi I = \sum_{j=1}^{M} \int_{\mathcal{U}} \partial_\pi \left[w_j(T_j(\mathbf{u}, \pi), \pi) \cdot \frac{f(T_j(\mathbf{u}, \pi), \pi)}{p_j(T_j(\mathbf{u}, \pi), \pi)} \right] \mathrm{d}\mu(\mathbf{u}), \tag{4.36}$$

with final estimator

$$\langle \partial_\pi I \rangle^{(\text{MIS a./a.})} = \sum_{j=1}^{M} \partial_\pi \left[w_j(T_j(\mathbf{u}, \pi), \pi) \cdot \frac{f(T_j(\mathbf{u}, \pi), \pi)}{p_j(T_j(\mathbf{u}, \pi), \pi)} \right]. \tag{4.37}$$

Attached estimators, detached MIS. In our experiments, we also considered a third logical option of combining attached estimators with detached MIS weights:

$$\partial_\pi I \overset{?}{=} \sum_{j=1}^{M} \int_{\mathcal{U}} w_j(T_j(\mathbf{u}, \pi_0), \pi_0) \cdot \partial_\pi \left[\frac{f(T_j(\mathbf{u}, \pi), \pi)}{p_j(T_j(\mathbf{u}, \pi), \pi)} \right] \mathrm{d}\mu(\mathbf{u}). \tag{4.38}$$

However, this combination can be severely biased. Differentiation of the fully attached case in Equation (4.36) via the product rule generates mixed terms of the form $(\partial_\pi w_j) \cdot f/p_j$ that are missing in Equation (4.38), and this introduces bias unless $\partial_\pi w_j = 0$ (which does not represent an interesting case).

Detached estimators, attached MIS. Finally, one can also attach the MIS weights of a set of detached estimators:

$$\partial_\pi I = \sum_{j=1}^{M} \int_{\mathcal{U}} \frac{\partial_\pi \left[w_j(T_j(\mathbf{u}, \pi_0), \pi) \cdot f(T_j(\mathbf{u}, \pi_0), \pi) \right]}{p_j(T_j(\mathbf{u}, \pi_0), \pi_0)} \, \mathrm{d}\mu(\mathbf{u}), \tag{4.39}$$

leading to one more estimator

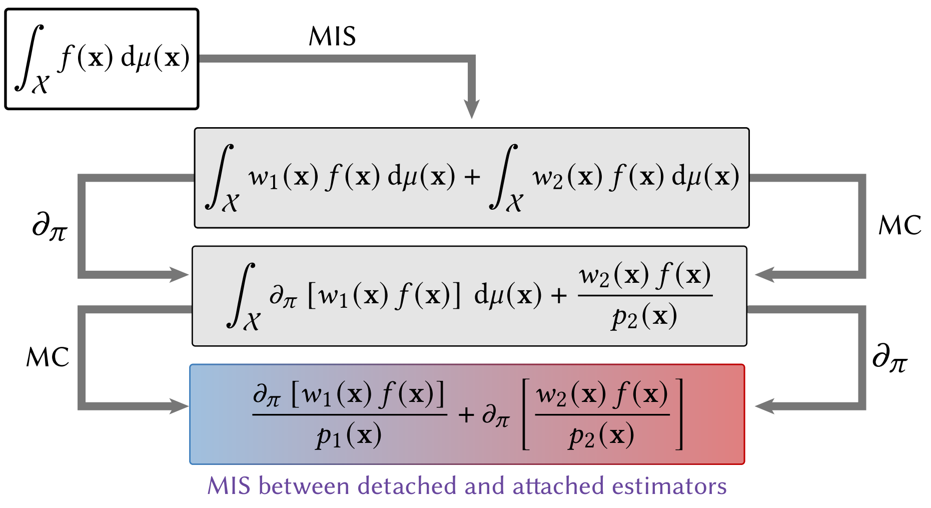
$$\langle \partial_\pi I \rangle^{(\text{MIS d./a.})} = \sum_{j=1}^{M} \frac{\partial_\pi \left[w_j(\mathbf{x}_j, \pi) \cdot f(\mathbf{x}_j, \pi) \right]}{p_j(\mathbf{x}_j, \pi_0)}. \tag{4.40}$$

The validity of this approach follows from the correctness of the individual steps that can be used to derive it: introducing MIS, differentiating, followed by Monte Carlo importance sampling. See also the sequence of steps leading to the bottom right in the taxonomy in Figure 4.9. We mainly mention this case for completeness and have not found it to be a compelling strategy in our experiments using standard MIS weights based on the balance or power heuristic.

Combining attached and detached strategies. The full set of options is even more fine-grained than the above list may suggest: mixing attached and detached estimators is also possible. The validity of this approach once more follows from the correctness of the individual steps, as highlighted in Figure 4.13. In some sense, this is not too surprising, as this type of combination will naturally arise if one of the strategies p_j is independent of the parameter π being differentiated.

Discussion. To summarize, MIS remains a helpful tool for combining sampling strategies. Not all possible combinations of attached MIS weights and estimators are useful or yield unbiased gradient estimates. Based on experimental evaluation, we recommend to either jointly attach or detach estimators and their MIS weights.

A curious thought that arises following this discussion is whether one can combine attached and detached versions of the *same* primal estimator via MIS to draw on each strategy where it performs best? This intuition from primal estimators sadly does not transfer to the differential world: standard MIS weights are guided by sampling probabilities, and those probabilities would be identical in such a combination (modulo minor differences in how differentiation is performed with respect to π). It will be interesting to explore extensions and generalizations of MIS that can perceive the deficiencies of a differential estimator and suitably adapt.



$\int_{\mathcal{X}} f(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x})$
MIS
$\int_{\mathcal{X}} w_1(\mathbf{x}) f(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x}) + \int_{\mathcal{X}} w_2(\mathbf{x}) f(\mathbf{x})\, \mathrm{d}\mu(\mathbf{x})$
∂_π
MC
$\int_{\mathcal{X}} \partial_\pi \left[w_1(\mathbf{x}) f(\mathbf{x})\right] \mathrm{d}\mu(\mathbf{x}) + \frac{w_2(\mathbf{x}) f(\mathbf{x})}{p_2(\mathbf{x})}$
MC
∂_π
$\frac{\partial_\pi \left[w_1(\mathbf{x}) f(\mathbf{x})\right]}{p_1(\mathbf{x})} + \partial_\pi \left[\frac{w_2(\mathbf{x}) f(\mathbf{x})}{p_2(\mathbf{x})}\right]$
MIS between detached and attached estimators

Figure 4.13: The decision of whether to attach or detach a sampling technique and its MIS weight can be made separately for each technique, as illustrated by this derivation sketch where the arrows on both sides refer to the left and right summands respectively.

Note that the variance guarantees normally associated with MIS [66, 67] also no longer hold in the differential setting. Similar to normal importance sampling, the presence of negative integrand regions severely changes the underlying estimation problem, and it is unclear how MIS is affected when the weights themselves undergo differentiation. Investigating new variance bounds for this generalized case would provide valuable additional insight for the estimators discussed in this section.

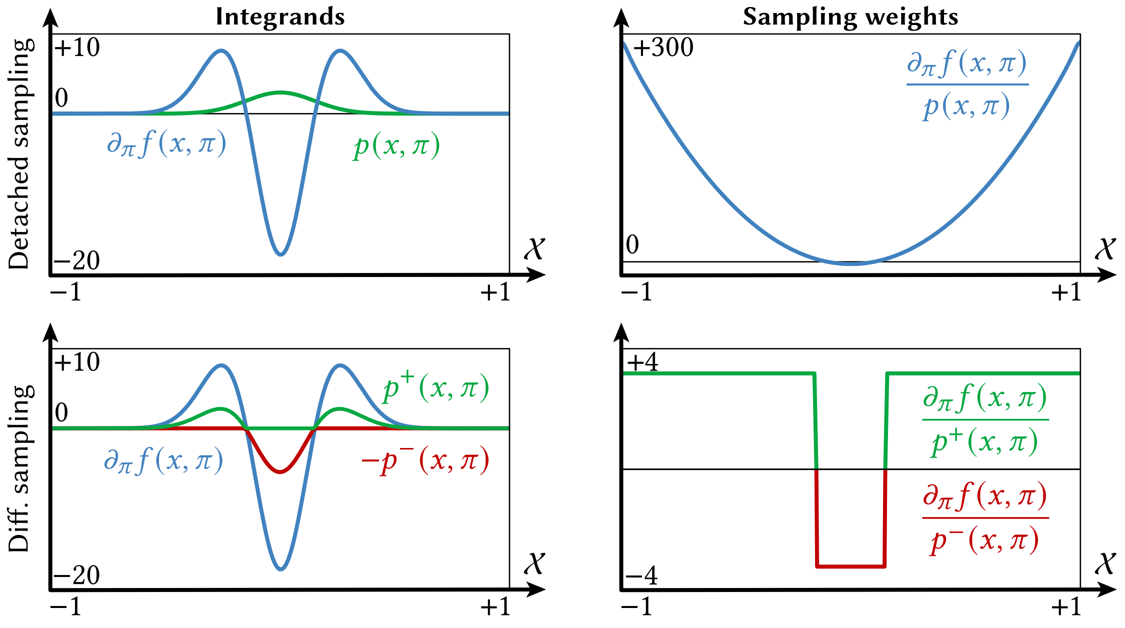


Figure 4.14: 1D examples of custom *differential* strategies. **Top row**: The wrapped normal distribution p is ill-suited for sampling its derivative $\partial_\pi f = \partial_\pi p$ and produces large weights. **Bottom row**: the densities p^+ and p^- are proportional to the positive and negative regions of $\partial_\pi f$ and produce constant sample weights that reduce the variance of the estimator.

4.3 Differential sampling strategies

Section 4.1.2 hinted at a fascinating possibility that arises when a differentiable renderer relies on decoupled primal and adjoint phases as part of radiative backpropagation: we can introduce additional strategies that are specifically designed to improve sampling of differential transport. From a high level, such a *differential sampling strategy* will involve an integral that looks identical to the detached case from Equation (4.28):

$$\partial_\pi I = \int_{\mathcal{U}} \frac{\partial_\pi f(T(\mathbf{u},\pi_0),\pi)}{p(T(\mathbf{u},\pi_0),\pi_0)}\,\mathrm{d}\mu(\mathbf{u}). \tag{4.41}$$

The key difference is that the sampling technique encoded in p and T is no longer constrained by the primal phase. Essentially anything could be used, and we can exploit this freedom to reduce variance in challenging situations. Figure 4.14 shows a simple 1D example of a differential sampling strategy tailored to a wrapped normal distribution.

Differential sampling strategies also address an issue that we had first observed in Equation (4.33), which appears when attached sampling techniques are invariably applied to product integrals that occur in rendering algorithms. Attached strategies will warp all factors, and this introduces additional derivative terms that can introduce significant variance. In contrast, the formulation in Equation (4.41) is static and does not suffer from this problem.

Not all scene parameters call for custom sampling strategies, however. Many material models include a directionally uniform albedo that is adequately handled by primal strategies. In contrast, scene parameters controlling surface roughness have a pronounced effect on the sampling process and constitute an example where differential strategies can make a large difference. This is apparent in many of the results we show later in Section 4.5, where the differential strategy generally performs best. We now discuss an example of a sampling technique targeting the family of microfacet BRDFs.

4.3.1 Differential microfacet sampling

As discussed previously in Section 2.2.5, microfacet distributions are integral building blocks of many widely used reflectance models. In numerical experiments, we found that derivatives with respect to their roughness parameter were characterized by severe variance. Figure 4.10 highlights the fundamental problem: changes in the shape of the integrand break detached estimators. On the other hand, attached estimators applied to a product integral with material and lighting terms tend to perform poorly when the parameter-dependent warp distorts the incident radiance function. We leverage the freedom of a decoupled differential transport simulation to introduce a specialized differential sampling strategy that will address these challenges.

Recall the expression of an (isotropic) microfacet BRDF from Section 2.2.5:

$$f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}, \alpha) = \frac{F(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{h})\, D(\boldsymbol{\omega}_\mathrm{h})\, G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{\omega}_\mathrm{h}, \alpha)}{4\,|\cos\theta_\mathrm{i}| \cdot |\cos\theta_\mathrm{o}|}, \tag{4.42}$$

where the roughness parameter α is included as an explicit argument this time. Like discussed previously, F refers to the Fresnel term (which does not depend on α), D is the microfacet distribution, G is the shadowing-masking term, and $\boldsymbol{\omega}_\mathrm{h}$ denotes the half-direction vector between $\boldsymbol{\omega}_\mathrm{i}$ and $\boldsymbol{\omega}_\mathrm{o}$. Its derivative with respect to α is

$$\partial_\alpha f_r(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i} \to \boldsymbol{\omega}_\mathrm{o}, \alpha) = \frac{F(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{h})}{4\,|\cos\theta_\mathrm{i}| \cdot |\cos\theta_\mathrm{o}|}\, \partial_\alpha \left[D(\boldsymbol{\omega}_\mathrm{h}, \alpha)\, G(\boldsymbol{\omega}_\mathrm{i}, \boldsymbol{\omega}_\mathrm{o}, \boldsymbol{\omega}_\mathrm{h}, \alpha)\right]. \tag{4.43}$$

Here, we ignore the derivative in G as it only has a minor effect on the directional distribution and focus on the microfacet distribution D, limiting our discussion to the two isotropic *Beckmann* [27] and *GGX* [31, 32] models. In spherical coordinates, these two distributions are defined as

$$D_\mathrm{GGX}(\theta, \alpha)\cos\theta = \frac{2\alpha^2 \sin\theta}{\cos^3\theta(\alpha^2 + \tan^2\theta)^2}, \tag{4.44}$$

$$D_\mathrm{Beck.}(\theta, \alpha)\cos\theta = \frac{2e^{\frac{-\tan^2\theta}{\alpha^2}} \sin\theta}{\alpha^2\cos^3\theta}, \tag{4.45}$$

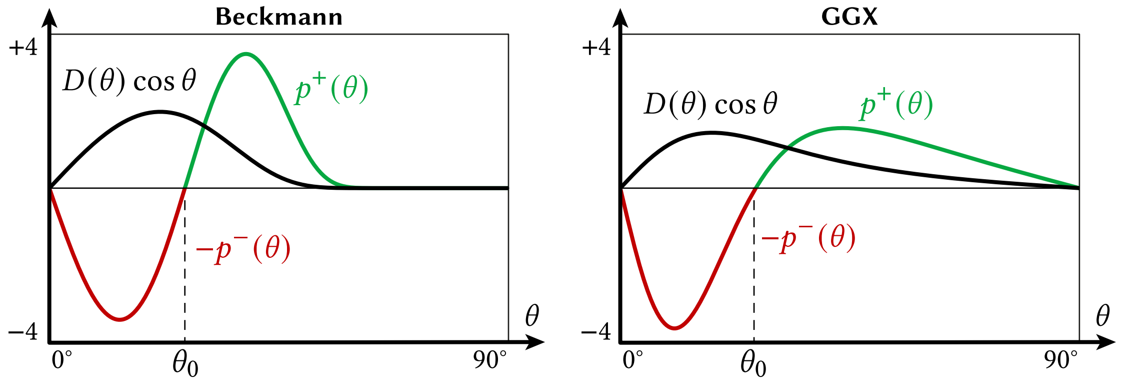


Figure 4.15: Plots of two microfacet distributions (Beckmann and GGX) with roughness parameter $\alpha = 0.5$ (black). The derivative with respect to α produces a signed function with positive and negative lobes of equal mass. Our differential microfacet sampling strategy specifically samples these two lobes using two densities p^+ and p^- (green and red) that are proportional to the absolute value of these derivatives. This enables efficient computation of gradients that characterize how the transport simulation changes with respect to perturbations of α.

where the cosine term on the left side is required for normalization. Following differentiation, the function splits into a positive and negative lobe of equal area (Figure 4.15) with a zero crossing at $\theta_0 = \arctan(\alpha)$. Our goal is to construct a method that samples proportionally to the absolute value of their derivatives

$$\partial_\alpha \left[D_{\text{GGX}}(\theta, \alpha) \cdot \cos\theta \right] = \frac{4\alpha \tan\theta (\tan^2\theta - \alpha^2)}{\cos^2\theta(\alpha^2 + \tan^2\theta)^3}, \tag{4.46}$$

$$\partial_\alpha \left[D_{\text{Beck.}}(\theta, \alpha) \cdot \cos\theta \right] = \frac{4e^{-\frac{\tan^2\theta}{\alpha^2}} \tan\theta (\tan^2\theta - \alpha^2)}{\alpha^5 \cos^2\theta}. \tag{4.47}$$

Details about the necessary inverse transform mapping can be found in Appendix B. Figure 4.16 contrasts samples drawn from primal and differential microfacet distributions.

Signed integrands and antithetics. As discussed many times already, e.g. in Section 2.4.3, Monte Carlo importance sampling produces zero variance when the sampling density is perfectly proportional to a non-negative integrand. Unfortunately, this property no longer holds for signed integrands—in the worst case, sign-related variance can fully negate the benefits of tailored importance sampling strategies even if they match the integrand in an absolute sense. Multiple techniques exist to handle such cases [57]. We rely on antithetic sampling (Section 2.4.5) where we generate paired and correlated samples from the two lobes. We accomplish this by performing two separate rendering passes using the same random number generator state that are finally averaged.

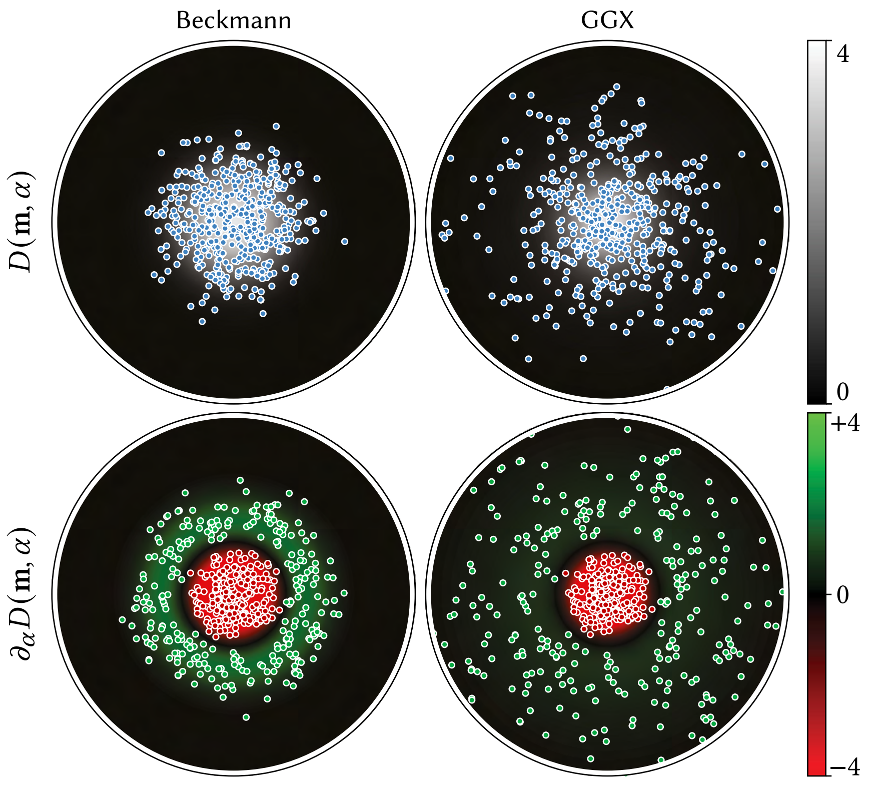


Figure 4.16: Visualization of samples produced by the *Beckmann* and *GGX* (top) and *dBeckmann* and *dGGX* (bottom) sampling techniques for $\alpha = 1/2$. Positive and negative lobes are highlighted in green and red.

Our use of antithetics in the context of differentiable rendering is inspired by Bangaru et al. [73] and also the previous *cross weighting scheme* by Loubet et al. [156]. Antithetic sampling was concurrently investigated by Zhang et al. [74] who also noticed high variance derivative estimates in the presence of highly specular materials. However, compared to our approach that targets roughness derivatives, Zhang et al. instead consider derivatives due to changes in geometric parameters.

Other considerations. One limitation of differential BSDF sampling strategies is that they require an incident illumination estimate for the newly sampled direction, which must be computed using recursive path tracing or an alternative primal algorithm. Regular BSDF sampling remains necessary to scatter the adjoint radiance to other parts of the scene, which propagates like normal light[6]. These branching random walks cause the method to have a quadratic time complexity as a function of path length, which is

[6]For clarification, we refer to the derivation of radiative backpropagation in Section 4.1.2. In particular, indirect propagation occurs via term (T3), and differential strategies target term (T2) of the differentiated rendering equation (4.11).

also a limitation of the original RB algorithm.

When the differentiation problem involves parameters beyond surface roughness, the differential transport simulation must also incorporate detached BSDF sampling to obtain low-variance estimates of the associated derivatives. The two strategies are then combined via detached MIS. Things become more complicated however in case multiple differential strategies are available: compared to forward rendering, where a single integral is computed to determine the intensity of a pixel, an adjoint rendering pass estimates a potentially large number of derivative integrals simultaneously, i.e. one per differentiated scene parameter. This means we face two conflicting goals. Ideally, light paths are sampled based on differential strategies that target specific parameters. But for computational efficiency of the adjoint simulation we want the sampling techniques to be shared across all parameters. It is currently unclear how these two factors should be best consolidated.

Combinations of differential and detached/attached strategies via MIS are also possible: we combine the differential microfacet sampling technique with standard emitter sampling for some of our tests.

The described technique is not specific to the reflective case and also enables differential sampling of rough transmission. We did not investigate generalizations to more advanced models with anisotropy or vNDF sampling [79] and consider them possible future work.

We mention for completeness that differential strategies could also be attached, which would entail tracking derivatives of T and p with respect to π in Equation (4.41). Attaching was of crucial importance when we were restricted to working with primal sampling techniques, but it is of limited use here as T and p can be arbitrarily chosen. Detached strategies can also directly handle integrands with static discontinuities, for instance when optimizing the materials of a scene with fixed geometry. No special handling of moving discontinuities (Section 4.4) is required in that case, which is beneficial as this comes at considerable additional runtime cost.

4.4 Reparameterizing discontinuous integrands

We finally return to the case of integrands containing discontinuities, whose position furthermore depends on π. To do so, we adopt the high-level framework of Loubet et al. [156] and transform the integrals using the recently proposed parameterization by Bangaru et al. [73]. This counteracts the motion of discontinuities, so that the tools from Section 4.2 are readily applicable.

However, both prior methods by Loubet et al. and Bangaru et al. are designed to work in a context where the primal and differential phases are rigidly coupled via reverse mode differentiation. This means that these methods suffer from severe overheads to store primal program variables that are later needed for differentiation. In contrast, our work operates within the decoupled RB framework [155], which turns the differentiation into an independent simulation that transports derivative radiation from sensors to differentiable objects.

The goal of this section is to clarify how reparameterization-based techniques can be cast into a suitable form to enable their use within such a differential transport simulation. We will also revisit the case of attached samplers to finally address a severe limitation with discontinuities encountered in Section 4.2.2.

4.4.1 Reparameterized radiative backpropagation

Recall the original derivation of RB from Section 4.1.2. It begins with the differential forms of the three equations that jointly define the problem solved by any rendering algorithm: scattering, transport, and measurement. Let us first cover the representative case of scattering, i.e. the rendering equation (2.50), to describe the needed changes. When reparameterized using R, its primal form reads

$$\begin{aligned} L_{\mathrm{o}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) &= L_{\mathrm{e}}(\mathbf{x}, \boldsymbol{\omega}_{\mathrm{o}}) \\ &\quad + \int_{S^2} f_s(\mathbf{x}, R(\boldsymbol{\omega}_{\mathrm{i}}, \pi) \to \boldsymbol{\omega}_{\mathrm{o}})\, L_{\mathrm{i}}(\mathbf{x}, R(\boldsymbol{\omega}_{\mathrm{i}}, \pi))\, |J_R(\boldsymbol{\omega}_{\mathrm{i}}, \pi)|\, \mathrm{d}\Omega^{\perp}_{\mathbf{x}}(\boldsymbol{\omega}_{\mathrm{i}}). \end{aligned} \tag{4.48}$$

Since the discontinuities are now static, it is legal to differentiate under the integral sign.

Application of the product rule then produces a total of four derivative terms:

$$\begin{aligned}\partial_\pi L_\mathrm{o}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o}) &= \partial_\pi L_\mathrm{e}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o})\\ &+ \int_{S^2} \partial_\pi f_\mathrm{s}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi)\to\boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi))\, |J_R(\boldsymbol{\omega}_\mathrm{i},\pi)|\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i})\\ &+ \int_{S^2} f_\mathrm{s}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi)\to\boldsymbol{\omega}_\mathrm{o})\, \partial_\pi L_\mathrm{i}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi))\, |J_R(\boldsymbol{\omega}_\mathrm{i},\pi)|\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i})\\ &+ \int_{S^2} f_\mathrm{s}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi)\to\boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi))\, \partial_\pi |J_R(\boldsymbol{\omega}_\mathrm{i},\pi)|\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}). \end{aligned} \tag{4.49}$$

These expressions simplify considerably however by noting that R and its Jacobian reduce to the identity when they occur in a term that is not differentiated:

$$\begin{aligned}\partial_\pi L_\mathrm{o}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o}) &= \partial_\pi L_\mathrm{e}(\mathbf{x},\boldsymbol{\omega}_\mathrm{o}) && \text{(T1)}\\ &+ \int_{S^2} \partial_\pi f_\mathrm{s}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi)\to\boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega}_\mathrm{i})\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}) && \text{(T2)}\\ &+ \int_{S^2} f_\mathrm{s}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}\to\boldsymbol{\omega}_\mathrm{o})\, \partial_\pi L_\mathrm{i}(\mathbf{x}, R(\boldsymbol{\omega}_\mathrm{i},\pi))\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}) && \text{(T3)}\\ &+ \int_{S^2} f_\mathrm{s}(\mathbf{x}, \boldsymbol{\omega}_\mathrm{i}\to\boldsymbol{\omega}_\mathrm{o})\, L_\mathrm{i}(\mathbf{x},\boldsymbol{\omega}_\mathrm{i})\, \partial_\pi |J_R(\boldsymbol{\omega}_\mathrm{i},\pi)|\, \mathrm{d}\Omega^\perp_\mathbf{x}(\boldsymbol{\omega}_\mathrm{i}). && \text{(T4)}\end{aligned} \tag{4.50}$$

Intuitively, this equation states that the process of differentiation can be modeled by simulating scattering, transport, and eventual measurement of a hypothetical "differential radiance" quantified by $\partial_\pi L_\mathrm{i}$ and $\partial_\pi L_\mathrm{o}$, and there is one such function per scene parameter π.

Equation (4.50) takes the role of an energy balance equation that indicates the following properties:

T1. Differential radiance is emitted when the primal emission L_e depends on π.

T2. Objects, whose material model depends on the parameter π, convert some of the ordinary radiance incident on the surface (L_i) into differential radiance ($\partial_\pi L_\mathrm{o}$).

T3. Once created, differential radiance scatters like ordinary light (i.e. involving the non-differentiated BSDF of scene objects).

T4. Differential radiance is also added or subtracted when the parameterization R expands or contracts space depending on π.

In contrast to RB without reparameterization, (T4) is new and all terms are now at least partially warped by the parameterization.

The same steps can also be applied to the measurement equation (2.47), which results in

$$\begin{aligned}\partial_\pi I_k = &\int_{\mathcal{M}}\int_{\mathcal{S}^2} \partial_\pi W_\mathrm{e}^{(k)}(\mathbf{x}, R(\boldsymbol{\omega}, \pi))\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}) \\ &+ \int_{\mathcal{M}}\int_{\mathcal{S}^2} W_\mathrm{e}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, \partial_\pi L_\mathrm{i}(\mathbf{x}, R(\boldsymbol{\omega}, \pi))\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}) \\ &+ \int_{\mathcal{M}}\int_{\mathcal{S}^2} W_\mathrm{e}^{(k)}(\mathbf{x}, \boldsymbol{\omega})\, L_\mathrm{i}(\mathbf{x}, \boldsymbol{\omega})\, \partial_\pi |J_R(\boldsymbol{\omega}, \pi)|\, \mathrm{d}\Omega_\mathbf{x}^\perp(\boldsymbol{\omega})\, \mathrm{d}A(\mathbf{x}). \end{aligned} \tag{4.51}$$

Bangaru et al. [73] relate the parameterization's Jacobian determinant to a vector field divergence. After reordering terms we get

$$\partial_\pi |J_R(\boldsymbol{\omega}, \pi)| = \nabla_{\boldsymbol{\omega}} \cdot \partial_\pi R = \partial_\pi \left(\nabla_{\boldsymbol{\omega}} \cdot R\right). \tag{4.52}$$

The divergence in parentheses can be cheaply approximated alongside the reparameterization itself (via the convolution in Equation (4.23)), involving derivatives of the weighting kernel w. We perform the outer derivative using reverse mode AD, which will then backpropagate gradients to the scene geometry.

At this point, Equations (4.50) and (4.51) could in principle be solved separately for each scene parameter π to compute differential radiance arriving at a image pixel, resulting in a *gradient image.* This approach does not scale to scenes with high-dimensional parameter spaces, as millions of such images would potentially need to be rendered per gradient descent step.

Analogously to Section 4.1.2, reparameterized RB exploits the reciprocal nature of this problem and transports derivatives in the opposite direction, i.e., from the camera towards scene objects. The radiation emanating from the camera in this phase is a signed quantity ("adjoint radiance") that specifies how the rendered image should change to optimally improve the optimization objective. Only a single transport problem needs to be solved in this case, which is substantially more efficient than the naïve approach mentioned above. Once the adjoint radiance reaches a specific surface location, it must still be converted into a scene parameter gradient. Here, it is useful to observe that a point is generally only characterized by a few local parameters, such as the positions of surrounding vertices, texels referenced by a texture lookup in a shader, etc. Whereas path tracing performs a random walk that *reads* such local surface properties, RB then performs an analogous random walk that *writes* local gradients at every interaction.

Algorithms 4.3 and 4.4 provide the pseudocode of the reparameterized RB method (relying on detached sampling) and can be contrasted to similar code fragments in Section 4.1.2, as well as the original paper [155].

```
def reparam_rb($\pi$, $\delta\mathbf{I}$):
    # Initialize parameter gradient(s) to zero
    $\delta\pi = 0$
    for _ in range(num_samples):
        # Sample ray proportional to sensor response and pixel filter
        ($\mathbf{x}$, $\omega$), s_val, s_pdf = sensor.sample_ray()
        # Query adjoint emitted radiance associated with current ray
        A_e = $A_\mathrm{e}(\delta\mathbf{I}, \mathbf{x}, \omega)$ / num_samples
        # Backpropagate through the reparameterized pixel filter
        $\delta\pi$ += adjoint([[ sensor.eval($\mathbf{x}$, $R(\omega, \pi)$) ]], A_e * $L_\mathrm{i}(\mathbf{x}, \omega)$ / s_pdf)
        # Backpropagate through the divergence
        $\delta\pi$ += adjoint([[ $\nabla_\omega \cdot R(\omega, \pi)$ ]], A_e * s_val * $L_\mathrm{i}(\mathbf{x}, \omega)$ / s_pdf)
        # Propagate adjoint radiance into the scene
        $\delta\pi$ += reparam_rb_Li($\pi$, $\mathbf{x}$, $R(\omega, \pi)$, A_e * s_val / s_pdf)
    # Finished, return gradients
    return $\delta\pi$
```

Algorithm 4.3: Reparameterized radiative backpropagation takes scene parameters $\boldsymbol{\pi}$ and an adjoint rendering $\delta\mathbf{I}$ as input. This pseudocode fragment is responsible for the measurement integral. It samples a set of sensor rays, queries the associated emitted adjoint radiance A_e and propagates these gradients into the scene, while accounting for geometric discontinuities via reparameterization.

```
def reparam_rb_Li($\pi$, $\mathbf{x}$, $\omega$, $\delta L$):
    # Find an intersection with the scene geometry
    $\mathbf{x}'$ = $\mathbf{r}(\mathbf{x}, \omega)$
    # T1: Backpropagate to reparameterized emitter parameters, if any
    $\delta\pi$ = adjoint([[ $L_\mathrm{e}(\mathbf{x}', -\omega)$ ]], $\delta L$)
    # Sample a direction from the BSDF
    $\omega'$, b_val, b_pdf = sample $f_s(\mathbf{x}', \omega' \to -\omega)$
    # T2: Backpropagate to reparameterized BSDF parameters
    $\delta\pi$ += adjoint([[ $f_s(\mathbf{x}', R(\omega', \pi) \to -\omega)$ ]], $\delta L$ * $L_\mathrm{i}(\mathbf{x}, \omega')$ / b_pdf)
    # T4: Backpropagate through the divergence
    $\delta\pi$ += adjoint([[ $\nabla_{\omega'} \cdot R(\omega', \pi)$ ]], $\delta L$ * b_val * $L_\mathrm{i}(\mathbf{x}, \omega')$ / b_pdf)
    # T3: Recurse to account for indirect differential radiance
    return $\delta\pi$ + reparam_rb_Li($\pi$, $\mathbf{x}'$, $R(\omega', \pi)$, $\delta L$ * b_val / b_pdf)
```

Algorithm 4.4: This pseudocode fragment provides `reparam_rb_Li()` referenced in Algorithm 4.3. It implements the reparameterized RB version of the transport and scattering equation that transports derivatives through the scene and backpropagates adjoint radiance δL to objects with differentiable parameters.

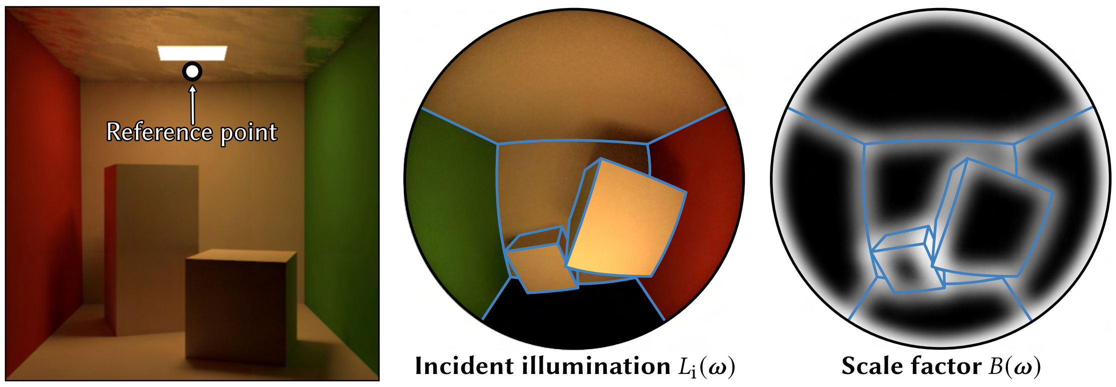


Figure 4.17: The incident illumination $L_\mathrm{i}(\boldsymbol{\omega})$ that is visible from a given shading point contains many discontinuities (blue lines). Attached sampling in such a scenario produces biased gradients, even if these discontinuities are static with respect to the differentiated scene parameter. To address this complication, we introduce a reparameterization that gradually slows down differential sample movement caused by such strategies based on a smooth scaling factor $B(\boldsymbol{\omega})$ which approaches one at all discontinuities, including the horizon of the visible hemisphere.

4.4.2 Reparameterizing attached strategies

As discussed earlier in Section 4.2.2, attached sampling provides a convenient way to reuse a primal sampling procedure in a differential estimator. Accounting for its parameter dependence during differentiation can be an effective variance reduction strategy.

Unfortunately, the parameter-dependent change of variables $T(\mathbf{u}, \pi)$ also causes previously static discontinuities to move with respect to perturbations in π (see Figure 4.12), which severely limits the utility of this approach. To address these challenges, we we define a secondary parameterization analogous to Equation (4.22), which will similarly counteract the movement of the sampled directions to prevent issues with bias due to discontinuities:

$$R^{(\mathrm{A})}(\boldsymbol{\omega}, \pi) = \boldsymbol{\omega} - B(\boldsymbol{\omega})\, T(\mathbf{u}, \pi) + B(\boldsymbol{\omega})\, T(\mathbf{u}, \pi_0). \tag{4.53}$$

Interestingly, our goal here is reversed from the previous reparameterization (4.22): instead of introducing a new dependence of samples on π, we now want to (partially) suppress movement. Again, π_0 only takes on the (primal) value of π but is not differentiated, and $\mathbf{u} = T^{-1}(\boldsymbol{\omega}, \pi_0)$. $B(\boldsymbol{\omega})$ can be interpreted as a smooth scaling factor used to slow down the moving samples as they approach discontinuities. For example, setting $B(\boldsymbol{\omega}) = 1\, \forall\, \boldsymbol{\omega}$ would freeze *all* sample movement and the attached and reparameterized strategy simplifies to the standard detached estimator (4.29).

Based on this observation, we will use $B(\boldsymbol{\omega})$ to transition back to a detached estimator near the discontinuities where the attached strategies are troublesome. At the same time, we can still take advantage of the variance reduction due to attached sampling away from discontinuities. We found the boundary test $\tilde{B}$ from Bangaru et al. [73] to work well for this purpose when turned into a smoothed interpolant using their harmonic convolution approach (4.23). Before the convolution, we also multiply $\tilde{B}$ with another scaling factor that freezes samples towards the horizon of the hemisphere defined from the shading frame. This is important when, e.g., evaluating a reparameterized BSDF and differentiation affects the surface normal defining its shading frame. We show an (abstract) visualization of such a scaling factor in Figure 4.17.

To apply the reparameterization $R^{(\mathrm{A})}$ in practice, we also need to compute derivatives of its Jacobian determinant:

$$\begin{aligned}\partial_\pi |J_{R^{(\mathrm{A})}}(\boldsymbol{\omega}, \pi)| &= \partial_\pi \left[\nabla_{\boldsymbol{\omega}} \cdot R^{(\mathrm{A})}(\boldsymbol{\omega}, \pi)\right] \\ &= \partial_\pi \left[\nabla_{\boldsymbol{\omega}} \cdot (B(\boldsymbol{\omega})\, T(\mathbf{u}, \pi))\right] \\ &= \nabla_{\boldsymbol{\omega}}\, B(\boldsymbol{\omega}) \cdot \partial_\pi T(\mathbf{u}, \pi) + B(\boldsymbol{\omega}) \cdot \partial_\pi \left[\nabla_{\boldsymbol{\omega}} \cdot T(\mathbf{u}, \pi)\right] \\ &= \nabla_{\boldsymbol{\omega}}\, B(\boldsymbol{\omega}) \cdot \partial_\pi T(\mathbf{u}, \pi) + B(\boldsymbol{\omega}) \cdot \partial_\pi |J_T(\mathbf{u}, \pi)|. \end{aligned} \tag{4.54}$$

For this we used again the relation between divergence and Jacobian determinant [73] and the fact that B is a scalar quantity that is independent of π. The directional derivative $\nabla_{\boldsymbol{\omega}}\, B(\boldsymbol{\omega})$ can be approximated alongside $B(\boldsymbol{\omega})$ as part of the harmonic convolution (4.23) and the Jacobian determinant has a closed form solution

$$|J_T(\mathbf{u}, \pi)| = \frac{p(T(\mathbf{u}, \pi_0), \pi_0)}{p(T(\mathbf{u}, \pi), \pi)} \tag{4.55}$$

using the probability density p associated with T.

As this new reparameterization is based on similar principles as prior work [73, 156], it also inherits its limitations. In particular it is only defined for integration over the unit sphere $\mathcal{S}^2$ and is not sufficient for attached strategies that involve differentiation through multiple scattering events at once. To ensure the correctness of the computed gradients in our implementation, we thus split the integrand in Equation (4.48) into two parts. The indirect illumination component of L_i is handled analogously to the previous section and uses a detached estimator. The direct component however involves a nested reparameterization where R and $R^{(\mathrm{A})}$ now counteract both types of discontinuities discussed in this section. Reverse mode differentiation will then automatically propagate gradients through both parameterizations. Finally, we can apply the attached estimator from Equation (4.31) to the integral. We call this approach the *reparameterized attached* strategy, and two example comparisons to “naïve” attaching can be seen in Figure 4.18.

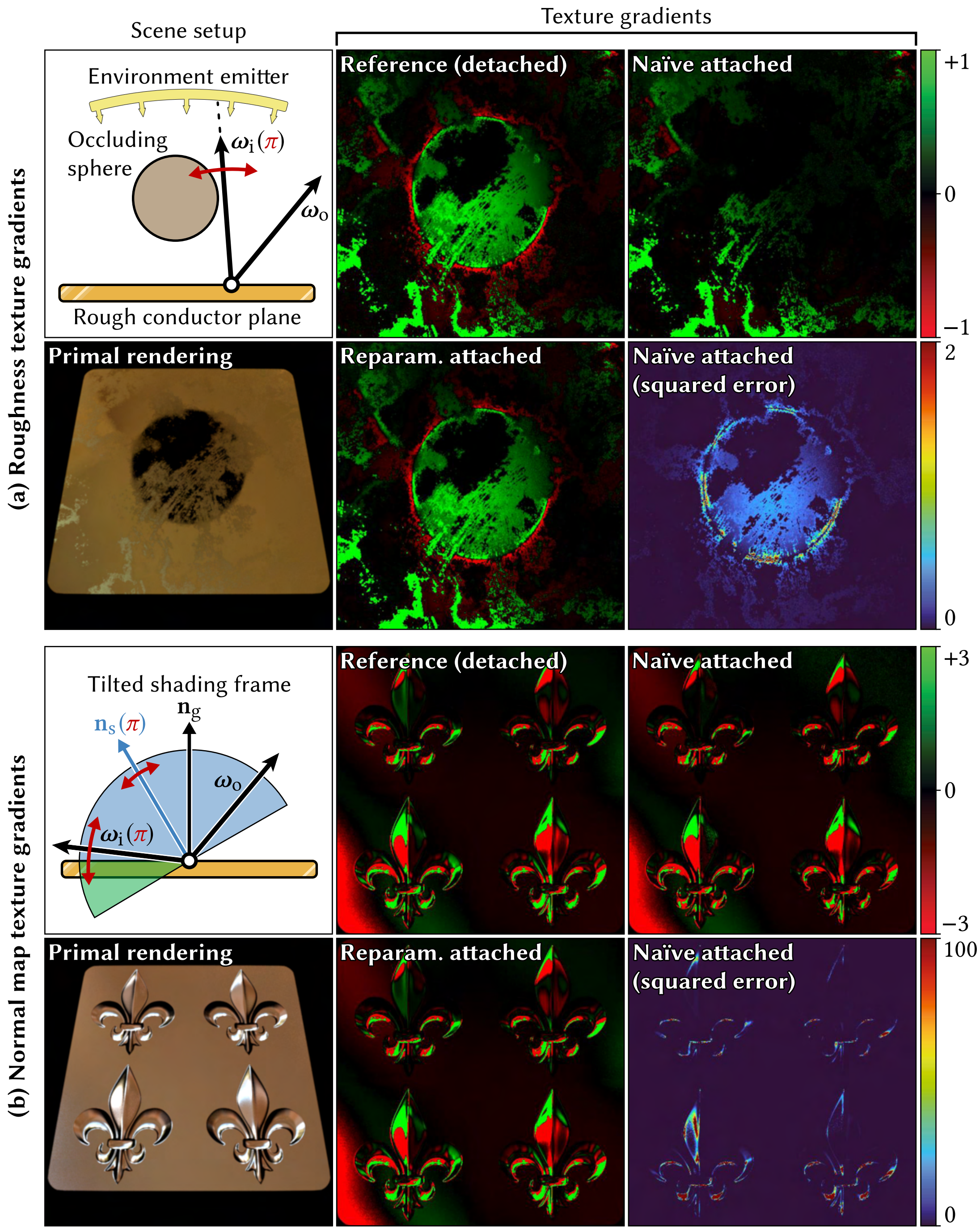


Figure 4.18: We show two typical examples where reparameterized attached sampling is crucial to compute correct parameter gradients in presence of static discontinuities. **(a)** We differentiate pixel intensity with respect to the textured roughness π of a metal surface. Part of the environment illumination is occluded by a sphere, causing a discontinuity for the sampled directions $\boldsymbol{\omega}_i$ that move based on π. **(b)** A similar setup where π represents a normal map that causes shading normals $\mathbf{n}_s$ to deviate from the geometric normal $\mathbf{n}_g$. No discontinuities are present apart from the boundary of the shading frame that is affected by π, i.e. $\boldsymbol{\omega}_i$ that are sampled based on $\mathbf{n}_s$ can "move" below the horizon. Naïve attached sampling misses important gradients compared to a (detached) reference that our reparameterized attached strategy is able to compute correctly.

4.5 Results

We evaluated our methods experimentally in a differentiable rendering system based on Mitsuba 2 [4]. All experiments were performed on an NVIDIA TITAN RTX graphics card (23 GiB of RAM) using OptiX 7.2 [181] for hardware-accelerated ray tracing.

4.5.1 Variance analysis

We now turn to concrete example scenes to analyze the statistical behavior of several differential estimators presented in this chapter. Figure 4.19 represents a controlled test setup based on a scene by Veach and Guibas [66] (compare also with Figure 2.27) with single-bounce glossy reflections for varying roughness values and light source sizes. Figures 4.20 and 4.21 revisit the complex scene from Figure 4.1 that contains many glossy interreflections involving varying degrees of roughness, as well as complex illumination from a combination of area lights and an environment map.

In both cases we compute gradient images with respect to a single value π that is added to all roughness parameters in the scene. In other words, this illustrates what happens to the renderings when all glossy objects are roughened slightly. Like in primal rendering, the efficiency of estimators depends also on the concrete values of the (differentiated) scene parameters so this allows us to assess the variance at various levels of roughness at once.

Recall that the desired roughness gradients are computed through the BSDF term (T3) in the differentiated (and reparameterized) rendering equation (4.50). As discussed throughout the previous sections, there exist many sampling strategies to construct a Monte Carlo estimator, and the following table lists all individual options:

BSDF sampling variants:

Detached	Reparam. attached	Diff. detached	Naïve attached
✓	✓	✓	✗

Emitter sampling variants:

Detached
✓

Most estimators (green ticks) are valid whereas the *naïve attached* strategy produces biased gradients (red cross) due to the discontinuities introduced by the attached sampling process. Note that emitter sampling only exists in a default detached option as it is independent of the differentiated roughness parameter.

When now combining emitter sampling and the four BSDF sampling strategies using MIS we arrive, in principle, with 16 additional estimators, because the involved MIS weights can also be either attached or detached:

BSDF sampling variants + MIS with (detached) emitter sampling:

	Detached	Reparam. attached	Diff. detached	Naïve attached
Detached estimator, detached MIS weights	✓		✓	
Attached estimator, attached MIS weights		✓		✗
Attached estimator, detached MIS weights		✗		✗
Detached estimator, attached MIS weights	✓		✓	

Most combinations however are incompatible (grey boxes): for example, attached MIS is not able to combine two detached strategies. Also recall that the MIS variant involving detached weights but attached estimators always produces biased results. As a consequence, only five variants are actually valid and unbiased.

Figures 4.19 to 4.21 compare all correct estimator options, with the exception of the *detached estimator, attached weights* combination (last row) which we found to perform slightly worse, but roughly equivalent to the version that detaches both the weights and the estimators (first row). In addition, we also show results using the *naïve attached* strategy (without any MIS) to again illustrate the bias due to discontinuities.

Like in forward rendering, (detached) emitter sampling is good at handling concentrated illumination reflected by relatively rough surfaces. In contrast, the analysis of material-based differential estimators is more nuanced. While weaknesses of individual strategies mostly carry over into their differentiated versions, the same is not the case for their strengths. For instance, the effectiveness of detached BSDF sampling on highly specular materials is greatly reduced compared to its primal counterpart, which occurs due to mismatches between integrand and sampling density (Section 4.2.1). The differential sampling strategy for microfacet BSDFs (Section 4.3) is generally the most robust in these tests. None of the discussed strategies is specifically designed to handle chains with multiple glossy interactions, and variance is consequently high in such image regions. Attached sampling (Section 4.2.2) is most complex in terms of differentiation, since it must also consider the parameter dependence of samples. If done naïvely, this dependence can introduce discontinuities that can add bias. Our reparameterized attached strategy (Section 4.4.2) avoids this bias and at times achieves significant improvements over detached BSDF sampling. Its effectiveness in Figures 4.20 and 4.21 is held back by limitations regarding product sampling between BSDF and incident illumination (Section 4.2.2). Adding MIS with the emitter sampling strategy improves robustness in all cases, as expected from analogous situations when rendering primal images.

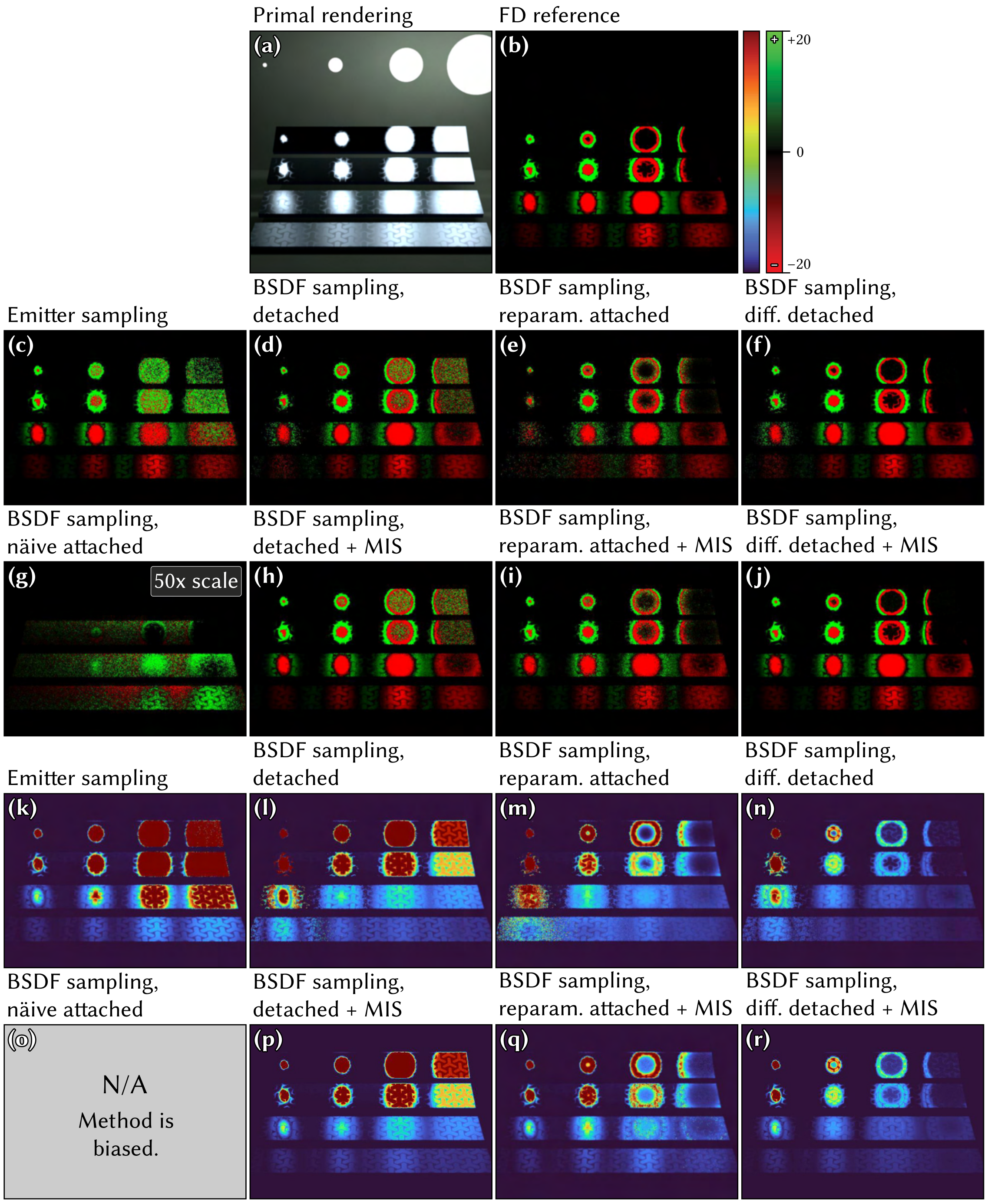

Primal rendering
FD reference
(a)
(b)
+20
0
−20
Emitter sampling
BSDF sampling, detached
BSDF sampling, reparam. attached
BSDF sampling, diff. detached
(c)
(d)
(e)
(f)
BSDF sampling, näive attached
BSDF sampling, detached + MIS
BSDF sampling, reparam. attached + MIS
BSDF sampling, diff. detached + MIS
(g)
50x scale
(h)
(i)
(j)
(k)
(l)
(m)
(n)
N/A
Method is biased.
(p)
(q)
(r)

Figure 4.19: Equal-time comparison of gradient estimators based on a classic scene by Veach and Guibas. We differentiate the roughness textures of the metal plates; the average Beckmann roughness increases from top to bottom ($\alpha_{\text{avg}} \in \{0.01, 0.02, 0.06, 0.13\}$). **(a)** Primal rendering of the scene. **(b)** Ground-truth gradients computed using finite differences at a high sample count. **(c)** Gradients computed using emitter sampling. **(d–f)** Three unbiased estimators using detached, differential detached, and reparameterized attached BSDF strategies. **(g)** As in Figure 4.18, naïve attached BSDF sampling exhibits bias due to parameter-dependent discontinuities. We had to scale the gradients of this technique by a factor of 50× so that they are visible. **(h–j)** Additional estimators involving MIS with emitter sampling. **(k–r)** Visualizations of the standard deviation for all estimators.

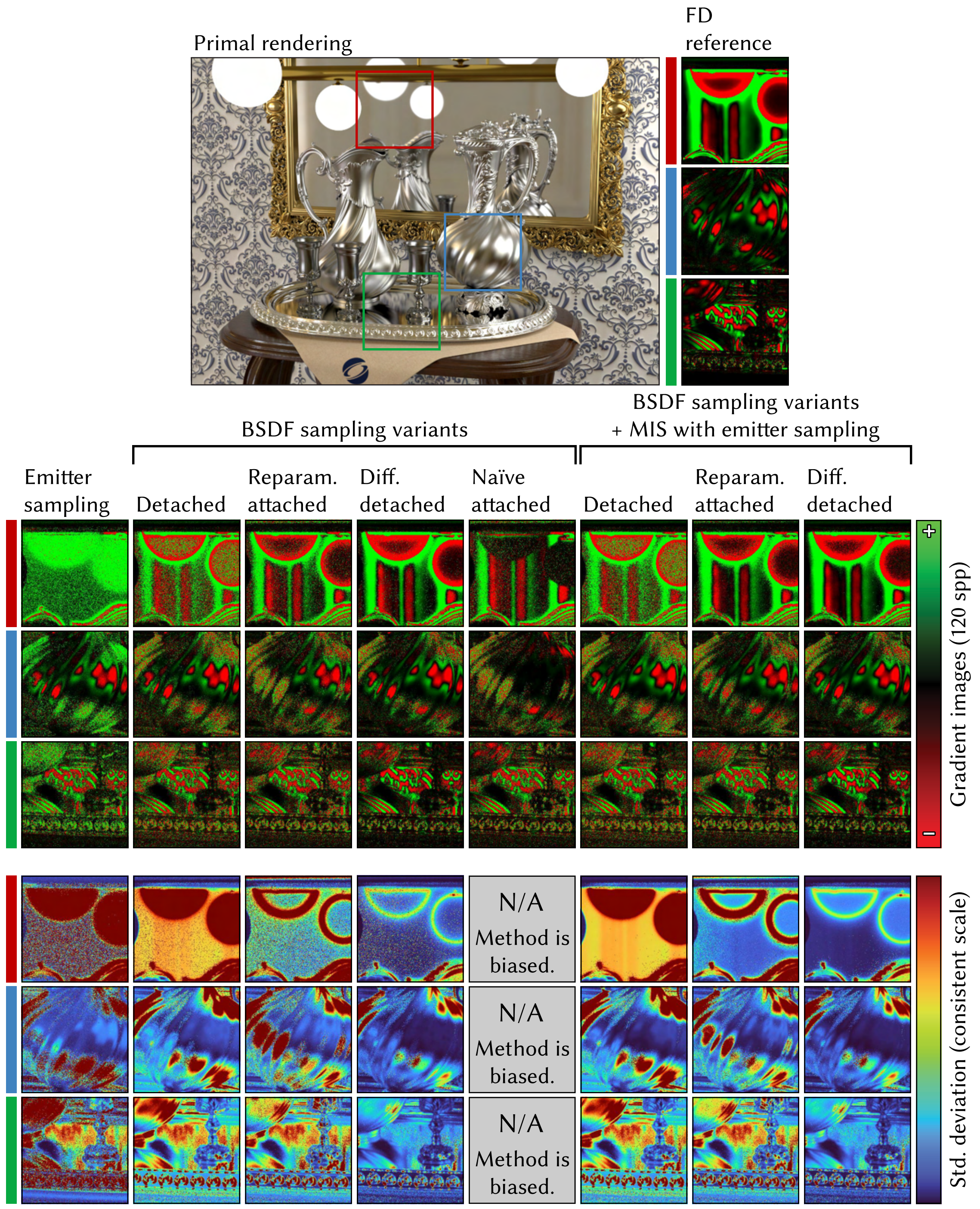


Figure 4.20: Equal-sample comparison of the discussed gradient estimators on a scene involving glossy interreflections and complex illumination. Insets focus on three representative regions of the image where we show both the computed gradient images, and the estimator standard deviation using false color visualizations.

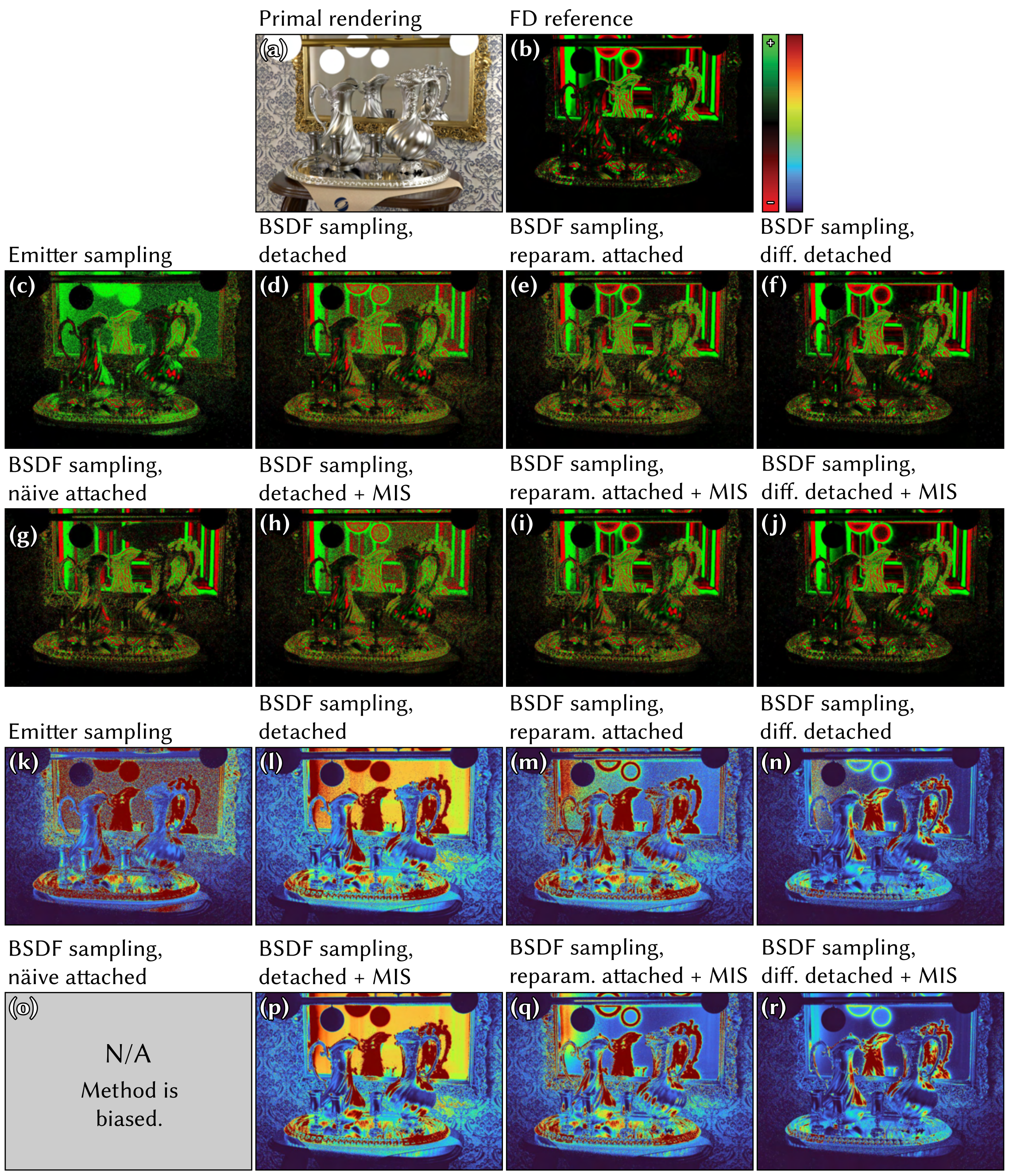

Primal rendering
FD reference
(a)
(b)
+
−
Emitter sampling
BSDF sampling, detached
BSDF sampling, reparam. attached
BSDF sampling, diff. detached
(c)
(d)
(e)
(f)
BSDF sampling, näive attached
BSDF sampling, detached + MIS
BSDF sampling, reparam. attached + MIS
BSDF sampling, diff. detached + MIS
(g)
(h)
(i)
(j)
Emitter sampling
BSDF sampling, detached
BSDF sampling, reparam. attached
BSDF sampling, diff. detached
(k)
(l)
(m)
(n)
BSDF sampling, näive attached
BSDF sampling, detached + MIS
BSDF sampling, reparam. attached + MIS
BSDF sampling, diff. detached + MIS
(o)
N/A
Method is biased.
(p)
(q)
(r)

Figure 4.21: The same comparison as in Figure 4.20 but using uncropped images.

4.5.2 Optimizing spatially varying roughness

Figure 4.22 compares the convergence of different estimators in a simple optimization task. We optimize the spatially varying roughness of a flat surface using standard stochastic gradient descent and a single view. We compare three estimators (detached, differential, and attached BSDF sampling) and two conditions: a rough microfacet (GGX) material under natural environment illumination, and a more specular microfacet (Beckmann) material under smooth synthetic directional illumination (three colored light sources with emission profiles modeled by spherical von Mises-Fisher distributions). Reparameterization is not needed here due to the lack of discontinuities.

All three methods compute the correct gradients in expectation, and they generate samples in a comparable amount of time. Therefore, the main distinguishing factor is their variance and the resulting impact on convergence speed. In all cases, we begin with a randomly initialized texture and run 100 iterations of SGD with a fixed learning rate. The observed convergence behavior is unsurprising and matches our previous observations on the variance of specific estimators: detached sampling handles rough reflections relatively well, but is clearly outperformed by the other methods in the more specular setting. Attached sampling based on the BSDF is expected to perform poorly when the reparameterization warps another factor with significant variation like the interior environment map. Nonetheless, this approach actually performs best in the second setting with smooth illumination. Differential sampling is robust in both cases and always outperforms detached sampling.

4.5.3 Efficient differentiation of geometric discontinuities

We compare our approach to geometric discontinuities to the method of Loubet et al. [156]. The differential evaluation of this prior work was based on conventional AD and therefore rigidly coupled to the primal computation. Both steps proceeded via *wavefronts*, where one or more computational kernels were launched per scattering event, exchanging intermediate state via global memory. However, both reverse mode AD and wavefront-style execution come at the cost of severe storage requirements that are proportional to both scene complexity (path length) and rendering quality (resolution, samples per pixel). Once the available GPU memory is exhausted, the computation must be split into multiple passes to curtail memory usage, which tends to further increase the cost of gradient evaluation.

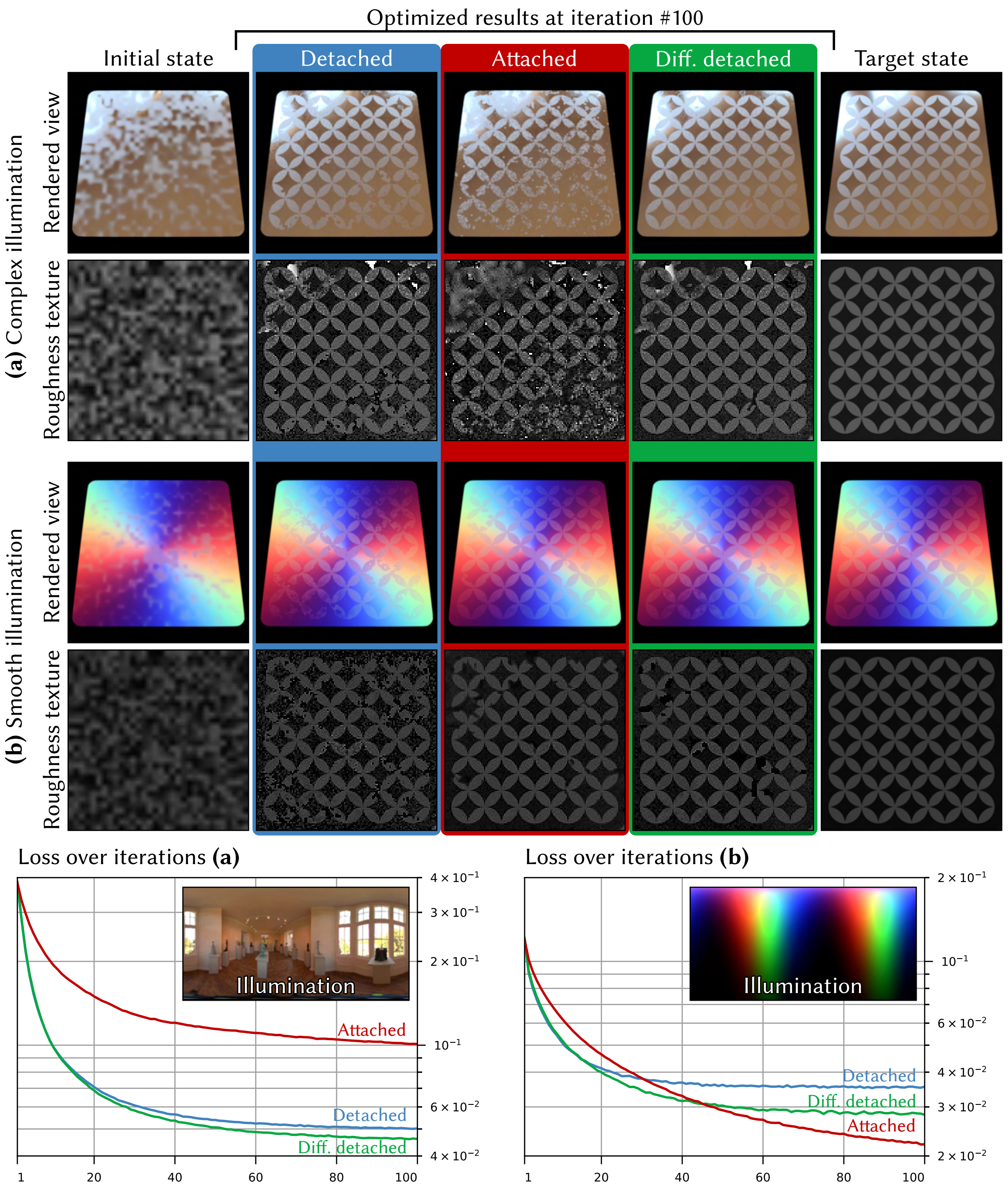


Figure 4.22: Convergence comparison of various BSDF sampling techniques in the context of roughness texture optimization. The statistical efficiencies when computing gradients using the various estimators (detached, differential detached, and attached) also manifest themselves in terms of varying convergence rates when using them during optimization based on stochastic gradient descent. Differential detached sampling (**green**) performs robustly in different settings and always outperforms the detached method (**blue**) that relies on less efficient, primal microfacet sampling in this case. Attached sampling (**red**) can outperform both detached variants in cases where incident illumination is smooth (**bottom**) but can be inefficient in case of more complex illumination (**top**) due to its limitations involving product integrals.

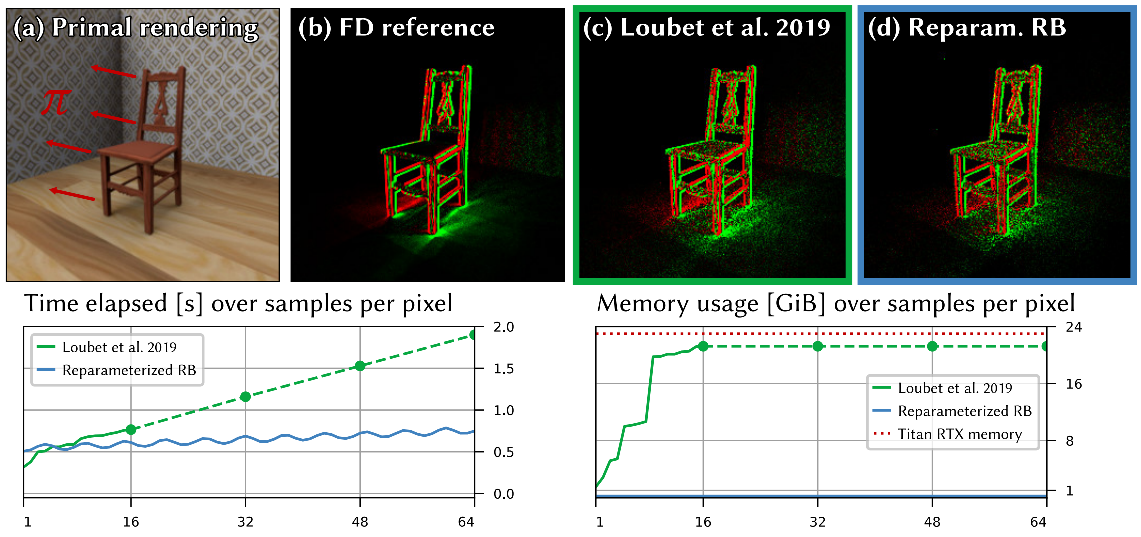


Figure 4.23: Performance comparison between reparameterized RB (**blue**) and the method of Loubet et al. [156] (**green**). When computing image gradients with respect to translation of the chair, both methods produce gradients with roughly equivalent variance, but involving substantially different time and memory requirements. Loubet et al.'s method is based on wavefront-style evaluation that saturates the 23 GiB of VRAM of a Titan RTX card at a resolution of 256×256 with 16 samples per pixel. Beyond this point, computation needs to be split into multiple passes or crops, causing a steeper increase in computation time (dashed lines). The adjoint approach of reparameterized RB is compatible with a more efficient megakernel-style evaluation with minimal memory requirements and improved runtime cost.

The technique described in Section 4.4 improves upon this in two ways: first, it incorporates Loubet et al. [156] change of variables into a differential light transport simulation that propagates the derivative of received radiance in reverse mode. This breaks the rigid coupling between primal and differential phases and thus also the need to memorize primal program variables associated with each scattering interaction. The second improvement comes as an immediate corollary and matches a corresponding step in prior work [155]: casting the differentiation task as a transport simulation enables the implementation of the method using *megakernel*-style evaluation that finally removes all need to maintain large memory regions for intermediate program state.

The benchmark in Figure 4.23 compares our method's computation time and memory usage to the method of Loubet et al. using their reference implementation. The test scene exhibits low complexity, and the path length was limited to only two bounces. Still, memory usage of the method of Loubet et al. easily exceeds the total capacity of the used NVIDIA TITAN RTX graphics card (23 GiB) at low image resolution and sample count. Increasing either quality knob then requires rendering in passes, which has adverse effect on the computation time. The memory usage of our method is independent

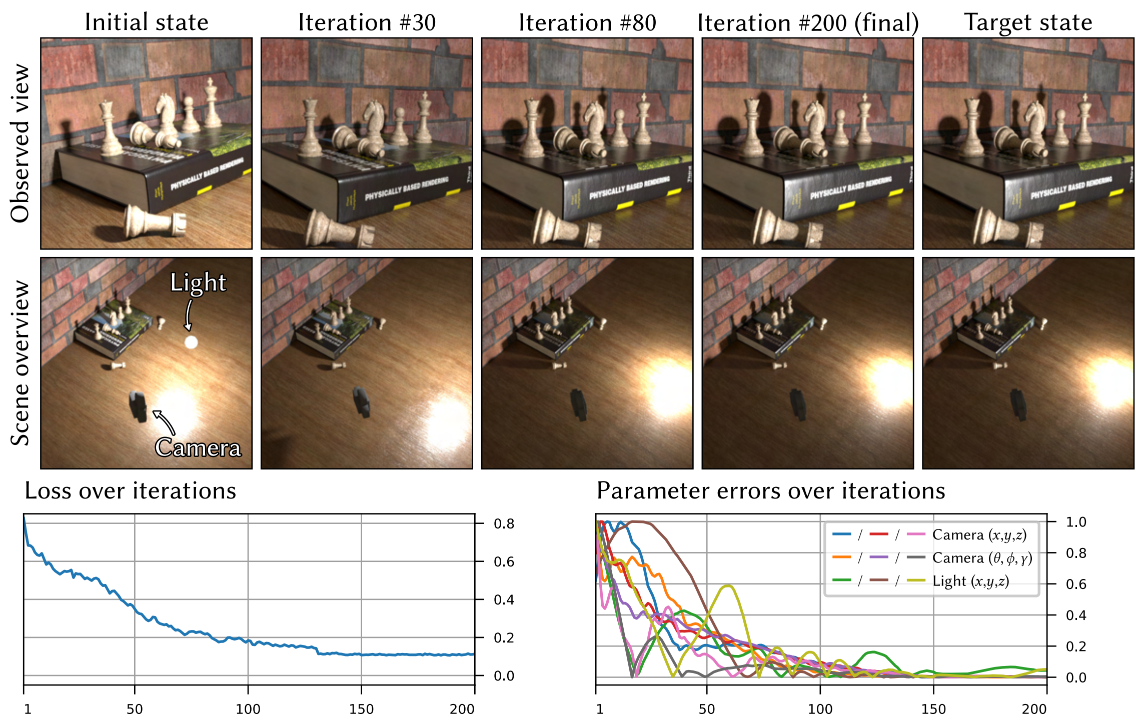


Figure 4.24: 6D camera pose and light position estimation in a scene with complex light transport involving soft shadows and glossy interreflections. **Top**: Camera view at different steps during optimization. **Bottom**: Visualization of the scene from the top at the same steps (camera shown as an actual object). **Right**: Convergence of the loss and error plots of the 9 parameters that are jointly optimized.

of resolution and sample count as no primal simulation variables must be stored, and this leads to improved scalability in such cases.

We now demonstrate two prototypical optimization tasks performed using the reparameterized RB method.

4.5.4 Camera and light source pose estimation

In Figure 4.24, we consider a 6D pose estimation application: we determine the position and orientation of the camera in a target image, as well as the location of a spherical area light. The scene used in this experiment exhibits complex effects like glossy interreflections, soft shadows, and global illumination which are all naturally handled using physically based methods but would be challenging for differentiable rasterizers. Our method reparameterizes all ray directions to avoid discontinuities from the moving light source. Recall from Section 4.4 that this step computes surface intersections such that the ray direction derivatives follow the direction of the silhouette discontinuities. Unsurprisingly, the moving camera causes a similar type of discontinuity requiring another reparameterization. It uses a slightly modified ray intersection, in which the camera origin and direction are explicitly part of the differentiation process.

4.5.5 Geometry and shading optimization

Figure 4.25 showcases the joint reconstruction of shape and material from a set of target images, using our reparameterized RB algorithm with detached BSDF sampling (all materials are diffuse, hence attached and detached strategies coincide). The reference is synthetic and rendered from 5 surrounding viewpoints, and we furthermore validate the resulting reconstruction using a hold-out viewpoint placed on top of the target. We parameterize the object using a displaced ellipsoid base mesh (64×64 displacement texture) and a diffuse albedo texture (1024×1024 pixels), which requires simultaneous differentiation with respect to more than one million parameters. For this to be feasible within a realistic amount of time, reverse mode differentiation is key. We have found the relative L1 loss function to be well-suited for this challenge as it focuses evenly on all regions of the images despite different brightness levels. To avoid convergence to local minima during the optimization, we use a multi-resolution scheme where the optimized textures start out at low resolution and are gradually upsampled to their target sizes throughout the process.

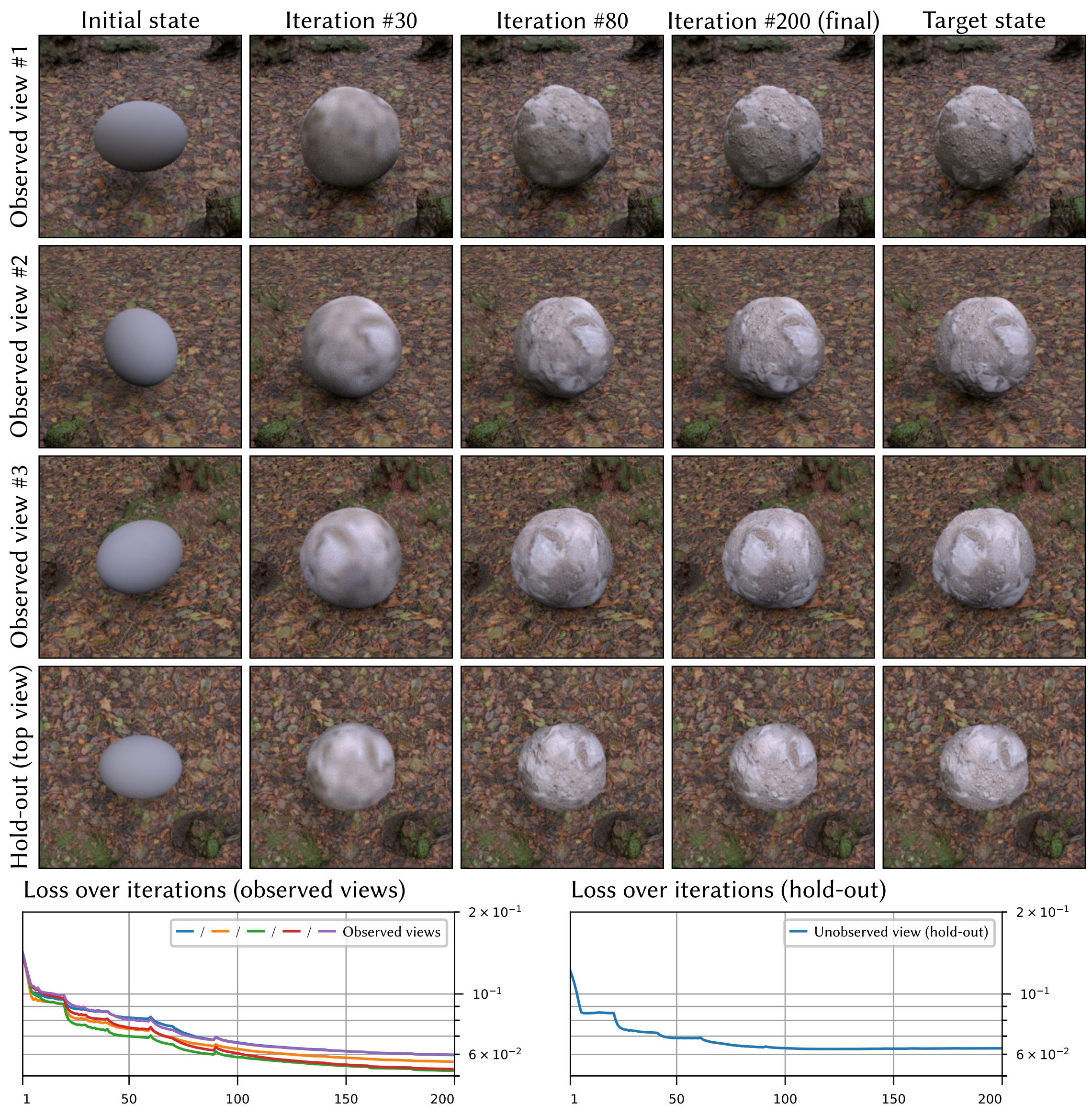


Figure 4.25: Joint optimization of a displacement map and a diffuse texture. More than a million parameters are optimized simultaneously (1024 × 1024 diffuse texture, 64 × 64 displacement texture) for five different view points scattered around the object. **Rows 1-3**: Subsequent states of the target object during the optimization for three out of the five observed views. **Row 4**: The same states from a hold-out point of view, looking at the object from the top. **Row 5**: Convergence rates for both optimized and unoptimized view points.

4.6 Summary and future work

Differentiable Monte Carlo rendering provides a powerful new instrument in the pursuit of complex visual inverse problems in computer graphics and beyond. The initial problem definition is easily stated: one must simply evaluate the derivative of an estimator. Yet, pursuing the path of this harmless differential leads to an astonishing proliferation of estimators, parameterizations, and parameterizations of parameterizations, revealing that we must now revisit many previously well-understood aspects of rendering in a different light. Our work represents a first survey of the large space of differential Monte Carlo transport estimators. Many specimens encountered by this exploration were still fundamentally based on an underlying primal algorithm, although we show that specialized differential estimators hold significant promise in improving the efficiency of differentiable rendering in the future. Many other directions are conceivable: we envision next event estimators for emitted differential radiance and differential path guiding. At the same time, our analysis shows that intuition from the primal world may not always transfer.

Discontinuous integrands remain a bothersome element of differentiable rendering. Our work shows how suitable reparameterizations can be integrated into an efficient adjoint method, enabling geometric optimization of scenes with vast numbers of parameters and essentially no memory overheads. On the flipside, these mappings increase the cost of differentiable rendering considerably, and their stochastic nature can inject extra variance into otherwise benign integrals. Another current issue entails chains of specular interactions that may each introduce a separate set of discontinuities. Current parameterization-based techniques only focus on directly visible discontinuities and therefore cannot handle such cases.

Multiple importance sampling for differentiable estimators remains highly useful, but other aspects of it are still poorly understood: differentiable estimators are potentially much worse than their primal analogues, and this is not "perceived" by MIS weights that are based on primal probabilities. Developing a truly differential form of MIS that transfers the optimality will be a promising avenue of future research.

While gradients are important for high-dimensional optimization, they alone may not be enough when the objective is highly non-convex. Certain scene representations (e.g. vertex positions of a triangle mesh) are particularly susceptible and produce undesirable local minima. Further research is necessary to understand how scene parameterizations affect the energy landscape of optimization tasks.

5 | Conclusion

We investigated two different use cases of light path gradients in this thesis that advance the areas of forward and inverse physically based rendering respectively.

The first part of the thesis is concerned with forward rendering of caustics and glints, two visually striking optical effects that are particularly troublesome for many previous rendering techniques. The main challenge here lies in finding specular light paths that satisfy the laws of reflection and refraction at each path vertex. We proposed a surprisingly simple, and extensible sampling strategy for these paths and can drastically improve convergence rates in these settings. At its core, the method employs numerical root-finding in the domain of light paths that is driven by their geometric gradients.

In the second part we discussed differentiable rendering and how it can be used to solve challenging inverse problems. By differentiating the entirety of light transport in a scene, we can apply gradient-based optimization to recover large numbers of unknown scene parameters from a set of image observations. We presented a comprehensive overview of the large space of differential Monte Carlo estimators that can solve such problems. Additionally, we examined how to differentiate the inconvenient but omnipresent case of discontinuous rendering integrals.

The underlying Newton solver of our caustic sampling method is, in principle, a differentiable computation itself. This means it is also conceivable to combine the two topics above. Gradients due to changes in specular geometry are especially helpful for inverse design tasks of caustics or lenses and specialized differentiable rendering algorithms targeting these use cases could be highly beneficial.

The use of gradients for forward or inverse rendering is not fundamentally new—there is a long tradition of accelerating rendering techniques using various forms of derivative information, and first instances of differentiable rendering for optimizing scene parameters also date back almost a decade. However, the latter has recently seen a tremendous rise in popularity and has developed into its own, fascinating sub-field of physically based rendering. Advances in this area have the potential to greatly impact imaging tasks in many scientific disciplines outside of core computer graphics. This presents an exciting time for rendering researchers: not only does differentiation require us to answer a plethora of new questions, there is also ample opportunity to repurpose and adapt existing forward rendering techniques.

The forward problem itself is also far from being solved. Path tracing strikes a good balance between rendering efficiency and implementation simplicity. With the help of dedicated ray tracing hardware on modern GPUs, it can even be pushed towards real-time performance. But it can also be surprisingly brittle in certain scenarios. For current applications, users of a rendering system have to carefully design input scenes around known limitations of the algorithm to achieve reasonable convergence rates. More work on robust techniques is needed to improved these workflows. It is also likely that these will simultaneously advance the state-of-the-art in inverse rendering techniques as the two sub-fields are tightly coupled.

There are a few notable research directions that are orthogonal to the two main chapters of this thesis.

For instance, most material models in current production systems are built from similar building blocks—usually a combination of diffuse reflectance and a specular response based on microfacet theory. While this can produce a wide range of plausible appearances it remains an idealized model and cannot match the true scattering distribution of many materials found in the real world.

Due to the relatively slow convergence rate of Monte Carlo integration, only few practical applications can actually afford to render completely noise-free images. Instead, denoising methods are an indispensable part of production pipelines. These exceed at recovering fine details from geometry or textures at extremely low sample counts—but often still struggle to resolve other noisy high-frequency content such as caustics.

It is likely that denoising could play an equally important role in the context of differentiable rendering, where sparse and noisy gradient observations could be reconstructed in order to significantly speed up optimization tasks.

We are excited to see where future progress in physically based forward and inverse rendering will lead and what new insights we can gain along this journey.

A | Derivatives of the new specular manifold constraints

Recall our proposed specular constraints from Section 3.3.1,

$$\mathbf{c}_i = \begin{pmatrix} \theta(\overrightarrow{\boldsymbol{\omega}_i}) - \theta(\mathrm{S}(\overleftarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \\ \phi(\overrightarrow{\boldsymbol{\omega}_i}) - \phi(\mathrm{S}(\overleftarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \end{pmatrix}, \tag{A.1}$$

here slightly rewritten using a shorthand notation

$$\overrightarrow{\boldsymbol{\omega}_i} := \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|} \quad \text{and} \quad \overleftarrow{\boldsymbol{\omega}_i} := \frac{\mathbf{x}_{i-1} - \mathbf{x}_i}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|}. \tag{A.2}$$

We will assume that each specular vertex $\mathbf{x}_i$ is ultimately parameterized with 2D coordinates (u_i, v_i). In order to use the multivariate Newton method to finds roots of Equation (A.1), we need access to the partial derivatives $\partial \mathbf{c}_i / \partial u_j$ and $\partial \mathbf{c}_i / \partial v_j$. Like in the case of the previous half-vector constraints by Jakob and Marschner [116], each $\mathbf{c}_i$ only involves the three vertices $\mathbf{x}_{i-1}, \mathbf{x}_i$ and $\mathbf{x}_{i+1}$, i.e. many entries of the Jacobian matrix are zero-valued. See Section 3.1 for details.

In the following paragraphs we list the required derivatives with respect to an arbitrary variable s that can either be the u or v dimension of the parameterization. Ultimately, these depend on the partial derivatives of surface positions and shading normals $\partial \mathbf{x} / \partial u$, $\partial \mathbf{n} / \partial u$, etc. that are usually computed during the ray-geometry intersection in most rendering systems [8].

Note also that in cases of total internal reflection, we use the alternate constraint

$$\mathbf{c}_i = \begin{pmatrix} \theta(\overleftarrow{\boldsymbol{\omega}_i}) - \theta(\mathrm{S}(\overrightarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \\ \phi(\overleftarrow{\boldsymbol{\omega}_i}) - \phi(\mathrm{S}(\overrightarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \end{pmatrix} \tag{A.3}$$

instead. The computation of its derivatives follows analogously however.

Constraint derivatives.

$$\frac{\partial \mathbf{c}_i}{\partial s_{i-1}} = -\frac{\partial}{\partial s_{i-1}} \theta(\mathrm{S}(\overleftarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \tag{A.4}$$

$$\frac{\partial \mathbf{c}_i}{\partial s_i} = \frac{\partial}{\partial s_i} \theta(\overrightarrow{\boldsymbol{\omega}_i}) - \frac{\partial}{\partial s_i} \theta(\mathrm{S}(\overleftarrow{\boldsymbol{\omega}_i}, \mathbf{n}_i, \eta_i)) \tag{A.5}$$

$$\frac{\partial \mathbf{c}_i}{\partial s_{i+1}} = \frac{\partial}{\partial s_{i+1}} \theta(\overrightarrow{\boldsymbol{\omega}_i}) \tag{A.6}$$

Spherical coordinates derivatives.

$$\frac{\partial}{\partial s}\theta(\boldsymbol{\omega}) = \frac{\partial}{\partial s}\arccos(\omega_z) = -\frac{1}{\sqrt{1-\omega_z^2}}\frac{\partial \omega_z}{\partial s} \tag{A.7}$$

$$\frac{\partial}{\partial s}\phi(\boldsymbol{\omega}) = \frac{\partial}{\partial s}\mathrm{arctan2}(\omega_y, \omega_x) = \frac{\omega_x \frac{\partial \omega_y}{\partial s} - \omega_y \frac{\partial \omega_x}{\partial s}}{\omega_x^2 + \omega_y^2} \tag{A.8}$$

Direction derivatives.

$$\begin{aligned}
\frac{\partial \overrightarrow{\boldsymbol{\omega}_i}}{\partial s_{i-1}} &= \frac{\partial}{\partial s_{i-1}} \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|} = 0\\
\frac{\partial \overrightarrow{\boldsymbol{\omega}_i}}{\partial s_{i}} &= \frac{\partial}{\partial s_{i}} \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|} = \frac{-1}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|}\left[\frac{\partial \mathbf{x}_i}{\partial s_i} - \overrightarrow{\boldsymbol{\omega}_i}\left(\overrightarrow{\boldsymbol{\omega}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial s_i}\right)\right]\\
\frac{\partial \overrightarrow{\boldsymbol{\omega}_i}}{\partial s_{i+1}} &= \frac{\partial}{\partial s_{i+1}} \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|} = \frac{1}{\|\mathbf{x}_{i+1} - \mathbf{x}_i\|}\left[\frac{\partial \mathbf{x}_{i+1}}{\partial s_{i+1}} - \overrightarrow{\boldsymbol{\omega}_i}\left(\overrightarrow{\boldsymbol{\omega}_i} \cdot \frac{\partial \mathbf{x}_{i+1}}{\partial s_{i+1}}\right)\right]
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial \overleftarrow{\boldsymbol{\omega}_i}}{\partial s_{i-1}} &= \frac{\partial}{\partial s_{i-1}} \frac{\mathbf{x}_{i-1} - \mathbf{x}_i}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|} = \frac{1}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|}\left[\frac{\partial \mathbf{x}_{i-1}}{\partial s_{i-1}} - \overleftarrow{\boldsymbol{\omega}_i}\left(\overleftarrow{\boldsymbol{\omega}_i} \cdot \frac{\partial \mathbf{x}_{i-1}}{\partial s_{i-1}}\right)\right]\\
\frac{\partial \overleftarrow{\boldsymbol{\omega}_i}}{\partial s_{i}} &= \frac{\partial}{\partial s_{i}} \frac{\mathbf{x}_{i-1} - \mathbf{x}_i}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|} = \frac{-1}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|}\left[\frac{\partial \mathbf{x}_i}{\partial s_i} - \overleftarrow{\boldsymbol{\omega}_i}\left(\overleftarrow{\boldsymbol{\omega}_i} \cdot \frac{\partial \mathbf{x}_i}{\partial s_i}\right)\right]\\
\frac{\partial \overleftarrow{\boldsymbol{\omega}_i}}{\partial s_{i+1}} &= \frac{\partial}{\partial s_{i+1}} \frac{\mathbf{x}_{i-1} - \mathbf{x}_i}{\|\mathbf{x}_{i-1} - \mathbf{x}_i\|} = 0
\end{aligned}$$

Here, we used the fact that $\partial \mathbf{x}_i / \partial s_j = 0$ for all $i \neq j$ together with the identity

$$\frac{\partial}{\partial s}\frac{\mathbf{v}(s)}{\|\mathbf{v}(s)\|} = \frac{1}{\|\mathbf{v}(s)\|}\left[\frac{\partial \mathbf{v}(s)}{\partial s} - \frac{\mathbf{v}(s)}{\|\mathbf{v}(s)\|}\left(\frac{\mathbf{v}(s)}{\|\mathbf{v}(s)\|} \cdot \frac{\partial \mathbf{v}(s)}{\partial s}\right)\right] \tag{A.9}$$

that gives the derivative of a normalized vector.

Reflection and refraction derivatives. The derivatives of the laws of reflection and refraction are

$$\frac{\partial}{\partial s_j}\mathrm{S}^{\mathrm{r}}(\boldsymbol{\omega}, \mathbf{n}_i) = 2\left[Q_{ij}\mathbf{n}_i + (\boldsymbol{\omega} \cdot \mathbf{n}_i)\frac{\partial \mathbf{n}_i}{\partial s_j}\right] - \frac{\partial \boldsymbol{\omega}}{\partial s_j} \tag{A.10}$$

and

$$\begin{aligned}
\frac{\partial}{\partial s_j}\mathrm{S}^{\mathrm{t}}(\boldsymbol{\omega}, \mathbf{n}_i, \eta_i) = &-\eta_i\left[\frac{\partial \boldsymbol{\omega}}{\partial s_j} - \left(Q_{ij}\mathbf{n}_i + (\boldsymbol{\omega} \cdot \mathbf{n}_i)\frac{\partial \mathbf{n}_i}{\partial s_j}\right)\right] &\text{(A.11)}\\
&-\left[\frac{\partial \mathbf{n}_i}{\partial s_j}R_i + \mathbf{n}_i\left(\frac{1}{2R_i}\left(-\eta_i^2\left(-2(\boldsymbol{\omega} \cdot \mathbf{n}_i)Q_{ij}\right)\right)\right)\right] &\text{(A.12)}
\end{aligned}$$

respectively, with

$$Q_{ij} = \left(\frac{\partial \boldsymbol{\omega}}{\partial s_j} \cdot \mathbf{n}_i\right) + \left(\boldsymbol{\omega} \cdot \frac{\partial \mathbf{n}_i}{\partial s_j}\right) \tag{A.13}$$

$$R_i = \sqrt{1 - \eta_i^2(1 - (\boldsymbol{\omega} \cdot \mathbf{n}_i)^2)}. \tag{A.14}$$

Here, depending on the $\boldsymbol{\omega}$ argument and $j \in \{i-1, i, i+1\}$, many of the terms simplify considerably. Note also that $\partial \mathbf{n}_i / \partial s_j = 0$ for all $i \neq j$, so some of the terms evaluate to zero.

B | Derivation of the differential microfacet sampling

Section 4.3.1 presented the derivative of standard microfacet distributions with respect to their roughness parameter α:

$$\partial_\alpha \left[D_{\mathrm{GGX}}(\theta,\alpha)\cdot\cos\theta\right] = \frac{4\alpha\tan\theta(\tan^2\theta-\alpha^2)}{\cos^2\theta(\alpha^2+\tan^2\theta)^3}, \tag{B.1}$$

$$\partial_\alpha \left[D_{\mathrm{Beck.}}(\theta,\alpha)\cdot\cos\theta\right] = \frac{4e^{-\frac{\tan^2\theta}{\alpha^2}}\tan\theta(\tan^2\theta-\alpha^2)}{\alpha^5\cos^2\theta}. \tag{B.2}$$

These are both functions that consist of a positive and negative lobe and have their zero crossing at $\theta_0 = \arctan(\alpha)$. Both lobes have equal area and can be individually normalized using normalization constants $N_{\mathrm{dGGX}} = {}^{1}/_{(2\alpha)}$ and $N_{\mathrm{dBeck.}} = {}^{2}/_{(\alpha\, e)}$ respectively. From this we get two densities each that are proportional to the absolute value of the derivatives, see Figure 4.15:

$$p^-_{\mathrm{dGGX}}(\theta) = -\hat{p}_{\mathrm{dGGX}}(\theta) \quad \text{and} \quad p^+_{\mathrm{dGGX}}(\theta) = \hat{p}_{\mathrm{dGGX}}(\theta) \tag{B.3}$$

with

$$\hat{p}_{\mathrm{dGGX}}(\theta) = \frac{8\alpha^2\tan\theta(\tan^2\theta-\alpha^2)}{\cos^2\theta(\alpha^2+\tan^2\theta)^3}, \tag{B.4}$$

and

$$p^-_{\mathrm{dBeck.}}(\theta) = -\hat{p}_{\mathrm{dBeck.}}(\theta) \quad \text{and} \quad p^+_{\mathrm{dBeck.}}(\theta) = \hat{p}_{\mathrm{dBeck.}}(\theta) \tag{B.5}$$

with

$$\hat{p}_{\mathrm{dBeck.}}(\theta) = \frac{2e^{1-\frac{\tan^2\theta}{\alpha^2}}\tan\theta(\tan^2\theta-\alpha^2)}{\alpha^4\cos^2\theta}. \tag{B.6}$$

Because of our use of antithetic sampling, we will actually sample from these positive and negative densities separately. Here, we show how this can be achieved using inverse transform sampling (Section 2.4.3).

We first integrate the derivative with respect to α over elevation angles θ to obtain associated CDFs:

$$P^-_{\mathrm{dGGX}}(\theta) = \int_0^\theta p^-_{\mathrm{dGGX}}(\theta')\,\mathrm{d}\theta' = \frac{4\alpha^2\sin^2(2\theta)}{\left((\alpha^2-1)\cos(2\theta)+\alpha^2+1\right)^2}, \tag{B.7}$$

$$P^+_{\mathrm{dGGX}}(\theta) = \int_{\theta_0}^\theta p^+_{\mathrm{dGGX}}(\theta')\,\mathrm{d}\theta' = \frac{\left((\alpha^2+1)\cos(2\theta)+\alpha^2-1\right)^2}{4\left((\alpha^2-1)\cos^2\theta+1\right)^2} \tag{B.8}$$

and

$$P^{-}_{\text{dBeck.}}(\theta) = \int_0^{\theta} p^{-}_{\text{dBeck.}}(\theta')\,\mathrm{d}\theta' = \frac{e^{1-\frac{\tan^2\theta}{\alpha^2}}\tan^2\theta}{\alpha^2}, \tag{B.9}$$

$$P^{+}_{\text{dBeck.}}(\theta) = \int_{\theta_0}^{\theta} p^{+}_{\text{dBeck.}}(\theta')\,\mathrm{d}\theta' = 1 - P^{-}_{\text{dBeck.}}(\theta). \tag{B.10}$$

These can then be inverted to arrive at the sampling techniques T that suitably transform uniform random variates $u \in [0,1)$ into elevation angles θ:

$$\theta^{-}_{\text{dGGX}} = \frac{1}{2}\arctan\left(\frac{2\alpha\sqrt{u(2-2\sqrt{1-u}+2\alpha^4(1+\sqrt{1-u})-u(\alpha^2-1)^2)}}{u+4\alpha^2\sqrt{1-u}-u\alpha^4}\right), \tag{B.11}$$

$$\theta^{+}_{\text{dGGX}} = \arctan\left(\alpha\sqrt{-\left(1+\frac{2}{\sqrt{u}-1}\right)}\right), \tag{B.12}$$

and

$$\theta^{-}_{\text{dBeck.}} = \arctan\left(\alpha\sqrt{-\mathrm{W}_0\left(-\frac{u}{e}\right)}\right), \tag{B.13}$$

$$\theta^{+}_{\text{dBeck.}} = \arctan\left(\alpha\sqrt{-\mathrm{W}_{-1}\left(\frac{u-1}{e}\right)}\right). \tag{B.14}$$

Note that the Beckmann variant uses the branches $k \in \{-1, 0\}$ of the *Lambert W function* $\mathrm{W}_k(x)$ which unfortunately is not available in analytic form. We found it easiest to use a numeric evaluation based on a few iterations of a Newton solver, though alternatively, approximated implementations that can be evaluated directly are available as well [182].

References

[1] Tizian Zeltner, Iliyan Georgiev, and Wenzel Jakob. "Specular Manifold Sampling for Rendering High-Frequency Caustics and Glints". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: `10.1145/3386569.3392408`.

[2] Tizian Zeltner, Sébastien Speierer, Iliyan Georgiev, and Wenzel Jakob. "Monte Carlo Estimators for Differential Light Transport". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 40.4 (Aug. 2021). DOI: `10.1145/3450626.3459807`.

[3] Tizian Zeltner and Wenzel Jakob. "The Layer Laboratory: A Calculus for Additive and Subtractive Composition of Anisotropic Surface Reflectance". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 37.4 (July 2018). DOI: `10.1145/3197517.3201321`.

[4] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. "Mitsuba 2: A Retargetable Forward and Inverse Renderer". In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Dec. 2019). DOI: `10.1145/3355089.3356498`.

[5] Seung-Hwan Baek, Tizian Zeltner, Hyun Jin Ku, Inseung Hwang, Xin Tong, Wenzel Jakob, and Min H. Kim. "Image-Based Acquisition and Modeling of Polarimetric Reflectance". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: `10.1145/3386569.3392387`.

[6] Guillaume Loubet, Tizian Zeltner, Nicolas Holzschuch, and Wenzel Jakob. "Slope-Space Integrals for Specular Next Event Estimation". In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39.6 (Dec. 2020). DOI: `0.1145/3414685.3417811`.

[7] Eric Veach. "Robust Monte Carlo Methods for Light Transport Simulation". PhD thesis. Stanford, CA, USA, 1998.

[8] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation (3rd ed.)* 3rd. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Oct. 2016. ISBN: 9780128006450.

[9] *ART - The Advanced Rendering Toolkit.* https://cgg.mff.cuni.cz/ART/. 2018.

[10] Laurent Belcour and Pascal Barla. "A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence". In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: `10.1145/3072959.3073620`.

[11] Ling-Qi Yan, Miloš Hašan, Bruce Walter, Steve Marschner, and Ravi Ramamoorthi. "Rendering Specular Microgeometry with Wave Optics". In: *ACM Trans. Graph.* 37.4 (July 2018). DOI: `10.1145/3197517.3201351`.

[12] Mengqi (Mandy) Xia, Bruce Walter, Eric Michielssen, David Bindel, and Steve Marschner. "A Wave Optics Based Fiber Scattering Model". In: *ACM Trans. Graph.* 39.6 (Nov. 2020). DOI: `10.1145/3414685.3417841`.

[13] Shlomi Steinberg and Ling-Qi Yan. "A generic framework for physical light transport". In: *ACM Transactions on Graphics* 40.4 (Aug. 2021). DOI: `10.1145/3450626.3459791`.

[14] W.R. McCluney. *Introduction to Radiometry and Photometry, Second Edition.* Artech House applied photonics series. Artech House Publishers, 2014. ISBN: 9781608078349.

[15] Pat Hanrahan. "Rendering Concepts". In: *Radiosity and realistic image synthesis.* Ed. by Michael F. Cohen and J. R. Wallace. 1993. Chap. 2.

[16] Subrahmanyan Chandrasekhar. *Radiative Transfer.* Dover Publications, 1960.

[17] H. von Helmholtz. *Handbuch der physiologischen Optik.* Added t.-p.: Allgemeine encyklopädie der physik ... hrsg. von G. Karsten. IX bd. Voss, 1867.

[18] Michael Oren and Shree K. Nayar. "Generalization of Lambert's Reflectance Model". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH '94. New York, NY, USA: Association for Computing Machinery, 1994. DOI: `10.1145/192161.192213`.

[19] Brent Burley. "Physically-Based Shading at Disney". In: *Practical physically-based shading in film and game production, ACM SIGGRAPH 2012 Courses.* 2012.

[20] Eric Heitz and Jonathan Dupuy. *Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation.* Technical Report. 2015. URL: `https://eheitzresearch.wordpress.com/research/`.

[21] Eugene d'Eon. "An Analytic BRDF for Materials with Spherical Lambertian Scatterers". In: *Computer Graphics Forum* (2021). DOI: `10.1111/cgf.14348`.

[22] Augustin Jean Fresnel. *Mémoire sur la loi des modifications que la réflexion imprime à la lumière polarisée.* Académie des Sciences, 1823.

[23] Eugene Hecht. *Optics.* 4th. Addison-Wesley, 1998.

[24] Max Born and Emil Wolf. *Principles of Optics: 60th Anniversary Edition.* 7th ed. Cambridge University Press, 2019. DOI: `10.1017/9781108769914`.

[25] P. E. Ciddor. "Refraction into an absorbing medium". In: *American Journal of Physics* 44.8 (1976). DOI: 10.1119/1.10317.

[26] Bui Tuong Phong. "Illumination for Computer Generated Pictures". In: *Commun. ACM* 18.6 (June 1975). DOI: 10.1145/360825.360839.

[27] Petr. Beckmann and Andre. Spizzichino. *The scattering of electromagnetic waves from rough surfaces.* Pergamon Press; [distributed in the Western Hemisphere by Macmillan, New York] goford, New York, 1963.

[28] Kenneth E Torrance and Ephraim M Sparrow. "Theory for off-specular reflection from roughened surfaces". In: *Josa* 57.9 (1967). DOI: 10.1364/JOSA.57.001105.

[29] R. L. Cook and K. E. Torrance. "A Reflectance Model for Computer Graphics". In: *ACM Trans. Graph.* 1.1 (Jan. 1982). DOI: 10.1145/357290.357293.

[30] Jos Stam. "An Illumination Model for a Skin Layer Bounded by Rough Surfaces". In: *Proceedings of the 12th Eurographics Workshop on Rendering Techniques.* Berlin, Heidelberg: Springer-Verlag, 2001. DOI: 10.5555/647653.732287.

[31] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. "Microfacet Models for Refraction Through Rough Surfaces". In: *Proceedings of the 18th Eurographics Conference on Rendering Techniques.* EGSR'07. Eurographics Association, 2007. DOI: 10.2312/EGWR/EGSR07/195-206.

[32] T. S. Trowbridge and K. P. Reitz. "Average irregularity representation of a rough surface for ray reflection". In: *J. Opt. Soc. Am.* 65.5 (May 1975). DOI: 10.1364/JOSA.65.000531.

[33] M. Ribardière, B. Bringier, D. Meneveaux, and L. Simonot. "STD: Student's t-Distribution of Slopes for Microfacet Based BSDFs". In: *Comput. Graph. Forum* 36.2 (May 2017). DOI: 10.5555/3128975.3129013.

[34] Murat Kurt, László Szirmay-Kalos, and Jaroslav Křivánek. "An Anisotropic BRDF Model for Fitting and Monte Carlo Rendering". In: *SIGGRAPH Comput. Graph.* 44.1 (Feb. 2010). DOI: 10.1145/1722991.1722996.

[35] Eric Heitz. "Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs". In: *Journal of Computer Graphics Techniques (JCGT)* 3.2 (June 2014).

[36] Jonathan Dupuy. "Photorealistic Surface Rendering with Microfacet Theory". Theses. Université Claude Bernard - Lyon I ; Université de Montréal, Nov. 2015.

[37] Bruce Walter, Zhao Dong, Steve Marschner, and Donald P Greenberg. "The ellipsoid normal distribution function". In: *Supplementary material* (2016).

[38] B. Smith. “Geometrical shadowing of a random rough surface”. In: *IEEE Transactions on Antennas and Propagation* 15.5 (1967). DOI: 10.1109/TAP.1967.1138991.

[39] Eric Heitz, Johannes Hanika, Eugene d’Eon, and Carsten Dachsbacher. “Multiple-Scattering Microfacet BSDFs with the Smith Model”. In: *ACM Trans. Graph.* 35.4 (July 2016). DOI: 10.1145/2897824.2925943.

[40] Joo Ho Lee, Adrian Jarabo, Daniel S. Jeon, Diego Gutierrez, and Min H. Kim. “Practical Multiple Scattering for Rough Surfaces”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275016.

[41] Feng Xie and Pat Hanrahan. “Multiple Scattering from Distributions of Specular V-Grooves”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275078.

[42] Andrea Weidlich and Alexander Wilkie. “Arbitrarily Layered Micro-Facet Surfaces”. In: *Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia.* GRAPHITE ’07. New York, NY, USA: Association for Computing Machinery, 2007. DOI: 10.1145/1321261.1321292.

[43] Laurent Belcour. “Efficient Rendering of Layered Materials using an Atomic Decomposition with Statistical Operators”. In: *ACM Transactions on Graphics* 37.4 (2018). DOI: 10.1145/3197517.3201289.

[44] Philippe Weier and Laurent Belcour. “Rendering Layered Materials with Anisotropic Interfaces”. In: *Journal of Computer Graphics Techniques (JCGT)* 9.2 (June 2020).

[45] H. C. van de Hulst. *Multiple light scattering : tables, formulas, and applications / H. C. van de Hulst.* Academic Press New York, 1980. ISBN: 0127107010, 0127107029.

[46] Wenzel Jakob, Eugene D’Eon, Otto Jakob, and Steve Marschner. “A Comprehensive Framework for Rendering Layered Materials”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014). DOI: 10.1145/2601097.2601139.

[47] Yu Guo, Miloš Hašan, and Shaung Zhao. “Position-Free Monte Carlo Simulation for Arbitrary Layered BSDFs”. In: *ACM Trans. Graph.* 37.6 (2018). DOI: 10.1145/3272127.3275053.

[48] Mengqi (Mandy) Xia, Bruce Walter, Christophe Hery, and Steve Marschner. “Gaussian Product Sampling for Rendering Layered Materials”. In: *Computer Graphics Forum* 39.1 (2020). DOI: `10.1111/cgf.13883`.

[49] Luis E. Gamboa, Adrien Gruson, and Derek Nowrouzezahrai. “An Efficient Transport Estimator for Complex Layered Materials”. In: *Computer Graphics Forum* 39.2 (2020). DOI: `10.1111/cgf.13936`.

[50] Ibón Guillén, Julio Marco, Diego Gutierrez, Wenzel Jakob, and Adrien Jarabo. “A General Framework for Pearlescent Materials”. In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 39.6 (Nov. 2020). DOI: `10.1145/3414685.3417782`.

[51] Fred Nicodemus. “Self-Study Manual on Optical Radiation Measurements”. In: *Washington: US Department of Commerce, National Bureau of Standard (NBS), 1976, edited by Nicodemus, Fred E.* (Jan. 1976).

[52] James T. Kajiya. “The Rendering Equation”. In: *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986). DOI: `10.1145/15886.15902`.

[53] Adrian Jarabo and Victor Arellano. “Bidirectional Rendering of Vector Light Transport”. In: *Computer Graphics Forum* 37.6 (2018). DOI: `10.1111/cgf.13314`.

[54] Patrick Billingsley. *Probability and measure.* 3. ed. A Wiley-Interscience publication. New York [u.a.]: Wiley, 1995. ISBN: 0471007102.

[55] J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods.* Methuen’s monographs on applied probability and statistics. Methuen, 1964. ISBN: 9789400958203.

[56] M.H. Kalos and P.A. Whitlock. *Monte Carlo Methods.* Wiley, 2008. ISBN: 9783527407606.

[57] Art B. Owen. *Monte Carlo theory, methods and examples.* 2013. URL: `https://statweb.stanford.edu/~owen/mc/`.

[58] David Donoho. “High-Dimensional Data Analysis: The Curses and Blessings of Dimensionality”. In: *AMS Math Challenges Lecture* (Jan. 2000).

[59] T. Hachisuka. *Five Common Misconceptions about Bias in Light Transport Simulation.* 2013.

[60] Luc Devroye. *Non-Uniform Random Variate Generation.* New York: Springer-Verlag, 1986. ISBN: 978-1-4613-8643-8.

[61] E. Heitz. “Can’t Invert the CDF? The Triangle-Cut Parameterization of the Region under the Curve”. In: *Computer Graphics Forum* 39.4 (2020). DOI: `10.1111/cgf.14058`.

[62] John von Neumann. “Various Techniques Used in Connection with Random Digits”. In: *Monte Carlo Method.* Ed. by A. S. Householder, G. E. Forsythe, and H. H. Germond. Vol. 12. National Bureau of Standards Applied Mathematics Series. Washington, DC: US Government Printing Office, 1951. Chap. 13.

[63] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. “Equation of State Calculations by Fast Computing Machines”. In: *Journal of Chemical Physics* 21.6 (1953). DOI: 10.1063/1.1699114.

[64] W. Keith Hastings. “Monte Carlo sampling methods using Markov chains and their applications”. In: *Biometrika* 57.1 (1970). DOI: 10.2307/2334940.

[65] S.P. Meyn and R.L. Tweedie. *Markov Chains and Stochastic Stability.* London: Springer-Verlag, 1993. ISBN: 978-1-4471-3267-7.

[66] Eric Veach and Leonidas J. Guibas. “Optimally Combining Sampling Techniques for Monte Carlo Rendering”. In: *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques.* SIGGRAPH ’95. New York, NY, USA: Association for Computing Machinery, 1995. DOI: 10.1145/218380.218498.

[67] Kondapaneni Ivo, Petr Vévoda, Pascal Grittmann, Tomáš Skřivan, Philipp Slusallek, and Jaroslav Křivánek. “Optimal Multiple Importance Sampling”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 38.4 (July 2019). DOI: 10.1145/3306346.3323009.

[68] Rex West, Iliyan Georgiev, Adrien Gruson, and Toshiya Hachisuka. “Continuous Multiple Importance Sampling”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: 10.1145/3386569.3392436.

[69] Pascal Grittmann, Iliyan Georgiev, Philipp Slusallek, and Jaroslav Křivánek. “Variance-Aware Multiple Importance Sampling”. In: *ACM Trans. Graph. (SIGGRAPH Asia 2019)* 38.6 (2019). DOI: 10.1145/3355089.3356515.

[70] Pascal Grittmann, Iliyan Georgiev, and Philipp Slusallek. “Correlation-Aware Multiple Importance Sampling for Bidirectional Rendering Algorithms”. In: *Comput. Graph. Forum (EUROGRAPHICS 2021)* 40.2 (2021). DOI: 10.1111/cgf.142628.

[71] J. M. Hammersley and K. W. Morton. “A new Monte Carlo technique: antithetic variates”. In: *Mathematical Proceedings of the Cambridge Philosophical Society* 52.3 (1956). DOI: 10.1017/S0305004100031455.

[72] Kartic Subr, Derek Nowrouzezahrai, Wojciech Jarosz, Jan Kautz, and Kenny Mitchell. "Error analysis of estimators that use combinations of stochastic sampling strategies for direct illumination". In: *Computer Graphics Forum* 33.4 (2014). DOI: 10.1111/cgf.12416.

[73] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. "Unbiased Warped-Area Sampling for Differentiable Rendering". In: *ACM Transactions on Graphics* 39.6 (2020). DOI: 10.1145/3414685.3417833.

[74] Cheng Zhang, Zhao Dong, Michael Doggett, and Shuang Zhao. "Antithetic Sampling for Monte Carlo Differentiable Rendering". In: *ACM Trans. Graph.* 40.4 (2021). DOI: 10.1145/3450626.3459783.

[75] Michael B. Giles. "Multilevel Monte Carlo Path Simulation". In: *Oper. Res.* 56.3 (May 2008). DOI: 10.1287/opre.1070.0496.

[76] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. "Wavelet Importance Sampling: Efficiently Evaluating Products of Complex Functions". In: *ACM Trans. Graph.* 24.3 (July 2005). DOI: 10.1145/1073204.1073328.

[77] Petrik Clarberg and Tomas Akenine-Moeller. "Practical Product Importance Sampling for Direct Illumination". In: *Computer Graphics Forum* (2008). DOI: 10.1111/j.1467-8659.2008.01166.x.

[78] Wojciech Jarosz, Nathan A. Carr, and Henrik Wann Jensen. "Importance Sampling Spherical Harmonics". In: *Computer Graphics Forum (Proceedings of Eurographics)* 28.2 (Apr. 2009). DOI: 10.1111/j.1467-8659.2009.01398.x.

[79] Eric Heitz and Eugene D'Eon. "Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals". In: *Computer Graphics Forum* 33.4 (July 2014). DOI: 10.1111/cgf.12417.

[80] Gregory J. Ward. "Adaptive Shadow Testing for Ray Tracing". In: *Photorealistic Rendering in Computer Graphics.* Ed. by P. Brunet and F. W. Jansen. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994. ISBN: 978-3-642-57963-9.

[81] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. "Monte Carlo Techniques for Direct Lighting Calculations". In: *ACM Trans. Graph.* 15.1 (Jan. 1996). DOI: 10.1145/226150.226151.

[82] Alejandro Conty Estevez and Christopher Kulla. "Importance Sampling of Many Lights with Adaptive Tree Splitting". In: *Proc. ACM Comput. Graph. Interact. Tech.* 1.2 (Aug. 2018). DOI: 10.1145/3233305.

[83] Cem Yuksel. “Stochastic Lightcuts”. In: *High-Performance Graphics (HPG 2019).* Strasbourg, France: The Eurographics Association, 2019. DOI: 10.2312/hpg.20191192.

[84] Benedikt Bitterli, Chris Wyman, Matt Pharr, Peter Shirley, Aaron Lefohn, and Wojciech Jarosz. “Spatiotemporal reservoir resampling for real-time ray tracing with dynamic direct lighting”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: 10.1145/3386569.3392481.

[85] Luke Anderson, Tzu-Mao Li, Jaakko Lehtinen, and Frédo Durand. “Aether: An Embedded Domain Specific Sampling Language for Monte Carlo Rendering”. In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073704.

[86] Eric P. Lafortune and Yves D. Willems. “Bi-Directional Path Tracing”. In: *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (COMPUGRAPHICS ’93).* 1993.

[87] Eric Veach and Leonidas Guibas. “Bidirectional Estimators for Light Transport”. In: *Proceedings of Eurographics Rendering Workshop (EGWR ’94).* 1994.

[88] Henrik Wann Jensen. “Global Illumination Using Photon Maps”. In: *Proceedings of the Eurographics Workshop on Rendering Techniques ’96.* Berlin, Heidelberg: Springer-Verlag, 1996.

[89] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping.* Natick, MA, USA: A. K. Peters, Ltd., 2001. ISBN: 1-56881-147-0.

[90] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. “Progressive Photon Mapping”. In: *ACM SIGGRAPH Asia 2008 Papers.* SIGGRAPH Asia ’08. New York, NY, USA: Association for Computing Machinery, 2008. DOI: 10.1145/1457515.1409083.

[91] Toshiya Hachisuka and Henrik Wann Jensen. “Stochastic Progressive Photon Mapping”. In: *ACM Trans. Graph.* 28.5 (Dec. 2009). DOI: 10.1145/1618452.1618487.

[92] Hao Qin, Xin Sun, Qiming Hou, Baining Guo, and Kun Zhou. “Unbiased Photon Gathering for Light Transport Simulation”. In: *ACM Trans. Graph.* 34.6 (Oct. 2015). DOI: 10.1145/2816795.2818119.

[93] Toshiya Hachisuka and Henrik Wann Jensen. “Robust Adaptive Photon Tracing Using Photon Path Visibility”. In: *ACM Trans. Graph.* 30.5 (Oct. 2011). DOI: 10.1145/2019627.2019633.

[94] Alejandro Conty Estevez and Christopher Kulla. “Practical Caustics Rendering with Adaptive Photon Guiding”. In: *ACM SIGGRAPH 2020 Talks*. New York, NY, USA: Association for Computing Machinery, 2020. DOI: 10.1145/3388767.3407370.

[95] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. “Light Transport Simulation with Vertex Connection and Merging”. In: *ACM Trans. Graph.* 31.6 (Nov. 2012). DOI: 10.1145/2366145.2366211.

[96] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. “A Path Space Extension for Robust Light Transport Simulation”. In: 31.6 (Nov. 2012). DOI: 10.1145/2366145.2366210.

[97] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. “A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm”. In: *Computer Graphics Forum* 21.3 (2002). DOI: 10.1111/1467-8659.t01-1-00703.

[98] Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. “Multiplexed Metropolis Light Transport”. In: *ACM Trans. Graph.* 33.4 (July 2014). DOI: 10.1145/2601097.2601138.

[99] Benedikt Bitterli, Wenzel Jakob, Jan Novák, and Wojciech Jarosz. “Reversible Jump Metropolis Light Transport Using Inverse Mappings”. In: *ACM Transactions on Graphics* 37.1 (Oct. 2017). DOI: 10.1145/3132704.

[100] Hisanari Otsu, Anton S. Kaplanyan, Johannes Hanika, Carsten Dachsbacher, and Toshiya Hachisuka. “Fusing State Spaces for Markov Chain Monte Carlo Rendering”. In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073691.

[101] Jacopo Pantaleoni. “Charted Metropolis Light Transport”. In: *ACM Trans. Graph.* 36.4 (July 2017). DOI: 10.1145/3072959.3073677.

[102] Adrien Gruson, Rex West, and Toshiya Hachisuka. “Stratified Markov Chain Monte Carlo Light Transport”. In: *Computer Graphics Forum* (2020). DOI: 10.1111/cgf.13935.

[103] Benedikt Bitterli and Wojciech Jarosz. “Selectively Metropolised Monte Carlo light transport simulation”. In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356578.

[104] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. "On-Line Learning of Parametric Mixture Models for Light Transport Simulation". In: *ACM Trans. Graph.* 33.4 (July 2014). DOI: `10.1145/2601097.2601203`.

[105] Thomas Müller, Markus Gross, and Jan Novák. "Practical Path Guiding for Efficient Light-Transport Simulation". In: *Computer Graphics Forum* 36.4 (2017). DOI: `10.1111/cgf.13227`.

[106] Florian Reibold, Johannes Hanika, Alisa Jung, and Carsten Dachsbacher. "Selective Guided Sampling with Complete Light Transport Paths". In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: `10.1145/3272127.3275030`.

[107] Thomas Müller, Brian Mcwilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. "Neural Importance Sampling". In: *ACM Trans. Graph.* 38.5 (Oct. 2019). ISSN: 0730-0301. DOI: `10.1145/3341156`.

[108] Jiří Vorba, Johannes Hanika, Sebastian Herholz, Thomas Müller, Jaroslav Křivánek, and Alexander Keller. "Path Guiding in Production". In: *ACM SIGGRAPH 2019 Courses*. SIGGRAPH '19. New York, NY, USA: ACM, 2019. DOI: `10.1145/3305366.3328091`.

[109] Laurent Hascoet and Valérie Pascual. "The Tapenade Automatic Differentiation Tool: Principles, Model, and Specification". In: *ACM Trans. Math. Softw.* 39.3 (May 2013). DOI: `10.1145/2450153.2450158`.

[110] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[111] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. *JAX: composable transformations of Python + NumPy programs*. Version 0.2.5. 2018. URL: `http://github.com/google/jax`.

[112] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Curran Associates, Inc., 2019.

[113] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. "DiffTaichi: Differentiable Programming for Physical Simulation". In: *ICLR* (2020).

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

[115] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. USA: Society for Industrial and Applied Mathematics, 2008. ISBN: 0898716594.

[116] Wenzel Jakob and Steve Marschner. "Manifold Exploration: A Markov Chain Monte Carlo Technique for Rendering Scenes with Difficult Specular Transport". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 31.4 (July 2012). DOI: 10.1145/2185520.2185554.

[117] Anton Kaplanyan and Carsten Dachsbacher. "Path Space Regularization for Holistic and Robust Light Transport". In: *Computer Graphics Forum* 32 (May 2013). DOI: 10.1111/cgf.12026.

[118] Philippe Weier, Marc Droske, Johannes Hanika, Andrea Weidlich, and Jiří Vorba. "Optimised Path Space Regularisation". In: *Computer Graphics Forum* 40.4 (2021). DOI: 10.1111/cgf.14347.

[119] Don Mitchell and Pat Hanrahan. "Illumination from Curved Reflectors". In: *SIGGRAPH Comput. Graph.* 26.2 (July 1992). DOI: 10.1145/142920.134082.

[120] Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. "Single Scattering in Refractive Media with Triangle Mesh Boundaries". In: *ACM Transactions on Graphics* 28.3 (Aug. 2009). DOI: 10.1145/1531326.1531398.

[121] Binh-Son Hua, Adrien Gruson, Victor Petitjean, Matthias Zwicker, Derek Nowrouzezahrai, Elmar Eisemann, and Toshiya Hachisuka. "A Survey on Gradient-Domain Rendering". In: *Computer Graphics Forum* 38.2 (2019). DOI: 10.1111/cgf.13652.

[122] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. "A first-order analysis of lighting, shading, and shadows". In: *ACM Transactions on Graphics (TOG)* 26.1 (2007). DOI: 10.1145/1189762.1189764.

[123] G J Ward and P S Heckbert. *Irradiance gradients*. Tech. rep. Lawrence Berkeley Lab., CA (United States); Ecole Polytechnique Federale, Lausanne (Switzerland); Technische Hogeschool Delft (Netherlands). Dept. of Technical Mathematics and Informatics, Apr. 1992.

[124] Jaroslav Krivánek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. "Radiance caching for efficient global illumination computation". In: *IEEE Transactions on Visualization and Computer Graphics* 11.5 (2005). DOI: 10.1109/TVCG.2005.83.

[125] Paul S Heckbert. "Fundamentals of texture mapping and image warping". MA thesis. University of California, Berkeley, 1989.

[126] Homan Igehy. "Tracing Ray Differentials". In: *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999. DOI: 10.1145/311535.311555.

[127] Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. "5D Covariance Tracing for Efficient Defocus and Motion Blur". In: *ACM Transactions on Graphics* 32.3 (July 2013). DOI: 10.1145/2487228.2487239.

[128] Wenzel Jakob. "Light transport on path-space manifolds". PhD thesis. Cornell University, 2013.

[129] Anton Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. "The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation". In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (Aug. 2014). DOI: 10.1145/2601097.2601108.

[130] Johannes Hanika, Anton Kaplanyan, and Carsten Dachsbacher. "Improved Half Vector Space Light Transport". In: *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34.4 (June 2015). DOI: 10.1111/cgf.12679.

[131] Min Chen and James Arvo. "Theory and Application of Specular Path Perturbation". In: *ACM Trans. Graph.* 19.4 (Oct. 2000). DOI: 10.1145/380666.380670.

[132] Johannes Hanika, Marc Droske, and Luca Fascione. "Manifold Next Event Estimation". In: *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)* 34.4 (June 2015). DOI: 10.1111/cgf.12681.

[133] Sebastien Speierer, Christophe Hery, Ryusuke Villemin, and Wenzel Jakob. "Caustic Connection Strategies for Bidirectional Path Tracing". In: *Pixar Technical Memo #18-01* (2018).

[134] David Koerner, Jan Novák, Peter Kutz, Ralf Habel, and Wojciech Jarosz. "Subdivision Next-Event Estimation for Path-Traced Subsurface Scattering". In: *Proceedings of the Eurographics Symposium on Rendering: Experimental Ideas & Implementations.* EGSR '16. 2016. DOI: 10.2312/sre.20161214.

[135] Pascal Weber, Johannes Hanika, and Carsten Dachsbacher. "Multiple Vertex Next Event Estimation for Lighting in Dense, Forward-Scattering Media". In: *Comput. Graph. Forum* 36.2 (May 2017). DOI: 10.1111/cgf.13103.

[136] Ling-Qi Yan, Miloš Hašan, Wenzel Jakob, Jason Lawrence, Steve Marschner, and Ravi Ramamoorthi. "Rendering Glints on High-Resolution Normal-Mapped Specular Surfaces". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (2014). DOI: 10.1145/2601097.2601155.

[137] Ling-Qi Yan, Miloš Hašan, Steve Marschner, and Ravi Ramamoorthi. "Position-Normal Distributions for Efficient Rendering of Specular Microstructure". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 35.4 (2016). DOI: 10.1145/2897824.2925915.

[138] Asen Atanasov, Alexander Wilkie, Vladimir Koylazov, and Jaroslav Křivánek. "A Multiscale Microfacet Model Based on Inverse Bin Mapping". In: *Computer Graphics Forum* 40.2 (2021). DOI: 10.1111/cgf.142618.

[139] Wenzel Jakob, Miloš Hašan, Ling-Qi Yan, Ravi Ramamoorthi, and Steve Marschner. "Discrete Stochastic Microfacet Models". In: *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 33.4 (July 2014). DOI: 10.1145/2601097.2601186.

[140] Asen Atanasov and Vladimir Koylazov. "A Practical Stochastic Algorithm for Rendering Mirror-like Flakes". In: *ACM SIGGRAPH 2016 Talks.* SIGGRAPH '16. New York, NY, USA: Association for Computing Machinery, 2016. DOI: 10.1145/2897839.2927391.

[141] Alexandr Kuznetsov, Miloš Hašan, Zexiang Xu, Ling-Qi Yan, Bruce Walter, Nima Khademi Kalantari, Steve Marschner, and Ravi Ramamoorthi. “Learning Generative Models for Rendering Specular Microgeometry”. In: *ACM Trans. Graph.* 38.6 (Nov. 2019). DOI: 10.1145/3355089.3356525.

[142] Beibei Wang, Miloš Hašan, Nicolas Holzschuch, and Ling-Qi Yan. “Example-Based Microstructure Rendering with Constant Storage”. In: *ACM Trans. Graph.* 39.5 (Aug. 2020). DOI: 10.1145/3406836.

[143] X. Chermain, B. Sauvage, J.-M. Dischler, and C. Dachsbacher. “Procedural Physically based BRDF for Real-Time Rendering of Glints”. In: *Computer Graphics Forum* 39.7 (2020). DOI: 10.1111/cgf.14141.

[144] Xavier Chermain, Simon Lucas, Basile Sauvage, Jean-Michel Dischler, and Carsten Dachsbacher. “Real-Time Geometric Glint Anti-Aliasing with Normal Map Filtering”. In: *Proc. ACM Comput. Graph. Interact. Tech.* 4.1 (Apr. 2021). DOI: 10.1145/3451257.

[145] Luis E. Gamboa, Jean-Philippe Guertin, and Derek Nowrouzezahrai. “Scalable Appearance Filtering for Complex Lighting Effects”. In: *ACM Trans. Graph.* 37.6 (Dec. 2018). DOI: 10.1145/3272127.3275058.

[146] Adithya Pediredla, Yasin Karimi Chalmiani, Matteo Giuseppe Scopelliti, Maysamreza Chamanzar, Srinivasa Narasimhan, and Ioannis Gkioulekas. “Path Tracing Estimators for Refractive Radiative Transfer”. In: *ACM Trans. Graph.* 39.6 (Nov. 2020). DOI: 10.1145/3414685.3417793.

[147] Marco Ament, Christoph Bergmann, and Daniel Weiskopf. “Refractive Radiative Transfer Equation”. In: *ACM Trans. Graph.* 33.2 (Apr. 2014). DOI: 10.1145/2557605.

[148] Beibei Wang, Miloš Hašan, and Ling-Qi Yan. “Path Cuts: Efficient Rendering of Pure Specular Light Transport”. In: *ACM Trans. Graph.* 39.6 (Nov. 2020). DOI: 10.1145/3414685.3417792.

[149] M. Spivak. *Calculus On Manifolds: A Modern Approach To Classical Theorems Of Advanced Calculus.* Mathematics monograph series. Avalon Publishing, 1971. ISBN: 9780813346120.

[150] Matt Jen-Yuan Chiang and Brent Burley. “Plausible Iris Caustics and Limbal Arc Rendering”. In: *ACM SIGGRAPH 2018 Talks.* SIGGRAPH ’18. ACM, 2018. DOI: 10.1145/3214745.3214751.

[151] John Hubbard, Dierk Schleicher, and Scott Sutherland. “How to Find All Roots of Complex Polynomials by Newton’s Method”. In: *Inventiones mathematicae* 146 (Oct. 2001). DOI: `10.1007/s002220100149`.

[152] Thomas E. Booth. “Unbiased Monte Carlo Estimation of the Reciprocal of an Integral”. In: *Nuclear Science and Engineering* 156.3 (2007). DOI: `10.13182/NSE07-A2707`.

[153] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. “Gradient-Domain Metropolis Light Transport”. In: *ACM Trans. Graph.* 32.4 (July 2013). DOI: `10.1145/2461912.2461943`.

[154] Marc Olano and Dan Baker. “LEAN Mapping”. In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.* I3D ’10. Association for Computing Machinery, 2010. DOI: `10.1145/1730804.1730834`.

[155] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. “Radiative Backpropagation: An Adjoint Method for Lightning-Fast Differentiable Rendering”. In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 39.4 (July 2020). DOI: `10.1145/3386569.3392406`.

[156] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. “Reparameterizing discontinuous integrands for differentiable rendering”. In: *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Dec. 2019). DOI: `10.1145/3355089.3356510`.

[157] Matthew M Loper and Michael J Black. “OpenDR: An approximate differentiable renderer”. In: *European Conference on Computer Vision.* Springer. 2014. DOI: `10.1007/978-3-319-10584-0_11`.

[158] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. “A Versatile Scene Model with Differentiable Visibility Applied to Generative Pose Estimation”. In: *Proceedings of ICCV 2015.* 2015. DOI: `10.1109/ICCV.2015.94`.

[159] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. “Neural 3D Mesh Renderer”. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018.* Computer Vision Foundation / IEEE Computer Society, 2018. DOI: `10.1109/CVPR.2018.00411`.

[160] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. *Soft Rasterizer: Differentiable Rendering for Unsupervised Single-View Mesh Reconstruction.* 2019. arXiv: 1901.05567 [cs.CV].

[161] Felix Petersen, Amit H. Bermano, Oliver Deussen, and Daniel Cohen-Or. *Pix2Vex: Image-to-Geometry Reconstruction using a Smooth Differentiable Renderer.* 2019. arXiv: 1903.11149 [cs.CV].

[162] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. "Modular Primitives for High-Performance Differentiable Rendering". In: *ACM Transactions on Graphics* 39.6 (2020). DOI: 10.1145/3414685.3417861.

[163] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. "Path-Space Differentiable Rendering". In: *ACM Transactions on Graphics* 39.4 (2020). DOI: 10.1145/3386569.3392383.

[164] Cheng Zhang, Zihan Yu, and Shuang Zhao. "Path-Space Differentiable Rendering of Participating Media". In: *ACM Trans. Graph.* 40.4 (2021). DOI: 10.1145/3450626.3459782.

[165] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. "Differentiable Monte Carlo Ray Tracing through Edge Sampling". In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 37.6 (2018). DOI: 10.1145/3272127.3275109.

[166] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. "Inverse Path Tracing for Joint Material and Lighting Estimation". In: *Proceedings of Computer Vision and Pattern Recognition (CVPR), IEEE* (2019). DOI: 10.1109/CVPR.2019.00255.

[167] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. "Inverse Volume Rendering with Material Dictionaries". In: *ACM Transactions on Graphics* 32.6 (Nov. 2013). DOI: 10.1145/2508363.2508377.

[168] Ioannis Gkioulekas, Anat Levin, and Todd Zickler. "An evaluation of computational imaging techniques for heterogeneous inverse scattering". In: *European Conference on Computer Vision.* Springer. 2016. DOI: 10.1007/978-3-319-46487-9_42.

[169] Shaung Zhao, Lifan Wu, Frédo Durand, and Ravi Ramamoorthi. "Downsampling Scattering Parameters for Rendering Anisotropic Media". In: *ACM Transactions on Graphics* 35.6 (2016). DOI: 10.1145/2980179.2980228.

[170] Chengqian Che, Fujun Luan, Shuang Zhao, Kavita Bala, and Ioannis Gkioulekas. *Inverse Transport Networks.* 2018. arXiv: `1809.10820 [cs.CV]`.

[171] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. “A Differential Theory of Radiative Transfer”. In: *ACM Transactions on Graphics* 38.6 (2019). DOI: `10.1145/3355089.3356522`.

[172] Yang Zhou, Lifan Wu, Ravi Ramamoorthi, and Ling-Qi Yan. “Vectorization for Fast, Analytic, and Differentiable Visibility”. In: *ACM Trans. Graph.* 40.3 (July 2021). DOI: `10.1145/3452097`.

[173] Paul S. Heckbert and Pat Hanrahan. “Beam Tracing Polygonal Objects”. In: *SIGGRAPH Comput. Graph.* 18.3 (Jan. 1984). DOI: `10.1145/964965.808588`.

[174] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. “Path Replay Backpropagation: Differentiating Light Paths using Constant Memory and Linear Time”. In: *Transactions on Graphics (Proceedings of SIGGRAPH)* 40.4 (Aug. 2021). DOI: `10.1145/3450626.3459804`.

[175] Iván Lux and Lázló Koblinger. *Monte Carlo Particle Transport Methods: Neutron and Photon Calculations.* Boston: CRC Press, 1990. ISBN: 9781351074834.

[176] Carole K. Hayakawa, Jerome Spanier, Frédéric Bevilacqua, Andrew K. Dunn, Joon S. You, Bruce J. Tromberg, and Vasan Venugopalan. “Perturbation Monte Carlo methods to solve inverse photon migration problems in heterogeneous tissues”. In: *Opt. Lett.* 26.17 (Sept. 2001). DOI: `10.1364/ol.26.001335`.

[177] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015.

[178] Samuli Laine, Tero Karras, and Timo Aila. “Megakernels Considered Harmful: Wavefront Path Tracing on GPUs”. In: *Proceedings of the 5th High-Performance Graphics Conference.* HPG ’13. New York, NY, USA: Association for Computing Machinery, 2013. DOI: `10.1145/2492045.2492060`.

[179] H. Kaufman. “The Mathematical Theory of Optimal Processes, by L. S. Pontryagin, V. G. Boltyanskii, R. V. Gamkrelidze, and E. F. Mishchenko. Authorized Translation from the Russian. Translator: K. N. Trirogoff, Editor: L.. W. Neustadt. Interscience Publishers (division of John Wiley and Sons, Inc. , New York) 1962.

viii 360 pages.” In: *Canadian Mathematical Bulletin* 7.3 (1964). doi: 10.1017/S0008439500032112.

[180] Joel Andersson. “A General-Purpose Software Framework for Dynamic Optimization (Een algemene softwareomgeving voor dynamische optimalisatie)”. PhD thesis. 2013.

[181] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. “OptiX: A General Purpose Ray Tracing Engine”. In: *ACM Transactions on Graphics* 29.4 (July 2010). doi: 10.1145/1778765.1778803.

[182] Darko Veberic. *Having Fun with Lambert W(x) Function.* 2018. arXiv: 1003.1628 [cs.MS].