LIGHT TRANSPORT ON PATH-SPACE MANIFOLDS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Wenzel Alban Jakob

August 2013

LIGHT TRANSPORT ON PATH-SPACE MANIFOLDS

Wenzel Alban Jakob, Ph.D.

Cornell University 2013

The pervasive use of computer-generated graphics in our society has led to strict demands on their visual realism. Generally, users of rendering software want their images to look, in various ways, "real", which has been a key driving force towards methods that are based on the physics of light transport.

Until recently, industrial practice has relied on a different set of methods that had comparatively little rigorous grounding in physics—but within the last decade, advances in rendering methods and computing power have come together to create a sudden and dramatic shift, in which physics-based methods that were formerly thought impractical have become the standard tool. As a consequence, considerable attention is now devoted towards making these methods as robust as possible.

In this context, robustness refers to an algorithm's ability to process arbitrary input without large increases of the rendering time or degradation of the output image. One particularly challenging aspect of robustness entails simulating the precise interaction of light with all the materials that comprise the input scene. This dissertation focuses on one specific group of materials that has fundamentally been the most important source of difficulties in this process.

Specular materials, such as glass windows, mirrors or smooth coatings (e.g. on finished wood), account for a significant percentage of the objects that surround us every day. It is perhaps surprising, then, that it is not well-understood how they can be accommodated within the theoretical framework that underlies some of the most sophisticated rendering methods available today.

Many of these methods operate using a theoretical framework known as

path space integration. But this framework makes no provisions for specular materials: to date, it is not clear how to write down a path space integral involving something as simple as a piece of glass.

Although implementations can in practice still render these materials by side-stepping limitations of the theory, they often suffer from unusably slow convergence; improvements to this situation have been hampered by the lack of a thorough theoretical understanding.

We address these problems by developing a new theory of path-space light transport which, for the first time, cleanly incorporates specular scattering into the standard framework. Most of the results obtained in the analysis of the ideally smooth case can also be generalized to rendering of glossy materials and volumetric scattering so that this dissertation also provides a powerful new set of tools for dealing with them.

The basis of our approach is that each specular material interaction locally collapses the dimension of the space of light paths so that all relevant paths lie on a submanifold of path space. We analyze the high-dimensional differential geometry of this submanifold and use the resulting information to construct an algorithm that is able to "walk" around on it using a simple and efficient equation-solving iteration.

This manifold walking algorithm then constitutes the key operation of a new type of Markov Chain Monte Carlo (MCMC) rendering method that computes lighting through very general families of paths that can involve arbitrary combinations of specular, near-specular, glossy, and diffuse surface interactions as well as isotropic or highly anisotropic volume scattering. We demonstrate our implementation on a range of challenging scenes and evaluate it against previous methods.

BIOGRAPHICAL SKETCH

Wenzel Jakob was born on November 4th, 1983 in Karlsruhe, Germany. After graduating from Helmholtz-Gymnasium, Karlsruhe, he began a dual mathematics and computer science course of studies at the Karlsruhe Institute of Technology (KIT) in 2004. Following completion of the Vordiplom (intermediate diploma) in 2006, he obtained an M.Eng. degree in 2008 at Cornell University's computer science program and subsequently joined the department as a Ph. D. student.

ACKNOWLEDGMENTS

I would like to begin by expressing my deep gratitude to my advisor Steve Marschner, who served as a continuous source of inspiration and guidance during my studies at Cornell. His incredible instinct for solving problems and positive attitude towards all things in life are qualities I shall aspire to for the rest of my life.

I enjoyed many hours of fruitful discussions with Kavita Bala, whose enthusiasm and differing viewpoint was invaluable in fleshing out new ideas and writing papers in a way that makes them accessible. It was a pleasure to talk to Bruce Walter, whose door was always open for me and the countless rendering-related questions I would tend to bring.

I am grateful to Andrew Myers for exposing me the fun world of optimizing compilers, to Michael Nussbaum for making me appreciate mathematical statistics, and to Doug James for coding assignments where things bounce around. In my last year, I was privileged to take an intensive Japanese language course run to utter perfection, for which I am indebted to Robert Sukle and Misako Terashima Chapman. Special thanks go to Becky Stewart for the feat of keeping the Ph.D. program running with a smile.

Steve Marschner, Bruce Walter, and Eugene D'Eon donated a huge amount of their personal time to proofread drafts of this dissertation. I thank them for their many comments and suggestions for improvements.

This dissertation would not have been possible if not for the stimulating environment created by my fellow students of the Cornell graphics group, in particular: Jonathan Moon, Jonathan Kaldor, Shuang Zhao, Edgar Velázquez-Armendáriz, and Chanxi Zheng.

I thank my wonderful fiancée Olesya for the great deal of love, support, and patience she has afforded to me over the many years of my absence. She has made the completion of this thesis something to truly look forward to. Olesya also deserves much credit for meticulously creating the example scenes used in the results section.

Finally, I am deeply obliged to my parents Otto and Veronika, who have ceaselessly fostered my developments since I was a little boy. Their passion, insight and willingness to let me take apart random electronic items I found around the house, have surely led me onto the path I still follow today.

My research has been supported by generous grants from the National Science Foundation (IIS-1011919, CCF-0347303, CCF-0541105), by Unilever Corporation, as well as the Intel Science and Technology Center for Visual Computing.

To Otto and Veronika

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

$\mathrm{d}\sigma_{\mathbf{x}}$	Solid angle measure with respect to the point $\mathbf{x}$
$\mathrm{d}\sigma^{\perp}_{\mathbf{x}}$	Projected solid angle measure w.r.t. $\mathbf{x}$ and $N(\mathbf{x})$
$\mathrm{d}A$	Area measure on surfaces
$f_s(\mathbf{x}, \omega' \to \omega)$	Bidirectional reflectance distribution function
$f_p(\mathbf{x}, \omega' \to \omega)$	Phase function of a volume
$f_j(\bar{\mathbf{x}})$	Measurement contribution function of pixel j
$\mathbf{G}$	Transport operator
$G(\mathbf{x} \leftrightarrow \mathbf{y})$	Geometric term between $\mathbf{x}$ and $\mathbf{y}$
$G(\mathbf{x} \leftrightarrow \cdots \leftrightarrow \mathbf{y})$	Generalized geometric term between $\mathbf{x}$ and $\mathbf{y}$
$\mathcal{P}$	Path space
I_j	Measurement of the radiance value arriving at pixel j
j	Index used to identify pixels in the output image
$\mathbf{K}$	Scattering operator
$\mathcal{M}$	Set of surfaces in the scene
$\mathbf{S}$	Solution operator derived from $\mathbf{G}$ and $\mathbf{K}$
S^2	Indicates integration over directions (the unit 2-sphere)
$L(\mathbf{x}, \omega)$	Radiance along the ray $(\mathbf{x}, \omega)$
$L_i(\mathbf{x}, \omega)$	Incident radiance function (when L is ambiguous)
$L_o(\mathbf{x}, \omega)$	Outgoing radiance function (when L is ambiguous)
$L_e(\mathbf{x}, \omega)$	Emitted radiance function
$N(\mathbf{x})$	Surface normal at the position $\mathbf{x} \in \mathcal{M}$
$r(\mathbf{x}, \mathbf{x}')$	Acceptance probability of a Metropolis-Hasting step
Ω	Simulation domain
Ω°	Interior of the simulation domain
$W_e(\mathbf{x}, \omega)$	Importance function of the camera
$\mathbf{x}$	Generally indicates a position in the domain Ω
$\bar{\mathbf{x}}$	Indicates a path $\bar{\mathbf{x}} = \mathbf{x}_0, \ldots, \mathbf{x}_n$
$\overrightarrow{\mathbf{xy}}$	Normalized direction vector from $\mathbf{x}$ to $\mathbf{y}$

$\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega)$	First intersected surface along the ray $(\mathbf{x}, \omega)$
$\pi(\bar{\mathbf{x}})$	Target distribution for the Metropolis-Hastings algorithm
σ_t	Extinction coefficient of a volume
σ_s	Scattering coefficient of a volume
$\tau(\mathbf{x} \leftrightarrow \mathbf{y})$	Integrated volume transmittance between $\mathbf{x}$ and $\mathbf{y}$
ω	Commonly used variable for directions

CHAPTER 1

INTRODUCTION

The central goal of physics-based rendering is the generation of photorealistic images by simulating the flow of light through increasingly elaborate mathematical models of our world. Due to the pervasive use of computer-generated graphics, the demands on these simulations have grown rapidly over the years, fueling the creation of an entire industry dedicated to realistic rendering of architecture, nature, human and animal characters, as well as a wide range of other visual phenomena.

What is truly remarkable about these developments, particularly in relation to industrial practice in the preceding decade, is that they have squarely established a new paradigm of physics-based modeling, which follows the conviction that a thorough understanding of how light interacts with an object should precede attempts to faithfully reproduce its appearance. As a result, formerly artist-driven endeavors, such as the design of a shader in a rendering system, have turned into a creative union of not only mathematics and optics, but also biology, zoology, ecology, medical physics, and other disciplines. This is not to say that the role of aesthetics has been diminished by these innovations—realized within a framework that is based on physical laws it remains a key driving force.

Of course, this physics-based approach does not stem from a surprising new insight; rather, it is the widespread availability of sufficient computational resources that has now made its pursuit feasible. At this time, is not uncommon for a company invested in computer graphics to employ one or more researchers solely devoted to the implementation and improvement of light transport models. Digital visual effects studios are an example of this trend, being

interested in objects ranging from everyday paint, metal and glass surfaces to organic substances such as the iridescent feather of a bird and the eyes and hair of humans and animals.

A less obvious aspect of the pursuit of realistic rendering is that it has created formidable technical challenges that currently limit the extent to which it can be implemented. Intuition leads us to believe that the realism of an image should improve as the mathematical model employed increasingly resembles the underlying physics: clearly, with each approximation that is removed, there is less room for modeling errors in the final result. But this view neglects the imperfect nature of the algorithms that are used to solve the equations of the resulting rendering problem. For instance, it may be desirable to give a virtual human a more vivid pair of eyes by carefully modeling cornea, iris and sclera, with their attendant index of refraction changes. Yet, as part of a virtual environment, this "enhanced" character may prove so difficult to render that it becomes entirely impractical to produce an acceptable image given any reasonable amount of time. In this way, the pursuit of more realistic models may effectively lead to inferior results.

Usually, certain classes of light paths that occur in the underlying simulation are to blame for these difficulties. Given the nature of currently used rendering algorithms, which generally rely on some form of random sampling to find light paths, problems tend to arise when an important class of paths is found with too low a probability. A particularly well-known example is that of *specular-diffuse-specular* paths, in which light is first scattered by a specular material such as a smooth conductor or a dielectric, followed by a diffuse (or other non-specular) material and then another specular interaction. This includes fairly common situations such as a tabletop seen through a drinking glass sitting on it, a bottle

containing shampoo or other translucent liquid, a shop window viewed and illuminated from outside, as well as the aforementioned ubiquitous example of scattering inside an eye. Even in scenes where these paths do not cause dramatic lighting effects, their presence can lead to excessively slow convergence in rendering algorithms that attempt to account for all transport paths.

Seen from a high level, most rendering algorithms solve a high-dimensional integration problem by randomly point-sampling an integrand that describes the amount of transported light. The presence of such "difficult" light paths leads to small regions in the integration domain where the integrand takes on high values; yet, their comparatively low volume means that only few samples are likely to be placed there, resulting in an inaccurate estimate of the integral's value. Depending on the rendering method used, these inaccuracies can manifest either as blur or as noise in the output image, quickly making it unusable as the errors increase.

To approach these challenges, this dissertation proposes a new theoretical framework and algorithms that enable a more systematic exploration of the integration domain, driven by geometric insights about the structure of these problematic paths. The foundation of this theory is a well-known physical property of light, namely that it travels along Fermat paths. Usually, these paths intuitively correspond to taking the "fastest" way possible; more generally, they are defined as having stationary optical length with respect to small variations of the path. Importantly, this principle means that the Fermat paths constitute a subset of the set of all potential light paths.

When an environment contains specular materials such as smooth dielectrics or conductors, a key observation with far-reaching implications is that this subset has a lower dimension; as we shall see later, each specular interaction effectively

collapses some of the dimensions of the space of light paths, producing a submanifold of path space. Building on this property, we propose an extended theory of path-space light transport; it is the first such theory that cleanly incorporates specular scattering, rather than relegating it to the position of an awkward corner case. This theory also has algorithmic implications that we demonstrate by developing a new rendering technique, which creates images by "walking around" on the manifold of paths informed by its high-dimensional differential geometry.

This work was initially motivated by the difficulties with rendering light paths involving perfectly specular materials, such as polished mirrors or glass. But wherever ideally specular paths are troublesome, nearly specular paths involving glossy (i.e. slightly roughened) materials are also troublesome. They can be more problematic, in fact, because they elude special mechanisms designed to handle specular interactions. These "glossy paths" have become more important as material models have evolved; finding them efficiently is a key open problem of light transport simulations.

Surprisingly, most of the results obtained in the analysis of the ideally smooth case can be generalized to rendering of glossy materials. Thus, this thesis also provides a powerful new set of tools for dealing with this very challenging class of light paths. After incorporating these generalizations, the end result of this dissertation is a Markov Chain Monte Carlo algorithm to compute lighting through very general families of paths that can involve arbitrary combinations of specular, near-specular, glossy, and diffuse surface interactions as well as isotropic or highly anisotropic volume scattering interactions.

1.1 Summary of original contributions

The work in this dissertation builds on the path-space framework and the Metropolis Light Transport algorithm proposed by Eric Veach [70]. Our contributions are as follows:

Generalized path space for surfaces and volumes. We provide a complete derivation of light transport operators and a path space integration framework that accounts for the effects of both surfaces and volumes. This leads to interesting differences compared to prior work.

Specular light transport on path-space manifolds. A key contribution of this work is an analysis of light transport involving specular paths as a manifold embedded in path space. Using this manifold representation, we develop a generalized geometric term that, for the first time, makes it possible to cleanly write down path-space integrals involving specular materials.

Manifold walking algorithm. We propose an iterative equation-solving algorithm that is able to move around on the specular manifold, making it possible to solve for configurations that lie in the neighborhood of a known path. By opting for a local rather than a global search, our algorithm is able to operate in a very general setting that makes minimal assumptions about the material type and surface representation.

Manifold Exploration. Combining our results on specular light transport and the manifold walking algorithm, we propose the *manifold perturbation*, a transition rule that explores the specular manifold to find the steady-state lighting distribution of a scene as part of a Metropolis-Hastings-type method.

This perturbation can be used in the frameworks of Metropolis Light Transport (MLT) or Energy Redistribution Path Tracing (ERPT), producing rendering algorithms with support for specular paths fundamentally built in at the core—we refer to this as *Manifold Exploration*. In equal-time comparisons on very challenging scenes, they compare favorably to previous work in Monte Carlo and Markov Chain Monte Carlo (MCMC) rendering.

1.2 Organization of the dissertation

The dissertation is divided into a total of 9 chapters and supplementary appendices. It is organized as follows: Chapter 2 presents the basics of light transport, as well as an overview of relevant prior work on global illumination rendering algorithms. In Chapter 3, we derive a path-space integration framework for surface and volume light transport that provides a foundation for the following chapters, but does not yet account for specular materials. In Chapter 4 we develop the theory of the *specular manifold*, used to handle interactions with ideal specular (polished) surfaces, and *offset specular manifolds*, which provide a graceful generalization to near-specular materials. In Chapter 5, we derive an algorithm to move from one path to another on a specular or offset specular manifold, and use it in Chapter 6 to build an algorithm that generates Markov sequences in path space as part of the Metropolis Light Transport or Energy Redistribution Path Tracing frameworks to provide methods for rendering scenes with any kind of light transport. We go on in Chapter 7 to extend the theory and algorithm to the case of participating media, and after showing results and comparisons in Chapter 8, we conclude in Chapter 9.

CHAPTER 2

BACKGROUND

In this dissertation, we first propose an extended theory of path space light transport involving specular materials and then show how to construct a practical rendering method that is based on it. Since these steps build upon a number of prior works, this chapter reviews relevant background material and introduces terminology that is used in the remainder of the dissertation. Most material is covered at a fairly high level; we refer the reader to the excellent books by Dutré et al. [10] and Pharr et al. [49] for an in-depth discussion of the theory of light transport and a wide range of different rendering techniques.

2.1 Problem setting

The principal purpose of light transport algorithms is the creation of *renderings*, usually two-dimensional images that depict a virtual environment, as if created by a hypothetical camera located in that environment. To accomplish this task, they require a complete description of the desired environment, including the placement of all objects and a characterization of their physical properties. The output image then results from a detailed simulation of all of the relevant physical laws of light, particularly *transport* and *scattering*, i.e. the propagation of light and its interaction with the materials that comprise the objects.

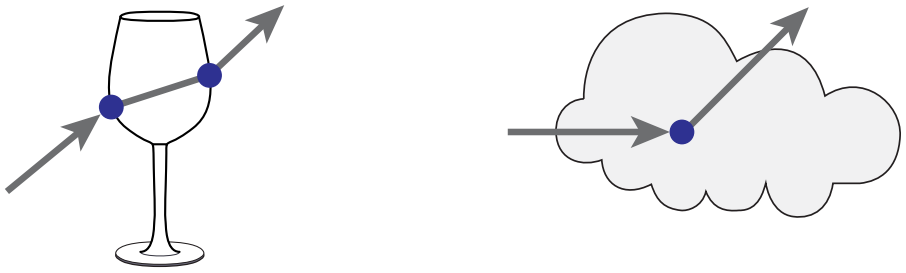
These laws exist in a hierarchy of increasing approximations, each of which can be derived from the preceding one by a simplification and, consequently, some loss of accuracy when compared to physical experiments. Quantum electrodynamics (QED), describing the interaction between the quanta of light and electric charge, currently constitutes the *de facto* top of this hierarchy. Foregoing

quantization, the next level is given by the theory of electromagnetic fields and Maxwell's equations. In the limit of small wavelengths, physical effects such as diffraction and interference become negligible and the considerably simpler laws of geometric optics emerge [41]. Even within geometric optics, further approximations exist—for instance, by neglecting the effects of polarization, or assuming that the index of refraction is piecewise constant.

Light transport simulations in computer graphics are generally located at the very bottom of this hierarchy, making use of significantly simplified variants of geometric optics. The reason for this apparent lack of rigor is that such detail is, quite simply, not needed for most rendering applications, as the assumptions underlying the approximations are satisfied without problems: the wavelength of visible light (380-780nm) is minuscule compared to the dimensions of any normal scene to be rendered, and the effects of quantization, wave properties, or polarization are usually too subtle to be visible to a human observer. Where needed, certain aspects of the more comprehensive theories (e.g. diffraction on metallic surfaces [61] or fluorescence [22]) can still be "imported" in a localized manner, while staying within the confines of geometric optics. The following section describes the specifics of the laws used in this dissertation.

2.2 Light transport model

We rely on a simplified model of geometric optics, which has become popular in computer graphics. This model assumes incoherent, unpolarized illumination that travels along a straight line until an interaction (i.e. a scattering event) occurs. In a slight abuse of terminology, this illumination is assumed to be conveyed by means of *photons*—in this context, these are idealized energy-carrying particles without their usual wave properties. Two types of photon



(a) Surface scattering (b) Volumetric scattering

Figure 2.1: We build upon geometric optics framework, which distinguishes between scattering by a *surface* (the interface of an object) and scattering by a *volume* (the interior of an object, or the surrounding space).

interactions are possible: scattering at a *surface*, and scattering in a *volume* that fills the region between surfaces. An example of the first case would be the refraction of a photon by a boundary between dielectrics, such as glass and air, whereas the second case models scattering in the interior of a turbid substance like fog or milk (Figure 2.1). We now briefly review the concept of radiance, which is required to precisely characterize these interactions.

2.2.1 Radiance

In geometric optics, the main quantity of interest is *radiance*. Given a point $\mathbf{x}$ and a direction ω, the radiance $L(\mathbf{x}, \omega)$ describes how much illumination flows through the point, in this direction. Intuitively, radiance can be measured approximately by registering the amount of energy (i.e. the rate of photons) arriving on a small surface patch $\mathrm{d}A$ at $\mathbf{x}$ that is perpendicular to ω and sensitive to a small cone of directions $\mathrm{d}\omega$ around ω. In a hypothetical apparatus that implements this measurement (Figure 2.2), we must also divide the resulting number by the exposure time to account for the length of the measurement, as

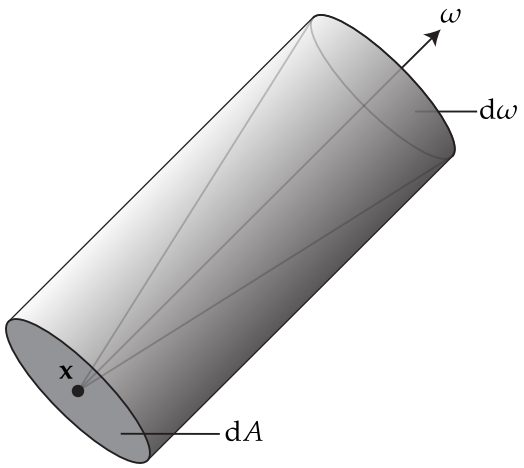


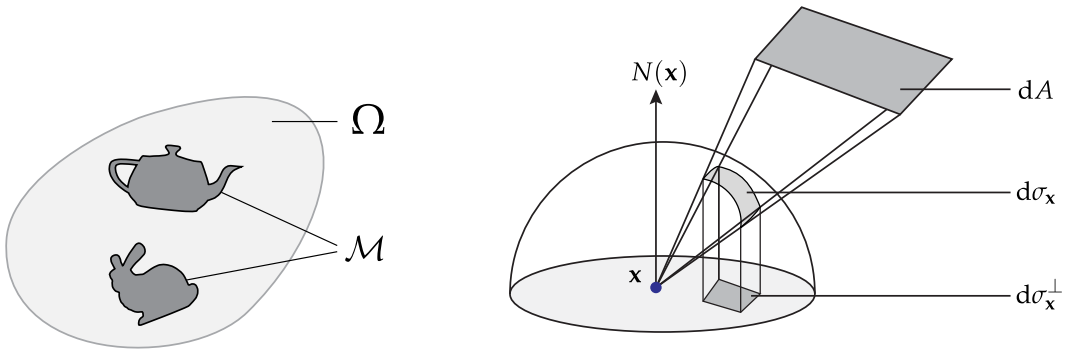
Figure 2.2: Radiance expresses the amount of energy flowing through a point **x**, in a direction ω. It can be measured by determining the power received by surface $\mathrm{d}A$ sensitive to a cone of directions $\mathrm{d}\omega$, where $\mathrm{d}A, \mathrm{d}\omega \to 0$.

well as the surface area $\mathrm{d}A$ and solid angle $\mathrm{d}\omega$ to account for the size of the sensor. In the limit of temporal steady state and small $\mathrm{d}A$ and $\mathrm{d}\omega$, we then obtain radiance, usually expressed in units of $\mathrm{W} \cdot \mathrm{sr}^{-1} \cdot \mathrm{m}^{-2}$. For a thorough review of radiance and many related radiometric quantities, we refer the reader to Preisendorfer [51] and Veach [67].

When light travels unobstructed (i.e. through vacuum), radiance remains invariant along rays:

$$L(\mathbf{x}, \omega) = L(\mathbf{x} + t\omega, \omega), \quad t > 0.$$

This is an important property, since it means that a complete mathematical description of a virtual environment can be obtained simply by specifying how L behaves in places where an obstruction interacts with the illumination. In our case, this can either be a surface or a volume; Sections 2.2.3 and 2.2.4 will provide such a specification for each type in the form of an energy balance equation. Before defining these equations, the next section will review necessary spaces and integration measures.



(a) The scene is described as union of manifolds inside a finite domain.

(b) The relation of the area, solid angle, and projected solid angle measures.

Figure 2.3: An overview of commonly used domains and measures

2.2.2 Commonly used domains and measures

We assume that the scene to be rendered is constructed from a set of surfaces that all lie inside a finite domain $\Omega \subseteq \mathbb{R}^3$. The union of these surfaces shall be denoted as $\mathcal{M} \subset \Omega$ and must be a well-defined smooth manifold[1]. We will often integrate functions over this manifold, using the notation

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

where $\mathrm{d}A(\mathbf{x})$ is the *area measure* and f is an integrable function on $\mathcal{M}$.

The set of directions is another important space that will be used many times. We represent directions using normalized vectors on the unit 2-sphere S^2 and from now on use the following notation to express integration with respect to the *solid angle measure*:

$$\int_{S^2} f(\omega) \, \mathrm{d}\sigma(\omega) \coloneqq \int_0^{2\pi} \int_0^{\pi} f(\theta, \phi) \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi,$$

where f is an integrable function on the sphere expressed in terms of polar coordinates or unit vectors on $\mathbb{R}^3$.

[1]This smoothness assumption is often violated in practice, e.g. at edges between triangles in a triangle mesh. This is permitted, as long as the non-smooth regions have Lebesgue measure zero; in other words, the results of this dissertation extend to discretized meshes, but not surfaces with fractal-like non-smooth structure. For simplicity, derivations assume C^∞ smoothness.

The *projected solid angle measure* finds use when integrating a physical quantity that arrives or leaves a point on a surface. It introduces an extra cosine factor in the definition:

$$\int_{S^2} f(\omega)\,\mathrm{d}\sigma^{\perp}(\omega) \coloneqq \int_0^{2\pi}\int_0^{\pi} f(\theta,\phi)\,|\cos\theta|\,\sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi.$$

The reason for this factor is best explained using an example: consider a thin beam of light that illuminates a surface from a perpendicular direction. When the beam is rotated so that it hits the surface at a grazing angle, it will spread out over much larger area, contributing less energy to each individual surface position within the beam. The cosine term captures this angle-dependent foreshortening effect. Integrating radiance incident on a surface in this way yields *irradiance*, the incident power per unit area.

When integrating a quantity that arrives at or leaves a specific point $\mathbf{x}$, we will emphasize this by writing

$$\int_{S^2} f(\omega)\,\mathrm{d}\sigma_{\mathbf{x}}(\omega) \quad \text{and} \quad \int_{S^2} f(\omega)\,\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega).$$

In the second case, it means that the cosine term is equal to the cosine of the angle between ω and the surface normal $N(\mathbf{x})$ (where $\mathbf{x} \in \mathcal{M}$). These two measures then are related by

$$\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) = \mathrm{d}\sigma_{\mathbf{x}}(\omega)\,|\omega \cdot N(\mathbf{x})|\,.$$

The projected solid angle measure can also be interpreted as a perpendicular projection of a spherical element onto a unit disc perpendicular to the normal direction, known as the *Nusselt analog*. Figure 2.3 (b) shows how all three measures $\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega)$, $\mathrm{d}\sigma_{\mathbf{x}}(\omega)$ and $\mathrm{d}A$ can be related in this way.

Finally, sometimes we will integrate functions that are defined on *ray-space* $\mathcal{M} \times S^2$. This is nothing other than the Cartesian product of surface positions

and directions, with the corresponding product measure, i.e.

$$\int_{\mathcal{M}} \int_{S^2} f(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A.$$

2.2.3 Energy balance equation for volumes

The behavior of light passing through an optically interacting material is governed by absorption and scattering. In turbid substances like fog or milk, these occur due to collisions between photons and the optically active contents of the material, such as water droplets or fat globules. Unfortunately, the sheer number of these small suspended particles makes it impractical to account for their effects individually.

Luckily, much like the analysis of gases in statistical physics, large populations of scattering particles can also be studied effectively using the tools of probability theory and statistics. The main insight of this approach is that the exact position of all the individual water droplets in a volume filled with humid air can be neglected during computation, as long as their average effects on light can somehow be modeled.

Linear transport theory provides a convenient mathematical framework that does precisely this by describing the propagation of light through a *random medium*, a statistical interpretation of the aggregate scattering behavior of many small particles that fill a region of space. This theory requires us to assign a particle density function to every point in space. A photon traveling along a straight line trajectory can then be absorbed or scattered (i.e. redirected) by particles that spontaneously and randomly "manifest" in front of the photon proportionally to this particle density, hence turning the trajectory into a random walk. The interactions are made precise by the *radiative transfer equation* (RTE), which we will now briefly describe.

Figure 2.4: Rendering of a glass of milk, whose contents were simulated using the Radiative Transfer Equation described in this section.

Roughly speaking, the RTE expresses all the different ways in which the host medium can deviate from conservation of radiance[2] along a ray $(\mathbf{x}, \omega)$. It does this by equating the directional derivative of radiance $L(\mathbf{x}, \omega)$ in direction ω to a sum of terms that each map to an intuitive physical explanation (Figure 2.5). On the domain $\Omega \subseteq \mathbb{R}^3$, the steady-state form of the RTE is given by

$$(\omega \cdot \nabla)\ L(\mathbf{x}, \omega) + \sigma_t(\mathbf{x})\ L(\mathbf{x}, \omega) = \\ \sigma_s(\mathbf{x}) \int_{S^2} f_p(\mathbf{x}, \omega' \to \omega)\ L(\mathbf{x}, \omega')\, \mathrm{d}\sigma_{\mathbf{x}}(\omega') + L_e(\mathbf{x}, \omega), \qquad \mathbf{x} \in \Omega^\circ, \quad (2.1)$$

where σ_s and σ_t are the *scattering* and *extinction coefficients*, f_p is the *phase function*, and L_e represents emitted radiance. We will later specify boundary conditions

[2] In vacuum, radiance is fully conserved, i.e. $(\omega \cdot \nabla)\ L(\mathbf{x}, \omega) = 0$

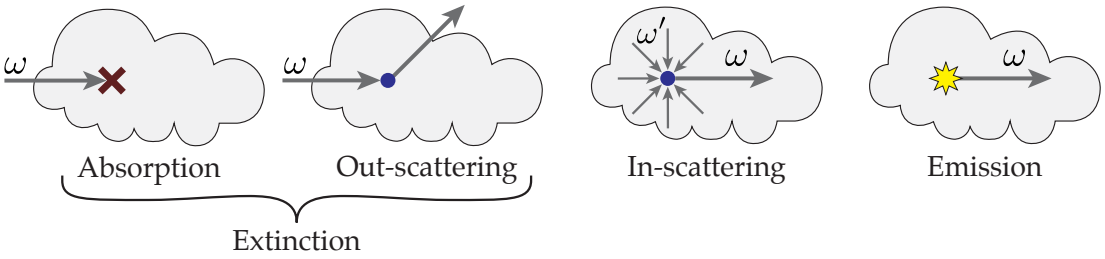


Figure 2.5: The different terms of the radiative transfer equation each have an intuitive physical interpretation

on the set of surfaces $\mathcal{M} \subseteq \Omega$, hence the above equation only holds on the interior of the domain, i.e. $\Omega^\circ := \Omega \setminus \mathcal{M}$.

In (2.1), the first term is the aforementioned directional derivative. The second *extinction* term accounts for light that collides with the particles that fill the volume. Such a collision causes it to be either absorbed or redirected into another direction and thus removed from the ray.

The particles also scatter a portion of the illumination that is traveling along *different* rays passing through the point $\mathbf{x}$, and some of it will be redirected into direction ω. This is modeled by the third term, which convolves the radiance and phase[3] functions over the sphere to obtain all light that is locally added to the ray $(\mathbf{x}, \omega)$. Given light arriving from direction ω', the phase function expresses the probability density of it being scattered into direction ω. The precise form of this function depends on the shape and optical properties of the particles that make up the volume. The measure $\mathrm{d}\sigma_{\mathbf{x}}$ indicates that the integration is done with respect to solid angles at $\mathbf{x}$.

When the medium itself emits illumination (e.g. a flame), the last term involving L_e describes this effect. Note that in contrast to L, the function L_e has units of radiance *per unit length* at interior points $x \in \Omega^\circ$.

[3]The name "phase function" is historically due to the phases (i.e. the varying brightness) of celestial bodies in astronomy.

Note the subtle difference between the term *volume*, which refers to a region of space, and *medium*, which describes its contents. In practice, these are often used interchangeably.

In the specification of a scene, the main objects of interest then are:

(i) the extinction cross-section σ_t, which is sometimes further decomposed into the product of a particle density (units of m^{-3}) and the particle surface area (units of m^2). By itself, σ_t can be interpreted as the inverse expected distance between interactions with particles (units of m^{-1}). The higher this number, the more "optically dense" the material will be.

(ii) the scattering albedo σ_s/σ_t, i.e. the relative probability of a non-absorbing scattering event taking place following the collision of a photon with a particle. (unitless).

(iii) the phase function $f_p(\omega_i \to \omega_o)$, a conditional probability distribution over scattered directions ω_o for a given incident direction ω_i (units of $1/\text{sr}$).

An enormous body of work on linear transport and the RTE exists; we refer the reader to [2, 24] for a detailed treatment.

Recent efforts have focused on the benefits of non-classical radiative transport, where additional statistical correlations are introduced into the RTE. For instance, in certain materials, scattering events become more (or less) likely once a photon has traveled a certain distance from the last event. This is related to the size and spatial arrangement of particles in the medium. Larsen and Vasques [37] introduce an additional parameter that captures this effect.

Jakob et al. [26] propose an anisotropic radiative transfer framework derived from scattering by oriented non-spherical particles. This enabled them to render media made of fibers and other structured materials that break an assumption of rotational invariance that is part of the standard derivation of the RTE.

Both of these extensions lead to non-classical RTEs which, although more involved, are structurally very similar to the classical equation. For simplicity, we use the classical equation in all derivations in this thesis, but point out that a non-classical version could be substituted if needed.

2.2.4 Energy balance equation for surfaces

Let's now consider the surface case: given a set of surfaces $\mathcal{M}$, we must specify their response to incident illumination, which can be thought of as a boundary condition for the RTE (2.1).

A potential issue is that by specifying such boundary conditions, we invariably also introduce discontinuities in the radiance function L. These discontinuities are problematic when referring to the surface radiance field, since it will generally not be identical on both sides.

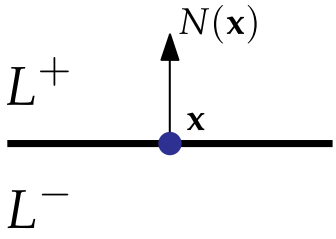


Figure 2.6: Limits of the radiance function L from above and below

Analogous to a one-sided limit in the 1D case, we therefore distinguish between the front and back components $L^+(\mathbf{x}, \omega)$ and $L^-(\mathbf{x}, \omega)$ as determined by the surface normal $N(\mathbf{x})$ for $\mathbf{x} \in \mathcal{M}$ (Figure 2.6). Based on this separation, the more intuitive incident and exitant radiance functions can then be defined as

$$L_i(\mathbf{x}, \omega) := \begin{cases} L^+(\mathbf{x}, -\omega), & \omega \cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x}, -\omega), & \omega \cdot N(\mathbf{x}) < 0 \end{cases} \quad \text{and}$$

$$L_o(\mathbf{x}, \omega) := \begin{cases} L^+(\mathbf{x}, \omega), & \omega \cdot N(\mathbf{x}) > 0 \\ L^-(\mathbf{x}, \omega), & \omega \cdot N(\mathbf{x}) < 0 \end{cases}$$

Away from surfaces (i.e. in free space), L is continuous, so $L^+ = L^-$, which means $L_o(\omega) = L_i(-\omega) = L(\omega)$. In other words, L_i and L_o only differ by a

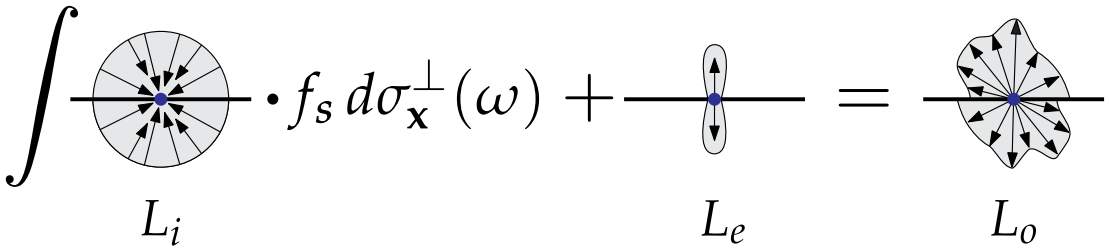


Figure 2.7: At surfaces, incident and outgoing radiance are related by an integral transformation involving the bidirectional scattering distribution function (BSDF) and the addition of emission (if any).

direction reversal. With the help of these definitions, we can introduce the surface energy balance equation

$$L_o(\mathbf{x}, \omega) = \int_{S^2} f_s(\mathbf{x}, \omega' \to \omega) L_i(\mathbf{x}, \omega') \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega') + L_e(\mathbf{x}, \omega), \quad \mathbf{x} \in \mathcal{M}. \tag{2.2}$$

which equates the outgoing radiance to a weighted integral over the incident illumination plus the emitted radiance, which has different units here than in (2.1) as will be explained shortly. The function f_s is the *bidirectional scattering distribution function* (BSDF) of the surface, which characterizes the surface's appearance when subjected to illumination (Figure 2.7). Given illumination reaching a point $\mathbf{x}$ from a direction ω', the BSDF intuitively captures how much of this illumination is scattered into the direction ω. A detailed definition and classification with respect to other types of scattering functions is given by Nicodemus [46].

Specular materials are characterized by having a "degenerate" BSDF f_s that is described by a Dirac delta distribution. For instance, a mirror reflects light arriving from ω into only a single direction $\omega' = 2N(\mathbf{x})\langle \omega, N(\mathbf{x}) \rangle - \omega$.

In comparison, rough materials usually have a smooth f_s. BSDFs based on *microfacet theory* [64, 8, 74] are a popular choice in particular—they model the interaction of light with random surfaces composed of microscopic dielectric or conducting facets that are oriented according to a *microfacet distribution*. Integra-

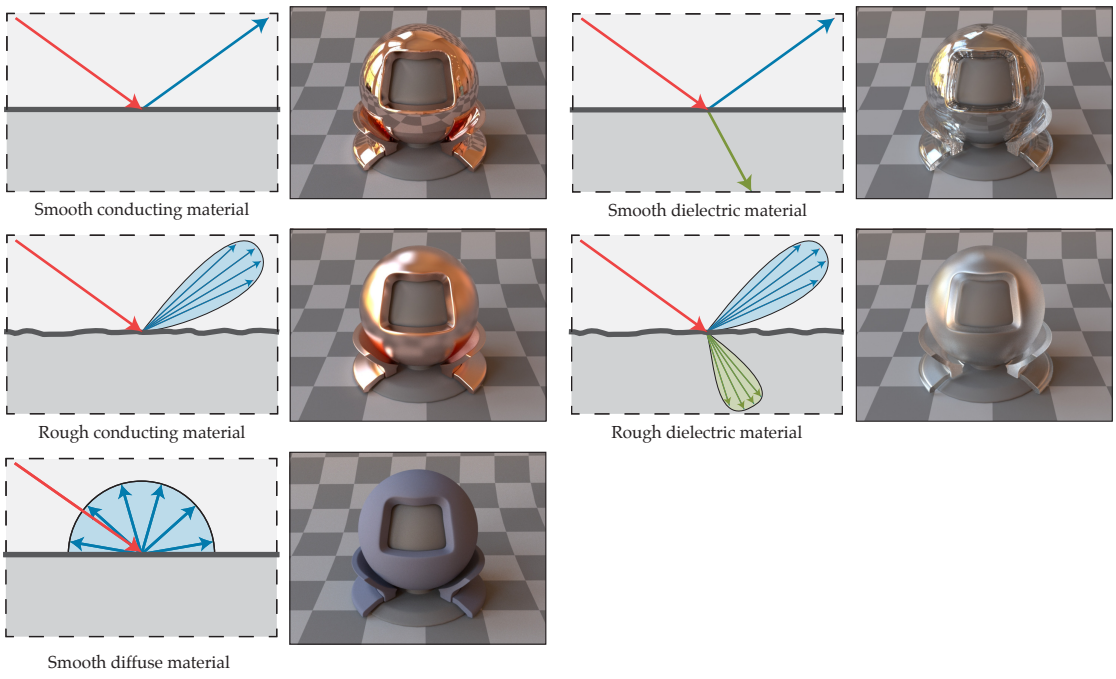


Figure 2.8: A schematic overview of the main classes of BSDFs used in this thesis, illustrated by renderings of a material test object.

tion over this distribution then leads to simple analytic expressions that describe the expected reflection and transmission properties at a macroscopic scale. Validations against real-world measurements have shown that microfacet models compare favorably against other families of parametric BRDF models [44]. To render rough dielectrics and conductors, our simulations make extensive use of the model by Walter et al. [74].

Figure 2.8 shows an overview of different classes of BSDFs used in this thesis, along with the resulting material appearance. The polar plots to the left of the renderings show the scattered radiance $f_s(\omega' \to \omega)$ when the surface receives illumination from a fixed incident direction ω' highlighted in red. The primary set of reflected directions is shown in blue, and the transmitted directions (if any) are shown in green. The color of the materials (e.g. the reddish hue in the case of copper) is not relevant for this classification.

As before, the function L_e is used to model emission. Recall that Section 2.2.3

introduced this function for interior points (i.e. away from surfaces), where it had units of radiance per unit length. We now extend this definition to surface points $\mathbf{x} \in \mathcal{M}$ (e.g. black body radiation from a hot surface), where L_e takes on units of radiance. Note the different measure when compared to (2.1); $\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega')$ indicates integration with respect to projected solid angles.

2.3 Light source and camera models

To characterize the light sources that are part of the simulated environment, the emission function L_e must be defined. Frequently used models here range from simple point lights that uniformly emit in all directions to measured light sources (e.g. distributed in the popular IES lighting exchange format) and sophisticated models for flames [45] or the sky [50, 21]. Another common approach is to make the entire light source part of the simulation by modeling a simple filament surrounded by a (potentially complex) glass enclosure and mirror elements.

As mentioned earlier, a rendering algorithm creates a rendering essentially by simulating the process of taking a photograph in a virtual setting. In addition to a full specification of all objects and light sources in the simulated environment, it therefore also requires knowledge about the camera. Such a camera generally captures the illumination on a regular pixel grid so that we can think of each pixel as a separate sensor, whose response may vary with respect to both position and direction of the incident illumination. Assuming a linear sensor, we can define a function $W_e(\mathbf{x}, \omega)$ that models the ratio of sensor response per unit of power arriving along a ray $(\mathbf{x}, \omega)$. The point $\mathbf{x}$ is taken to lie on the aperture of the sensor, which is part of the set of surfaces $\mathcal{M}$. The function W_e is referred to as the *importance* and is general enough to represent any kind of

linear sensor, including perspective, orthographic, or fisheye cameras with any kind of aperture.

The precise form of the importance function depends on the type of camera model and is usually a simple analytic expression. As with BSDFs, W_e can become "degenerate" when the response curve is limited to a discrete set of points or directions. This is the case e.g. for pinhole cameras and orthographic sensors, where the positional and directional dependencies reduce to Dirac delta functions, respectively.

For a pixel j, the camera performs an associated "measurement" I_j by computing a ray-space integral (Section 2.2.2) over the importance function $W_e^{(j)}(\mathbf{x}, \omega)$ associated with camera pixel j and the incident radiance L_i:

$$I_j = \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x})$$

The full set of virtual measurements I_j then constitutes the output image. This integral can be used to define an inner product so that the above can be written more compactly:

$$= \langle W_e^{(j)}, L_i \rangle. \tag{2.3}$$

This expression is also known as the *measurement integral*.

Reciprocity

Physical light transport satisfies a fundamental reciprocity property, which states that a sensor and an emitter can be exchanged without influencing the value of a hypothetical measurement performed between them (see e.g. Dual Photography [55] for an interesting real-world application of this property). In light of this reciprocity, the subscript e in W_e highlights the symmetry between emitters and sensors, which allows one to think of importance as an emitted

quantity analogous to radiance. This is the basis for algorithms like bidirectional path tracing, which simulate two simultaneous scattering processes involving radiance and importance transport. We will not go into detail about reciprocity and refer the reader to Veach [67], who provides an in-depth discussion of its consequences.

2.4 Solution techniques

Simulating light transport has been a major effort in computer graphics for over 25 years, beginning with the complementary approaches of finite-element simulation, or radiosity [12], and ray-tracing [76]. In this section, we will briefly review commonly used solution techniques, focusing on surface light transport for simplicity.

Recall that to create a rendering, we must find the value of the measurement integral (2.3) for each pixel j:

$$I_j = \langle W_e^{(j)}, L_i \rangle.$$

$W_e^{(j)}$ is usually a simple analytic function, hence the main challenge in computing this integral is the evaluation of L_i. At this point, it is not clear how to do this at all, since we lack an explicit functional representation of L_i.

It will be convenient to establish some further notation: recall that $\mathcal{M}$ denotes the set of surfaces. We can define the distance to the next surface encountered by the ray $(\mathbf{x}, \omega) \in \mathbb{R}^3 \times S^2$ as

$$d_{\mathcal{M}}(\mathbf{x}, \omega) := \inf \{d > 0 \mid \mathbf{x} + d\omega \in \mathcal{M}\}$$

where $\inf \emptyset = \infty$. Based on this distance, we define a *ray-casting function* $\mathbf{x}_{\mathcal{M}}$:

$$\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega) = \mathbf{x} + d_{\mathcal{M}}(\mathbf{x}, \omega)\omega. \tag{2.4}$$

A very useful application of the ray-casting function involves relating the quantities L_i and L_o based on the preservation of radiance along rays[4]—specifically,

$$L_i(\mathbf{x}, \omega) = L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega), -\omega)$$

In other words, to find the incident radiance along a ray $(\mathbf{x}, \omega)$, we must only determine the nearest surface visible in this direction and evaluate its outgoing radiance into the opposite direction. Using this, we can rewrite the measurement integral (2.3) as

$$I_j = \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega), -\omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}) \tag{2.5}$$

Furthermore, we can also eliminate L_i from the energy balance equation (2.2):

$$L_o(\mathbf{x}, \omega) = \int_{S^2} f_s(\mathbf{x}, \omega' \to \omega) L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega'), -\omega') \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega') + L_e(\mathbf{x}, \omega) \tag{2.6}$$

Together with a specification of the visible surfaces $\mathcal{M}$ and the BSDF f_s for each $\mathbf{x} \in \mathcal{M}$, the above set of equations then constitutes a well-defined integral equation problem which can be solved for L_o. Although the answer is not given explicitly, the equations are in a form that is suitable for standard solution techniques.

Due to the complexity of these integrals, we must unfortunately abandon all hope of simple analytic solutions and turn to numerical integration. However, this is made difficult by the ill-behaved nature of the integrands, which are usually riddled with discontinuities caused by changes in visibility in the ray-casting function $\mathbf{x}_{\mathcal{M}}$. Practical solution methods often rely on a Neumann series expansion of the underlying integral operators; the resulting high number of dimensions rules out standard deterministic integration rules, whose number

[4]Note that for simplicity, this discussion is restricted to pure surface scattering; the addition of volume scattering breaks this property. In Section 3, we develop a path-space integration framework that does not suffer from this limitation.

of function evaluations grows exponentially with dimension. Monte Carlo methods are resilient to both of these issues and hence see significant use in rendering. We will discuss this approach, focusing on the example of a simple path tracer.

2.4.1 Overview of Monte Carlo methods

The introduction of Monte Carlo methods for ray tracing [7], followed by Kajiya's formulation of global illumination in terms of the Rendering Equation [31], established the field of Monte Carlo global illumination. Unbiased sampling methods, in which each pixel in the image is a random variable with an expected value exactly equal to the solution of the Rendering Equation, started with Kajiya's original path tracing method and continued with bidirectional path tracing proposed by Veach et al. [68] and Lafortune et al. [35], in which light transport paths can be constructed partly from the light and partly from the eye, and the Metropolis Light Transport [70] algorithm, which uses bidirectional path tracing methods in a Markov Chain Monte Carlo framework.

We shall briefly review Monte Carlo integration using the evaluation of the integral (2.5) as an example application. Suppose $p_X(x)$ is the probability density of an as of yet unspecified random variable X defined with respect to the measure $\mathrm{d}\sigma^{\perp}_{\mathbf{x}}\,\mathrm{d}A$ on ray-space $S^2 \times \mathcal{M}$. In the continuous case, the expected value of a function g of X then equals

$$E_X\left[g(X)\right] = \int_{\mathcal{M}} \int_{S^2} g(\mathbf{x},\omega)\, p_X(\mathbf{x},\omega)\,\mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega)\,\mathrm{d}A(\mathbf{x})$$

Given this, we can define g in the following specific way

$$g(\mathbf{x},\omega) := \frac{W_e^{(j)}(\mathbf{x},\omega) L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega), -\omega)}{p_X(\mathbf{x},\omega)}, \tag{2.7}$$

with the result that[5]

$$I_j = E_X\left[g(X)\right]. \tag{2.8}$$

In other words, the integral whose value we desire to know now arises as the expectation of the random variable $g(X)$. Given a sequence $X_1, X_2, \ldots$ of independent realizations of X (i.e. random variates distributed with respect to the probability distribution p_X), we can define a statistical estimator $\langle I_j \rangle$ by taking averages of the $g(X_i)$:

$$\langle I_j \rangle \coloneqq \frac{1}{N} \sum_{i=1}^{N} g(X_i).$$

The law of large numbers then guarantees that $\langle I_j \rangle \to I_j$ as $N \to \infty$. This simple idea is the foundation of Monte Carlo integration: by randomly sampling points in the domain and evaluating a suitably chosen function, we obtain a sequence of numbers whose arithmetic mean converges to the correct answer.

For practical computation, it is also important to consider the variance of this estimator: clearly, the lower the variance (and thus, the standard deviation), the more accurate estimates of the value of the integral we will obtain. For this estimator, it is given by

$$\operatorname{Var}\left[\langle I_j \rangle\right] = \frac{1}{N} \int_{\mathcal{M}} \int_{S^2} \left(g(\mathbf{x}, \omega) - I_j\right)^2 p_X(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}) \tag{2.9}$$

Inspecting this immediately reveals the main issue with Monte Carlo integration: variance decreases linearly as $N \to \infty$, hence the standard deviation only falls off proportionally to $\sqrt{N}$, a comparatively poor convergence speed. In practice, this means that to reduce integration errors by a factor of two, we must increase the number of samples four-fold.

Furthermore, the variance of $\langle I_j \rangle$ is highly related to the choice of density function p_X that is used to pick sample points on the domain. To see this, we

[5] This assumes that g is well-defined on all of $\mathcal{M} \times S^2$, i.e. $p_X = 0$ iff the numerator vanishes.

can use the identity $\mathrm{Var}[X] = E[X^2] - E[X]^2$, to rewrite (2.9) as

$$\mathrm{Var}\left[\langle I_j \rangle\right] = \frac{1}{N}\left[\int_{\mathcal{M}}\int_{S^2} \frac{\left(W_e^{(j)}(\mathbf{x},\omega) L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega), -\omega)\right)^2}{p_X(\mathbf{x},\omega)} \,\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega)\,\mathrm{d}A(\mathbf{x}) - I_j^2\right].$$

Suppose that we could choose p_X to be proportional to the product of importance and outgoing radiance in the numerator of the above fraction. Due to its definition (2.7), g then turns into the factor of proportionality, which must equal I_j as a consequence of Equation (2.8), and the variance simplifies to

$$= \frac{1}{N}\left[\underbrace{I_j \int_{\mathcal{M}}\int_{S^2} W_e^{(j)}(\mathbf{x},\omega) L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega), -\omega)\,\mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega)\,\mathrm{d}A(\mathbf{x})}_{= I_j} - I_j^2\right],$$

in this case, we thus find that $\mathrm{Var}[\langle I_j \rangle] = 0$! But using such a p_X requires already knowing the answer to the problem, as the factor of proportionality is the value we originally set out to compute.

When it is difficult to generate samples proportionally to a function that resists analytic treatment, a common trade-off entails focusing on a simpler term that can be separated out (e.g. a factor in a product expression). In our current example, the integrand is the product of $W_e^{(j)}$, a (usually) simple analytic expression for the importance, i.e. the sensitivity profile of the camera, and L_o, an as-of-yet unknown and potentially very involved function describing the outgoing radiance of surfaces in the scene. Hence, in this case, we could set p_X proportional to $W_e(j)$ and devise a sampling procedure that can generate appropriately distributed random samples. Since a pixel in a camera generally only responds to radiance arriving from a small set of positions and directions, this will be vastly superior compared to uniform sampling on the entire domain.

A critical point that we have neglected until now is that the described method assumes the ability to evaluate the outgoing radiance function L_o, but this is an unknown quantity itself! Thus far, we only know that it satisfies the integral

equation (2.6). Fortunately, there is a simple way out of this dilemma: it can be shown that Monte Carlo integration of a product expression still works if we replace the evaluation of the individual factors with statistically independent estimators, as long as they have the right expected value. Hence, whenever L_o needs to be evaluated, can we simply perform a recursive Monte Carlo integration using an estimator $\langle L_o \rangle$ created analogously to $\langle I_j \rangle$.

This brings up another problem: since L_o is defined as the value of an integral that itself contains L_o, this approach leads to an infinite recursion that must eventually be stopped to yield a method that can finish in a finite amount of time. A simple way to accomplish this is by means of an auxiliary estimator

$$\langle L_o \rangle_{\mathrm{rr}} = \begin{cases} a^{-1} \langle L_o \rangle, & \text{with probability } a \\ 0, & \text{with probability } 1 - a \end{cases}$$

where $0 < a < 1$. In computer graphics, this technique is known as *russian roulette*. In an implementation, this is realized by drawing a uniform variate from a pseudorandom number source whenever a recursive Monte Carlo integration is to be performed. When the resulting number is below an appropriately chosen threshold a, the recursive integration is performed, and its result scaled by a^{-1}. Otherwise, the procedure simply substitutes zero for the nested integral's value and terminates the recursion. The estimator obtained in this way has the same expected value and terminates eventually with probability one.

2.4.2 Path tracing

In practice, one more improvement is necessary to yield a viable method: the outgoing radiance function L_o is separated into two terms corresponding to light that has interacted with at most one surface since its emission from a light

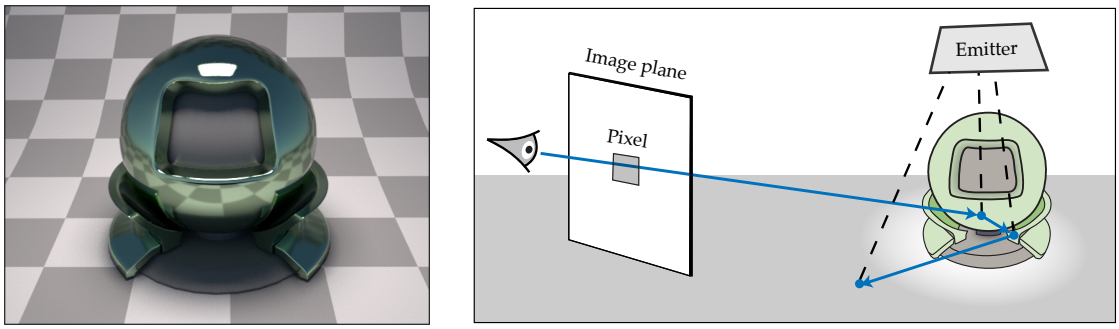


Figure 2.9: The incremental creation of light paths in a path tracer through recursive sampling of directions in the domain of Equation (2.6).

source (*direct illumination*), and light that has undergone multiple scattering events already (*indirect illumination*).

$$L_o = L_o^{(\text{direct})} + L_o^{(\text{indirect})}$$

The recursive Monte Carlo integration approach discussed earlier is then used for the second term, while considerably more efficient specialized methods [56] are used for the direct illumination term.

Implementing the described integration scheme gives rise to a method known as *path tracing* [31]. Its ability to produce photorealistic renderings, while being simple to understand and implement, has made it an extremely popular approach. Figure 2.9 visualizes the operation of a path tracer. By recursively sampling points in the domain of the underlying integral equations, a path tracer effectively builds a light path via a *random walk* starting at the camera. A single random walk is shown in the figure, which entails casting a ray associated with a pixel in the output image and searching for the first visible intersection. A new direction is then chosen at the intersection, and the ray-casting step repeats over and over again, until the russian roulette stopping criterion is triggered.

At every surface intersection, the path tracer tries to create a connection to the light source to find a complete path, along which light can flow from the

light source to the camera. The fact that this can only succeed when there is no occluding object between the intersection and the light source immediately suggests the range of scenes where a path tracer can be expected to operate efficiently: this is the case when the emitters are easily "accessible" to the contents of the scene.

For instance, a light transport simulation of an exterior scene lit by an overcast sky causes no problems, while an interior scene that is lit through a slightly ajar door will likely produce a very noisy rendering. Here, the low probability of a successful connection between objects in the scene and the light source in the adjacent room causes the associated statistical estimator to have a high variance, which manifests as noise in the output image. While the noise will be reduced with increased number of samples, the slow $1/\sqrt{N}$ convergence may cause this to be impractically slow.

Perhaps somewhat surprisingly, another feature of scenes that frequently causes problems with path tracing-like algorithms is the presence of specular materials. Although they transmit and reflect light, materials like mirrors or dielectric boundaries effectively act like occluders during path sampling. As a consequence, taking a scene and embedding its light sources in glass enclosures tends to create a significantly harder rendering problem. This is unfortunate, since such precise modeling is often highly beneficial for improved realism.

Let us briefly reinterpret the repeated recursive integrations of path tracing as a method for sampling a *single* integral over a higher-dimensional domain[6]. Seen from a high level, such difficult-to-reach light sources and specular materials create small regions in this domain—that is, small sets of paths—where the integrand takes on high values; yet, their small size and the random sampling-based approach of path tracing mean that only few samples are likely to be

[6]This notion will be formalized in Chapter 3.

(a) Path tracer, 32 samples/pixel **(b)** Bidirectional path tracer, 32 samples/pixel

Figure 2.10: A bidirectional path tracer finds light paths by generating partial paths starting at the camera and light sources and connecting them in every possible way. The resulting statistical estimators tend to have lower variance than unidirectional techniques. Modeled after a scene by Eric Veach.

placed there. To work around these issues, a number of more sophisticated rendering techniques have been developed in the past.

2.4.3 Bidirectional path tracing

A bidirectional path tracer (BDPT) [68, 35] computes radiance estimates by starting two separate random walks from the light sources and the camera. The resulting *subpaths* are connected at every possible interaction vertex, creating many complete paths of different lengths. This significantly more general type of sampling procedure cannot be expressed in a recursive Monte Carlo sampling framework like the one described in Section 2.4.1; to give the algorithm a sound theoretical footing, Veach [67] introduced *path-space integration* to rendering. The techniques proposed in this dissertation also rely on path space—we defer a full discussion until Chapter 3, which contains a detailed derivation that generalizes the framework of Veach with support for both surface and volume scattering.

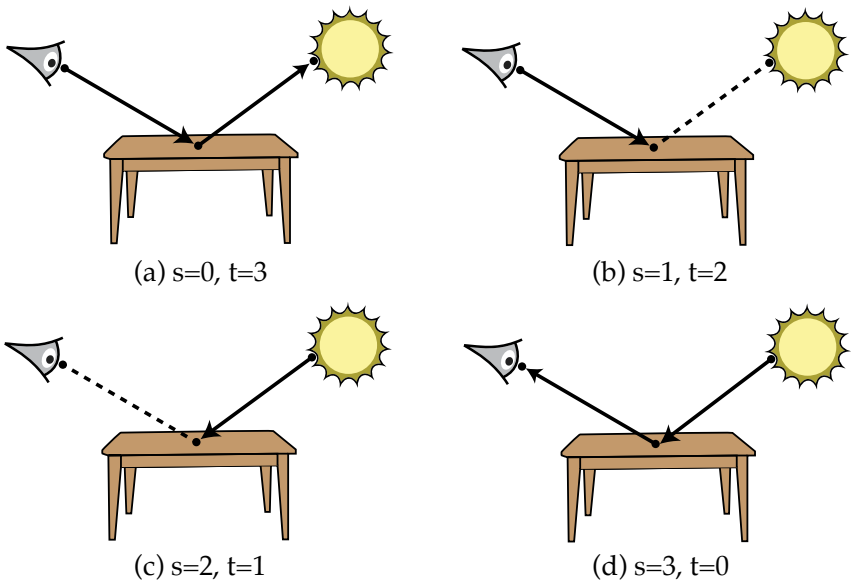


Figure 2.11: The four different ways in which bidirectional path tracing can create a direct illumination path (matching the first row in Figure 2.12): **(a)** Standard path tracing without direct illumination sampling, **(b)** path tracing with direct illumination sampling, **(c)** particle tracing with recording of scattering events observed by the camera, **(d)** particle tracing with recording of particles that hit the camera.

The key idea of path space is a transformation that expresses the measurement integral (2.3) as an integral over paths. With this re-formulation, instead of sampling rays and directions, the problem now becomes selecting *paths* at random from the set of all possible paths. This setting provides a fertile ground for the development of a wide range of different path sampling strategies.

BDPT's approach of exhaustively connecting two subpaths supplies it with an entire family of *connection strategies* so that a path containing n scattering events can now be created in $n + 3$ different ways. Figure 2.11 illustrates how the same direct illumination path (i.e. $n = 1$) can be produced in four different ways. Here, s and t indicate the number of sampling steps from the camera and light source, respectively[7].

[7]Sampling a position on the camera aperture or light source is also counted as a sampling step, hence the numbers may seem larger than they should be.

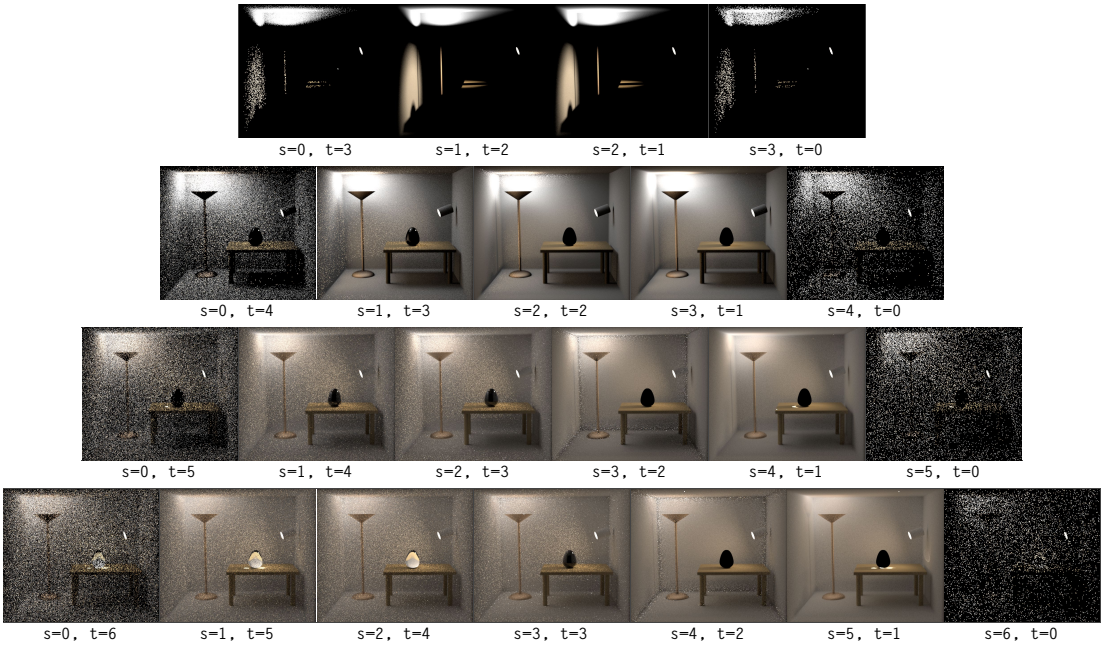


Figure 2.12: The individual sampling strategies that comprise the BDPT rendering shown earlier, but *without* multiple importance sampling. Each row corresponds to light paths of a certain length. Note how almost every sampling strategy has deficiencies of some kind.

Figure 2.13: The same sampling strategies, but now weighted using multiple importance sampling—effectively "turning off" each strategy where it does not perform well. The final result is computed by summing all of these images.

One way to think of about these different sampling strategies is that each one effectively acts like a change of variables: for instance, the four direct illumination strategies correspond to four different reparameterizations of the same integral with the goal of making it more amenable to numerical integration. In practice, each of the strategies is usually successful at dealing with certain types of light paths, while being an exceptionally poor choice for others.

A key insight by Veach [69] is that it is possible to create linear combinations of the strategies in a way that, roughly speaking, locally re-weights them based on their predicted utility. Using this approach, known as *multiple importance sampling* (MIS), BDPT is able to rely on each strategy in regions of the domain where it is good, discarding it elsewhere (Figures 2.12 and 2.13).

Yet, BDPT cannot overcome all problems of path tracing. This becomes apparent when rendering difficult scenes that, for instance, contain the class of specular-diffuse-specular paths mentioned in Chapter 1. In such situations, all of the constituent sampling strategies tend to perform poorly, and hence it is not possible to create a superior strategy by means of re-weighting. Nonetheless, bidirectional path tracing is often a valuable improvement over standard path tracing; we use it as a generator of light paths to seed the rendering method proposed in this dissertation.

Similar to BDPT, our new method also makes use of many different strategies for finding light paths bidirectionally. As we shall see later, this ability arises quite naturally within the framework of Metropolis Light Transport, removing the need for multiple importance sampling in the main part of the algorithm.

2.4.4 Two-pass methods

Various two-pass methods have been proposed for rendering difficult classes of light paths. They operate by precomputing an approximate representation of the scene illumination, for instance by solving a linear system [12, 58], or by sending out energy from the light sources in the form of photons [57, 28, 73] or virtual point lights [33] that are traced through the scene and stored in a simple list or spatial data structure. A second pass then renders the image using ray tracing, making use of the precomputed data via hierarchical clustering [72, 71], density estimation [29, 27, 14], or other kinds of queries.

Photon mapping and other two-pass methods are characterized by storing an approximate representation of some part of the illumination in the scene, which requires assumptions about the smoothness of the illumination distribution. On one hand, this enables rendering of some modes of transport that are difficult for unbiased methods, since the exact paths by which light travels do not need to be found; separate paths from the eye and light that end at nearby points suffice under assumptions of smoothness. Much like relaxation techniques for discrete search problems, this "relaxes" the original integration problem so that it becomes easier to handle. However, this inherently leads to smoothing errors in images: the results are biased, in the Monte Carlo sense.

Glossy materials also limit the effectiveness of two-pass methods and remain challenging even with the smoothing they introduce. The reason for this can be seen when examining the behavior of these methods with respect to different materials. For a fixed outgoing direction, a diffuse material responds equally to illumination arriving from all directions, hence it suffices to use a radiance representation that only considers positions. Glossy materials, on the other hand, only reflect light arriving from a small cone of directions into the

given outgoing direction. To render such a material effectively, the underlying radiance representation must therefore sample position-direction space, which increases its dimension from 2D to 4D. Some photon mapping variants avoid this by computing radiance on glossy materials using a recursive Monte Carlo integration, but this means that the resulting methods increasingly resemble path tracing as the number of glossy surfaces in the input scene grows.

Many-light algorithms [33, 72, 18] convert the steady-state illumination of a scene into a large set of point light sources (e.g. thousands to millions). Indirect illumination is already part of this representation, hence the main rendering step only has to account for direct illumination light paths. Furthermore, clustering can be used to significantly reduce the number of light sources that must be dealt with. To avoid distracting image artifacts, the contribution of each individual light source is usually limited to a certain maximum amount, which is referred to as clamping.

These steps limit the types of materials that can be rendered well in practice, and the clamping of the lights introduces statistical bias. As is the case with photon mapping, many-light algorithms have difficulty rendering interreflection between glossy materials.

The focus of this dissertation is the solution of the original non-relaxed integration problem, hence we do not consider two-pass methods.

2.4.5 Overview of Markov Chain Monte Carlo

So far, we have discussed several different approaches for constructing light paths—either by building them incrementally from the camera or, bidirectionally, from the light sources and the camera, potentially by means of an intermediate photon data structure. What these methods all have in common is that, in one

form or another, a part of them relies on the generation of random samples. Given a probability distribution defined over the domain of an integrand, they generate *statistically independent* random samples according to this distribution and render the output image by evaluating the integrand at the sample locations. Rendering would be a much simpler problem if it was possible to let this sampling probability be *exactly* proportional to the integrand, but this is prevented by the complexity of the latter so that severe compromises must be made in practice.

Yet, *Markov Chain Monte Carlo* (MCMC) is a method that promises exactly this: to generate samples proportional to an arbitrary probability distribution that may be impractical to sample using any other technique. It is able to accomplish this by giving up the property of independence: samples obtained from a MCMC method are *statistically correlated*, which introduces a different set of tradeoffs that we will review in this section. We begin by looking at MCMC in a general setting; afterwards, we focus on its application to rendering. Our introduction follows Liu [38] and Pharr [48].

A *Markov chain* is a sequence of random variables $\mathbf{X}_1, \mathbf{X}_2, \ldots$ in a state space in which the probability of a state appearing at a given position in the sequence depends only on the previous state. In a *homogeneous* Markov chain, the rule that governs transitions between states is furthermore invariant over time[8] so that we can specify a *transition probability*

$$\Pr\left(\mathbf{X}_i = \mathbf{x}_i \mid \mathbf{X}_{i-1} = \mathbf{x}_{i-1}, \ldots, \mathbf{X}_1 = \mathbf{x}_1\right) = \Pr\left(\mathbf{X}_i = \mathbf{x}_i \mid \mathbf{X}_{i-1} = \mathbf{x}_{i-1}\right) =: P(\mathbf{x}_i, \mathbf{x}_{i-1})$$

that only depends on the realizations $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$, but not on i. The type of coupling between states in a Markov chain ensures that its behavior can be predicted based on the observation of just one of its states—additional history

[8] Here, time refers to the discrete time of a stochastic process rather than physical time.

does not improve such a prediction, a property referred to as *memorylessness*. This term can be slightly misleading, because the correlation between adjacent states does in fact create a form of short-term memory. We will see later that this is one the chief advantages of Metropolis-type methods.

When the state space of a Markov chain is finite, and the chain is able to pass from any state in the domain to another using a finite expected number of steps, and if the length of this journey is in a sense "irregular" (i.e. aperiodic), it is referred to as *ergodic* and *irreducible*. In this case, it can be shown that the distribution of states in the sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots$ converges to a unique *stationary distribution* regardless of the initial state $\mathbf{x}_1$. In the continuous case, the characterization of this convergence is more involved. We will not go into details, as all of the Markov chains considered in this thesis trivially satisfy ergodicity and irreducibility by being able to directly reach any state in the domain starting from any position, using a single transition.

The basic idea of MCMC, first proposed by Metropolis et al. [40], is to define a Markov chain that has a stationary distribution proportional to the function to be integrated, meaning that if the chain is run for a long time, the distribution of states it visits will be proportional to the desired distribution. Surprisingly, the only requirement for this is the ability to evaluate the target distribution; in particular, we need not be able to integrate or normalize it, or to compute the inverse of its cumulative distribution function—all steps which cause severe difficulty in traditional sampling approaches.

Defining a Markov chain amounts to defining a *transition rule*: a process for selecting a new state $\mathbf{x}_+$ randomly, in a way that depends on the current state $\mathbf{x}$. Metropolis et al. provided a way to take a transition rule that may not produce the desired stationary distribution π and turn it into one that does.

Given a method for sampling a new state $\mathbf{x}'$ from a *proposal distribution* $T(\mathbf{x}, \mathbf{x}')$, the Metropolis transition rule operates in two steps:

1. Choose $\mathbf{x}'$ according to the probability distribution $T(\mathbf{x}, \mathbf{x}')$.
2. $\mathbf{x}_+ = \begin{cases} \mathbf{x}' & \text{with probability } \min(1, \pi(\mathbf{x}')/\pi(\mathbf{x})) \\ \mathbf{x} & \text{otherwise} \end{cases}$

In step 1 we say $\mathbf{x}'$ is *proposed* as the next state, and in step 2 it is either *accepted* and becomes the next state, or it is *rejected* and the next state repeats the previous one. The probability $\min(1, \pi(\mathbf{x}')/\pi(\mathbf{x}))$ is known as the *acceptance probability*.

The original Metropolis algorithm only works when $T(\mathbf{x}, \mathbf{x}') = T(\mathbf{x}', \mathbf{x})$. In 1970, Hastings [17] proposed a new acceptance probability:

$$r(\mathbf{x}, \mathbf{x}') = \min\left\{1, \frac{\pi(\mathbf{x}')T(\mathbf{x}', \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x}, \mathbf{x}')}\right\} \tag{2.10}$$

which relaxes the symmetry restriction to one of symmetric support: $T(\mathbf{x}, \mathbf{x}')$ must be nonzero exactly when $T(\mathbf{x}', \mathbf{x})$ is nonzero. The Metropolis–Hastings algorithm is the starting point for MCMC rendering methods.

One noteworthy aspect of (2.10) is that the desired stationary distribution π occurs both in the denominator and numerator, which means that normalizing constants can be neglected without affecting the behavior of the Markov chain. This is important in many fields, where this constant may be unknown. The same holds true for T: any factors that are shared between the forward transition probability $T(\mathbf{x}, \mathbf{x}')$ and the reverse probability $T(\mathbf{x}', \mathbf{x})$ cancel. This useful in the context of rendering because it means that certain factors of T don't have to be computed, thus reducing the necessary amount of computation per transition.

The most common application of samples produced by MCMC methods is to estimate the expectation of a function g of $\mathbf{X}$, where $\mathbf{X}$ is distributed according

to the target distribution π, i.e.

$$E_\pi[g(\mathbf{X})] = \int_\Omega g(\mathbf{x})\,\pi(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} g(\mathbf{x}_i) \quad (N \in \mathbb{N}).$$

It is usually also necessary to discard an initial part of the sequence so that the bias introduced by the choice of $\mathbf{x}_1$ becomes negligible; this is referred to as the *burn-in period*. In many disciplines, the length of this burn-in period is an important parameter. For instance, when simulating the state of atoms in a high-dimensional molecular model, the starting position might be a rather unrealistic configuration of atoms, where a long sequence of transitions is necessary to move into a region of the state space where π takes on non-negligible positive values. In computer graphics, this is a much smaller concern, because the Markov chain can be seeded using a standard unbiased sampling technique. We use the bidirectional path tracing algorithm for this purpose and hence do not require a burn-in period[9].

The most important aspect of a Metropolis-type method is the choice of a suitable proposal distribution. Here, we seek to balance two conflicting goals: on one hand, we desire an algorithm that takes small steps so that it can easily explore small regions of the state space where π takes on large values. On the other hand, we wish that a subsequence of the Markov chain $\mathbf{x}_i, \ldots, \mathbf{x}_{i+k}$ captures a "representative" subset of the relevant parts of the state space, which means that it must move around quickly enough to do so (this is referred to as the *mixing ratio* of the chain). This matter is complicated by the fact that overly large steps tend to leave local maxima of π, thus causing them to be rejected immediately. Consequently, a chain taking large steps can produce long repetitions of the same state, which may effectively cause it to have a *worse*

[9]The details of this seeding process are discussed in Veach's PhD thesis [67], Section 11.3.1.

mixing ratio than that of a chain taking smaller steps.

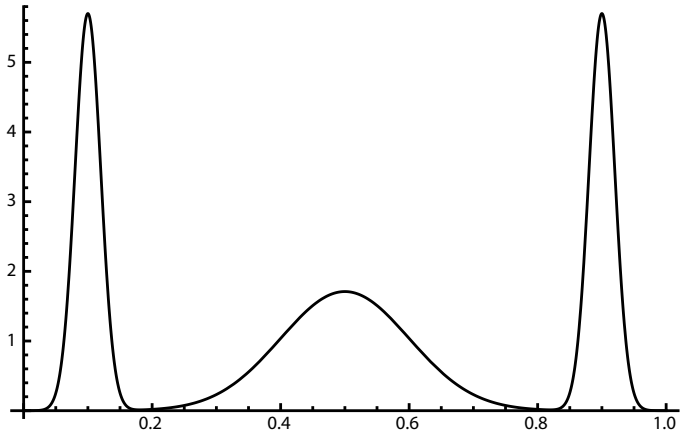
Figure 2.14 visualizes these tradeoffs using a simple example in which the Metropolis-Hasting algorithm is used to draw samples from a linear combination of one-dimensional normal distributions. We show the behavior of three different types of transitions rules (also referred to as *mutators*) by constructing histograms over the first 1000 visited states, always starting the chain at $x_1 = 0.5$. Ideally, the histogram should be in good agreement with the density function. The first mutator is given by

$$\textsc{Mutate}_1(x) := \text{generate a uniform sample on } [0, 1]$$

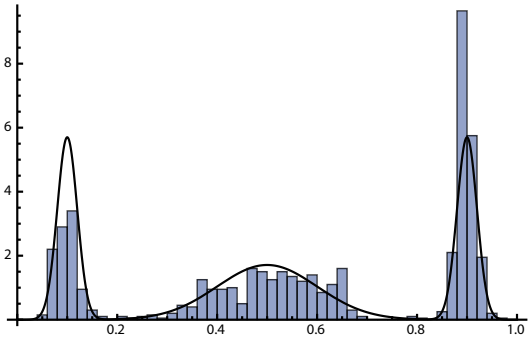
This is also known as an *independence sampler*, since the proposals are not correlated with the current state. Such an independence sampler is often needed for a MCMC algorithm to work properly, since it ensures that the underlying Markov chain satisfies the irreducibility and ergodicity properties discussed earlier. However, it does not lead to a good sampling method when used on its own, as can be seen in Figure 2.14 (b): due to many rejected proposals, the histogram is still fairly unconverged, and the estimated probability mass in the modes is considerably off. The second mutator is defined as

$$\textsc{Mutate}_2(x) := \text{generate a uniform sample on } [x - 0.05, x + 0.05]$$

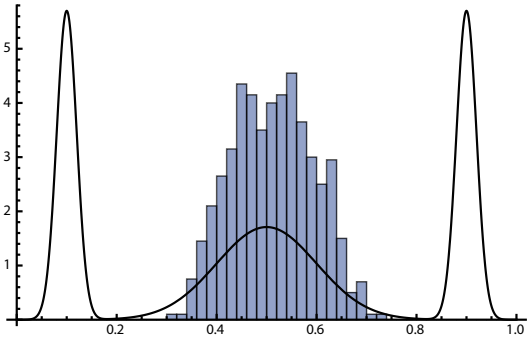
Veach refers to such a mutator as a *perturbation*, since it only makes minute changes to the current state. The histogram of $\textsc{Mutate}_2$ shown in Figure 2.14 (b) reveals that this strategy produced a smooth histogram that is a nonetheless a very poor approximation of the underlying target density. Due to the small steps taken and the low acceptance probability of proposals in regions where the target density is close to zero, this chain was effectively "stuck" in the middle



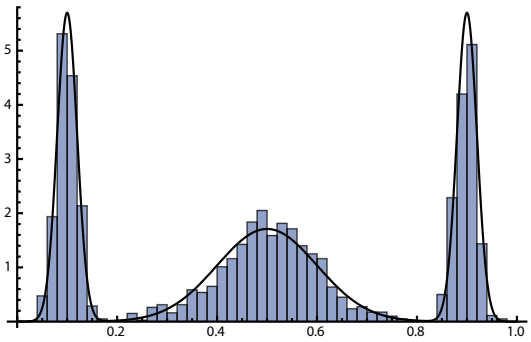
(a) Target density function



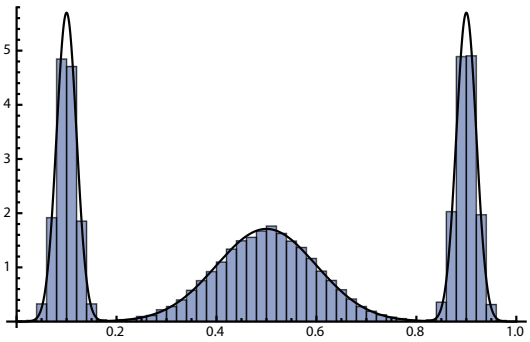
(b) Histogram obtained from MUTATE$_1$



(c) Histogram obtained using MUTATE$_2$



(d) Histogram obtained using MUTATE$_3$



(e) MUTATE$_3$ run for 10 000 iterations

Figure 2.14: Using the Metropolis-Hastings algorithm to generate samples from a simple 1D function

.

mode. The last mutator is defined as

$$\textsc{Mutate}_3(x) \coloneqq \text{use } \textsc{Mutate}_1 \text{ or } \textsc{Mutate}_2 \text{ with equal probability}$$

This finally leads to a good match in Figure 2.14 (c). Increasing the number of samples to 10000 in Figure 2.14 (d) improves the histogram further. Due to the superior performance of this combination, Metropolis-type rendering algorithms usually combine independent large-scale mutations and small-scale perturbations in much the same way.

When the domain is euclidean (e.g. $\Omega = \mathbb{R}^n$ for some $n \in \mathbb{N}$), a straightforward and commonly used proposal density is to use spherical symmetric normally distributed steps, i.e. $T(\mathbf{x}, \mathbf{x}') = f_N(\mathbf{x} - \mathbf{x}')$, where f_N is the n-dimensional density function of a normal distribution having mean 0 and variance σ^2. For a simple setup based on such proposals, Roberts and Gilks [53] determined that σ^2 should be set so that an asymptotic acceptance rate of 0.234 is maintained, which they found to maximize the statistical efficiency. While this result does not extend to general state spaces, it can serve as a good rule of thumb and mental warning note that acceptance rates too close to 100% can be detrimental, since the associated chain will likely not explore the space very well.

2.4.6 Metropolis Light Transport

The Metropolis Light Transport algorithm mentioned above introduced the tools of MCMC to rendering. In the rendering context, the state space is the space of all paths through the scene, points in the space are paths, and the desired probability distribution over paths is proportional to their contribution to the rendered image (i.e. the amount of illumination they carry to the camera). The final image is the projection of the path distribution into the image plane.

Following the notation of Veach, we indicate variables describing light paths using a bar marker, as in $\bar{\mathbf{x}}$. The target distribution π is equal to the contribution weighting function f of the measurement integral, which is covered in detail in Chapter 3. For now, f can be thought of as a black box that measures the differential amount of light traveling along a path. A simple pseudocode version of the algorithm looks as follows:

METROPOLIS-LIGHT-TRANSPORT()

1 $\bar{\mathbf{x}}_1 =$ Seed path created using BDPT

2 **for** $i = 2$ **to** N

3 $\quad \bar{\mathbf{x}}'_i = \text{MUTATE}(\bar{\mathbf{x}}_{i-1})$

4 $$\quad \bar{\mathbf{x}}_i = \begin{cases} \mathbf{x}'_i, & \text{with probability } \min\left\{1, \frac{f(\bar{\mathbf{x}}'_i)T(\bar{\mathbf{x}}'_i, \bar{\mathbf{x}}_{i-1})}{f(\bar{\mathbf{x}}_{i-1})T(\bar{\mathbf{x}}_{i-1}, \bar{\mathbf{x}}'_i)}\right\} \\ \mathbf{x}_{i-1}, & \text{otherwise} \end{cases}$$

5 $\quad$ Increase the luminance of the image pixel associated with $\bar{\mathbf{x}}_i$

6 Re-scale the image

We now describe each step in more detail:

Seed path generation (line 1): MLT is normally seeded with a path obtained from another rendering method. As mentioned earlier, the sampling scheme in unbiased techniques like plain path tracing and BDPT is usually not in perfect agreement with the underlying function to be integrated. As a consequence, each path must be assigned a *sampling weight* to account for the ratio between the integrand f and the actual sampling density.

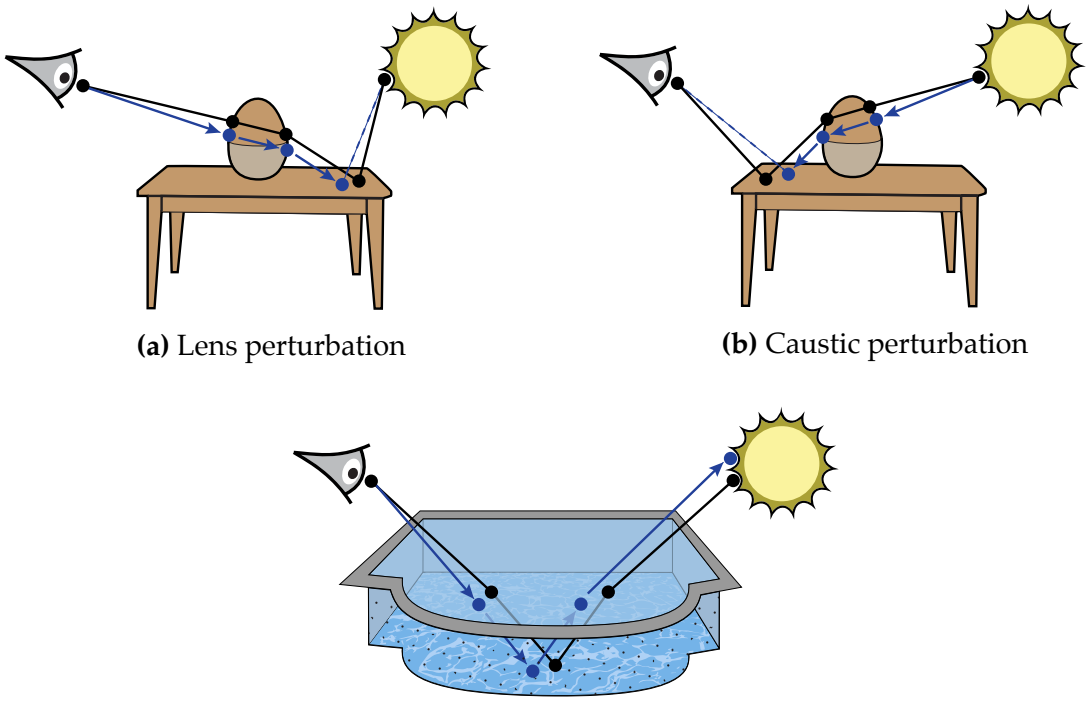
To seed the MLT algorithm in a practical way that eliminates the need for a burn-in phase, Veach proposed using BDPT to generate a large number of seed path *candidates* along with sampling weights. Following this, $\mathbf{x}_1$ is randomly chosen from the candidates with a probability that is proportional to their

weights. When rendering on a machine with multiple cores, one candidate is chosen for each core. In this case, the Markov chains are run in parallel, after which the final output image is created by averaging the output of the individual cores.

Mutations and Perturbations (line 3): At the center of an MCMC rendering algorithm is an implementation of a transition rule, and any rule with symmetric support is admissible. But to avoid very low acceptance probabilities, which lead to poor performance, it is desirable for the transition probability to approximate the contribution: that is, paths with more light flowing along them should be chosen more often. Veach's [70] transition rule is based on a set of *mutations* that change the structure of the path and *perturbations* that move the vertices by small distances while preserving the structure, both using the building blocks of bidirectional path tracing to sample paths. One of the following types of operations is randomly selected in each iteration:

(i) **Bidirectional mutation**: This mutation replaces a segment of an existing path with a newly generated segment (possibly of different length) drawn from a bidirectional-path-tracing-like sampling strategy. This essential mutation rule guarantees ergodicity and thus forms the backbone of the MLT algorithm. Due to the large-scale modifications of its proposals, many of them are ultimately rejected, and the bidirectional mutation therefore relies on the presence of additional perturbations to create a practical rendering technique.

(ii) **Lens subpath mutation**: The lens subpath mutation replaces a *lens subpath* (the path segment starting at the camera and reaching until the first non-specular vertex) with a different lens subpath that does not need to be



(a) Lens perturbation

(b) Caustic perturbation

(c) Multi-chain perturbation

Figure 2.15: An overview of the three different types of perturbations supported in Metropolis Light Transport.

similar to the preceding one. This transition rule thus makes large-scale changes to the end of a path.

(iii) **Lens perturbation**: This transition rule shown in Figure 2.15 (a) perturbs the outgoing direction at the camera and propagates the resulting path until the first non-specular material is encountered. It then attempts to create a connection to the (unchanged) remainder of the path. If the resulting path segment has a different length or *path configuration* (i.e. if the sequence of specular/nonspecular vertices does not match), it is rejected immediately. This conservative behavior is shared by all perturbations.

(iv) **Caustic perturbation**: The caustic perturbation works just like the lens perturbation, except that it proceeds in the other direction, starting from the light source.

(v) **Multi-chain perturbation**: This transition rule is used when there are several chains of specular interactions, as seen in the swimming pool example in Figure 2.15 (c). After an initial lens perturbation, a cascade of additional perturbations is performed until a connection to the remainder of the path can finally be established. Depending on the path type, the entire path may be changed by this.

The random selection of a mutation or perturbation in each iteration causes the underlying Markov chain transition matrix to turn into a linear combination of several different Metropolis-Hastings type transition matrices. This leads to a straightforward computer implementation but, strictly speaking, does not fit within the MCMC framework discussed so far.

To adhere to the framework, we would need to construct a single mutation that subsumes the effects of all desired path modifications, while accounting for potential overlaps between them when computing transition probabilities (for instance, the lens perturbation and bidirectional mutation might propose the same modification with different probabilities).

Tierney [63] showed that under certain conditions, both of these approaches are acceptable: in particular, a MCMC method that only computes transition probabilities with respect to the currently chosen mutation still has the correct stationary distribution, as long as each one of the mutators preserves it. Hence, we rely on this simpler variant.

Acceptance/Rejection (line 4): The acceptance/rejection step in practice closely follows the given pseudocode. Unchanged path segments cause identical terms to appear in the nominator and denominator of the acceptance probability, which do not need to be computed in an implementation—this can considerably accelerate mutations involving long paths.

Recording the path in the output image (line 5): In each iteration, MLT determines the pixel associated with the current path and increases its intensity by a fixed amount.

Re-scaling the output image (line 6): One issue with the algorithm outlined so far is that it essentially creates a histogram of the lighting distribution in image space. Image luminances are only recovered in a relative sense: MLT can for instance determine that a certain pixel is approximately twice as bright as another one, but an absolute scale factor is still needed to turn it into a valid rendering[10].

This scale factor is usually found by computing the average image luminance using a standard unbiased sampling technique like unidirectional or bidirectional path tracing. This computation can be conveniently integrated into the seeding phase (line 1) at almost no extra cost.

Additional details: So far, we have neglected to explain how color information is handled in the rendering process. Commonly, the MLT target density is chosen as the luminance of the spectral or RGB-valued contribution function f so that paths are sampled proportional to the luminance they carry to the camera. To retain color information, step 5 is modified so that it adds the value of f divided by its luminance to the corresponding pixel, rather than simply incrementing its value.

[10] Some tonemapping techniques are invariant to scaling of the input and hence do not require this step.

2.4.7 Other MCMC rendering methods

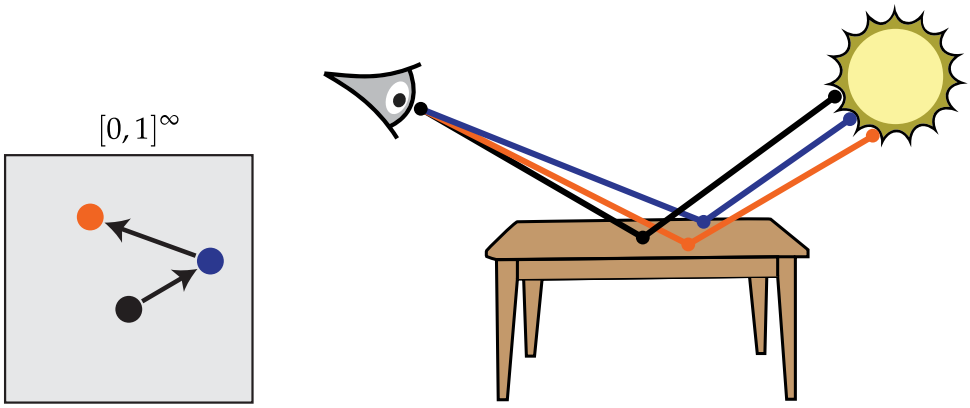
A considerably simpler MLT variant was later proposed by Kelemen et al. [32], which we will refer to as *Primary Sample Space MLT* (PSSMLT). Like Veach's MLT, this method explores the space of light paths, searching with preference for those that carry a significant amount of energy from an emitter to the sensor. The main difference is that PSSMLT does this exploration by "piggybacking" on another rendering technique and manipulating the random number stream that drives it, whereas MLT operates directly on light paths.

The main insight of Kelemen et al.'s approach is that unbiased sampling strategies like uni- or bidirectional path tracing can be interpreted as a combination of a uniformly distributed random number generator on the interval $[0, 1]$ and a deterministic mapping from a sequence of these random numbers to a path in $\mathcal{P}$. If we define $\Omega :=$"$[0, 1]^\infty$" with a slight abuse of notation to be the infinite-dimensional space containing all possible realizations of a sequence of uniform variates and let $\Phi : \Omega \to \mathcal{P}$ denote the aforementioned mapping, where w_j is the associated sampling weight for pixel j, then this algorithm computes the same answer I_j using an integral of the following form

$$I_j := \int_\Omega w_j(\Phi(\mathbf{x}))\, \mathrm{d}\mathbf{x} \tag{2.11}$$

using mutation and perturbation strategies that operate directly on the *primary sample space* Ω (Figure 2.16). Due to path termination criteria such as russian roulette, only a finite (but random) number of uniform variates is required in practice, which makes the approach feasible. This algorithm has several desirable properties: first, due to the simple structure of this space, a symmetric proposal density can be used, which removes the need to compute transition probabilities[11] when computing acceptance probabilities. Another important

[11] This can be a rather difficult part of mutations on path space.



(a) Primary sample space view **(b)** Path space view

Figure 2.16: Primary Sample Space MLT performs mutations in an abstract random number space. A deterministic mapping Φ induces corresponding mutations in path space.

aspect of the method lies in the generality of Equation (2.11). The function Φ is a black-box mapping that could denote virtually any Monte Carlo rendering algorithm, including path tracing, particle tracing, and bidirectional path tracing. Finally, perturbations within primary sample space tend to interact nicely with the implemented importance sampling strategies in a way that gives them the right "scale". For instance, when perturbing the coordinates that are used to sample the outgoing direction on a surface, the magnitude of the direction change is generally related to how glossy this material is.

The main disadvantage is the considerable loss of flexibility when compared to the original MLT algorithm. Because of the black-box nature of the mapping Φ, the behavior of the different dimensions of Ω can be difficult to predict: for instance, a small change to the first coordinate may cause a ripple change that causes large-scale modifications to later parts of a path. A useful feature of the MLT scheme by Veach is that it can construct a path from one direction (e.g. from the camera to the light source) and later perturb it in the other direction. It is also possible to expand or contract subpaths by inserting or removing vertices.

Such operations are not available when using the simpler primary sample space.

In the field of applied mathematics, Doucet et al. [9] proposed a MCMC algorithm for solving general Fredholm and Volterra equations of the second kind. When translating their method into the context of the measurement integral (2.3), one arrives at a method that resembles the MLT algorithm and supports perturbations in addition to path contractions and expansions. Because it only modifies one vertex at a time, the resulting Markov chain is not guaranteed to converge to the correct stationary distribution, which Veach points out in [67].

Considerable research activity has extended Metropolis light transport in various ways. Pauly et al. [47] proposed a perturbation rule for rendering participating media with single scattering. Other projects include Metropolis Instant Radiosity [54], Population Monte Carlo rendering [36], and Replica Exchange light transport [34]. Recently, two groups [3, 15] have combined the transition rule of Kelemen et al. with photon mapping to obtain robust methods based on density estimation.

However, to generate proposals, all of these algorithms ultimately rely on local path sampling strategies (i.e. path tracing). Specifically, they choose the next interaction vertex along a light path by sampling from a directional distribution associated with the current vertex, followed by an intersection search. In this dissertation, we introduce a new kind of transition rule with different properties.

The original MLT algorithm and subsequent variants all render an image by running a Markov chain for a long (e.g. $> 10^6$) sequence of steps, and they guarantee ergodicity using a transition rule that can generate any path in the domain with some probability. These methods can in practice suffer from sufficient control over the distribution of samples in image space, which

slows down convergence. In principle, the lens subpath mutation attempts to spread out samples in image space, but its large-scale nature means that these mutations are likely to be rejected.

The Energy Redistribution path tracing (ERPT) method by Cline et al. [6], which is readily adapted to work with our method, is an interesting departure from MLT. It draws on the property that the Metropolis-Hastings algorithm preserves the stationary distribution of samples even if the underlying transition rule is *not* ergodic (e.g. when it cannot reach certain parts of path space). This requires that the original samples that are used to seed the Markov chain already have the right distribution.

ERPT then works as follows: in a first step, it samples a large set of paths via standard path tracing. Due to the limitations of unbiased sampling, these paths are not distributed proportionally to the target distribution, hence each one is assigned a sampling weight by the path tracer—however, together with those weights, they can be thought of as being drawn from the correct distribution. Following this, ERPT runs Markov chains for short bursts ($\approx 10^3$ steps) starting at each sample. It uses the same perturbations as the original MLT algorithm, namely the lens, caustic, and multi-chain perturbation. The relaxation of the ergodicity requirement makes it possible to dispense with the bidirectional and lens subpath mutation, creating a method that explores paths in a very local fashion. The main advantage of ERPT is that it enables the use of an initial set of path samples that is carefully chosen to have a good distribution, e.g. one path per pixel.

CHAPTER 3

PATH SPACE FOR VOLUMES AND SURFACES

One of the main motivations for using path space is that it provides an *explicit* expression for the value of a measurement as an integral over paths, as opposed to the unwieldy recursive integration encountered in Section 2.4.1. The explicit form allows for considerable freedom in how these paths are found—essentially any technique for randomly choosing paths can be turned into a workable rendering algorithm that computes the right answer given enough time.

Having discussed the underlying energy balance equations, we will now show how they can be used as the starting point for a derivation of a path-space framework suitable for the construction of advanced rendering algorithms.

Although each of these steps is in principle well-understood, there currently exists no detailed derivation that accounts for the effects of both surfaces and volumes (the closest being work by Pauly et al. [47], which only supplies definitions). For completeness, we therefore provide one here. Note that we postpone all treatment of specularity to Chapter 4 and focus purely on the non-specular case for now. Readers who are familiar with path space and interested in the main results of this thesis may consider skipping over this chapter.

3.1 Integral form of the radiative transfer equation

As a prerequisite, we require a full specification of the underlying conservation laws of light transport in *integral equation* form. The surface equation (2.2) already satisfies this condition and can be used as-is.

Since the RTE is an *integro-differential* equation, we begin by converting it

into a pure integral form. As an intermediate result, we will obtain an operator-based description of surface and volume light transport similar in style to the frameworks presented by Arvo [1] and Veach [67], which we then use to derive the path-space measurement integral. We roughly follow the approach of Chapter 4 in Eric Veach's Ph.D. thesis [67] and refer to this work for a detailed discussion of the notation used.

Definitions: We shall make use of the following notation to express integration along ray segments

$$\int_{\mathbf{x}}^{\mathbf{y}} f(\mathbf{z})\,\mathrm{d}\mathbf{z} := \int_0^{\|\mathbf{x}-\mathbf{y}\|} f(\mathbf{x}+t\,\overrightarrow{\mathbf{xy}})\,\mathrm{d}t, \quad \text{where} \quad \overrightarrow{\mathbf{xy}} := \frac{\mathbf{y}-\mathbf{x}}{\|\mathbf{y}-\mathbf{x}\|}.$$

and

$$\int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{z})\,\mathrm{d}\mathbf{z} := \int_0^{d_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{x}+t\omega)\,\mathrm{d}t.$$

where $\mathbf{x}_{\mathcal{M}}$ was defined in (2.4). When dealing with rays that do not intersect any surfaces ($d_{\mathcal{M}}(x,\omega)=\infty$), we consider the ray-casting operator $\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)$ to return points at infinity.

Recall the radiative transfer equation (2.1):

$$(\omega\cdot\nabla)\,L(\mathbf{x},\omega)+\sigma_t(\mathbf{x})\,L(\mathbf{x},\omega) = \sigma_s(\mathbf{x})\int_{S^2} f_p(\mathbf{x},\omega'\to\omega)\,L(\mathbf{x},\omega')\,\mathrm{d}\sigma_{\mathbf{x}}(\omega')+L_e(\mathbf{x},\omega), \qquad \mathbf{x}\in\Omega^\circ,$$

where Ω° was the interior of the domain. Restricted to a line of direction ω parameterized by $s\in\mathbb{R}$, it can be rewritten as a one-dimensional ordinary differential equation of the form

$$L'(s)+\sigma_t(s)L(s)=Z(s)$$

where $Z(s)$ represents the emitted and in-scattered radiance at s. Given an

arbitrary initial condition at $s = 0$, this ODE has the solution

$$\begin{aligned} L(s) &= \exp\left(-\int_0^s \sigma_t(r)\,\mathrm{d}r\right)\left[L(0) + \int_0^s \exp\left(\int_0^t \sigma_t(r)\,\mathrm{d}r\right) Z(t)\,\mathrm{d}t\right] \\ &= \exp\left(-\int_0^s \sigma_t(r)\,\mathrm{d}r\right) L(0) + \int_0^s \exp\left(-\int_t^s \sigma_t(r)\,\mathrm{d}r\right) Z(t)\,\mathrm{d}t \end{aligned} \tag{3.1}$$

where the first term corresponds to attenuated radiance from the surface at $s = 0$ and the second term accounts for in-scattered radiance.

In this solution, the parameterization and boundary condition at $s = 0$ were chosen arbitrarily. In general, the boundary condition will be specified by the nearest surface visible along the ray $(\mathbf{x}, -\omega)$. To formalize this, we will first define the *reduced surface radiance* at $(\mathbf{x}, \omega)$ as

$$L_{\text{red}}(\mathbf{x}, \omega) \coloneqq \begin{cases} L_o(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega), \omega)\,\tau(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega) \leftrightarrow \mathbf{x}), & d_{\mathcal{M}}(\mathbf{x}, -\omega) < \infty \\ 0, & \text{otherwise} \end{cases}$$

where

$$\tau(\mathbf{x} \leftrightarrow \mathbf{y}) \coloneqq \exp\left(-\int_{\mathbf{x}}^{\mathbf{y}} \sigma_t\left(\mathbf{z}\right)\,\mathrm{d}\mathbf{z}\right)$$

and L_o represents the surface radiance scattered into direction ω. By changing to a suitable parameterization and substituting the appropriate expressions for $L(0)$ and $Z(t)$ into Equation (3.1), the first term becomes the reduced surface radiance and we arrive at the integral form of the radiative transfer equation (Figure 3.1):

$$\begin{aligned} L(\mathbf{x}, \omega) = L_{\text{red}}(\mathbf{x}, \omega) + \int_0^{d_{\mathcal{M}}(\mathbf{x}, -\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{x} - t\omega)\Big(\sigma_s(\mathbf{x} - t\omega) \\ \int_{S^2} f_p(\mathbf{x} - t\omega, \omega' \to \omega) L(\mathbf{x} - t\omega, \omega')\,\mathrm{d}\sigma_{\mathbf{x}-t\omega}(\omega') + L_e(\mathbf{x} - t\omega, \omega)\Big)\mathrm{d}t \end{aligned}$$

which can be written more compactly using the notation introduced earlier:

$$\begin{aligned} L(\mathbf{x}, \omega) = L_{\text{red}}(\mathbf{x}, \omega) + \\ \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y})\left(\sigma_s(\mathbf{y}) \int_{S^2} f_p(\mathbf{y}, \omega' \to \omega) L(\mathbf{y}, \omega')\,\mathrm{d}\sigma_{\mathbf{y}}(\omega') + L_e(\mathbf{y}, \omega)\right)\mathrm{d}\mathbf{y}. \end{aligned} \tag{3.2}$$

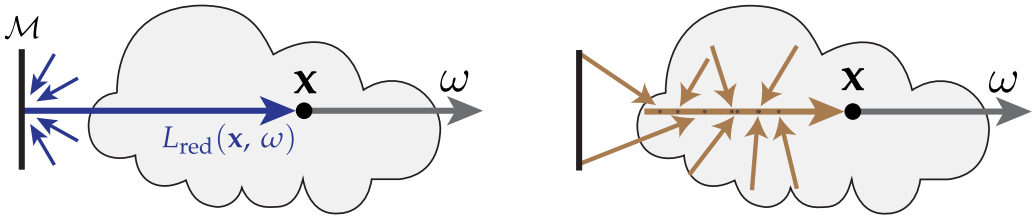


Figure 3.1: The integral form of the RTE describes radiance at $(\mathbf{x}, \omega)$ as the sum of the reduced surface radiance and an integral over in-scattered radiance.

Contrasting this equation with the integral equation for surfaces (2.2), one conspicuous difference is that the surface equation is expressed in terms of the incident and exitant radiance functions L_i and L_o, whereas 3.2 is given in terms of L. Away from surfaces, the distinction between L_i and L_o is technically not necessary (involving only a change of direction, see Section 2.2.4) but we introduce it here for the purpose of a more uniform notation. Expressed in terms of L_i and L_o, the integral form of the RTE (3.2) is given by

$$L_o(\mathbf{x}, \omega) = L_{\text{red}}(\mathbf{x}, \omega) + \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x}, -\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \left(\sigma_s(\mathbf{y}) \int_{S^2} f_p(\mathbf{y}, -\omega' \to \omega) L_i(\mathbf{y}, \omega') \, \mathrm{d}\sigma_{\mathbf{y}} + L_e(\mathbf{y}, \omega) \right) \mathrm{d}\mathbf{y}. \quad (3.3)$$

For completeness, we repeat the associated surface boundary condition (2.2):

$$L_o(\mathbf{x}, \omega) = \int_{S^2} f_s(\mathbf{x}, \omega' \to \omega) L_i(\mathbf{x}, \omega') \, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega') + L_e(\mathbf{x}, \omega), \quad \mathbf{x} \in \mathcal{M}.$$

Note the negated ω' term in the spherical integral (3.3), which has a positive correspondence in (2.2). This discrepancy is simply due to the different parameter conventions of surface and volume scattering models.[1]

[1] As a somewhat unfortunate consequence of notations in different fields, in volume scattering models, the incident direction argument points *towards* the scattering location, whereas it points *away* in the surface case.

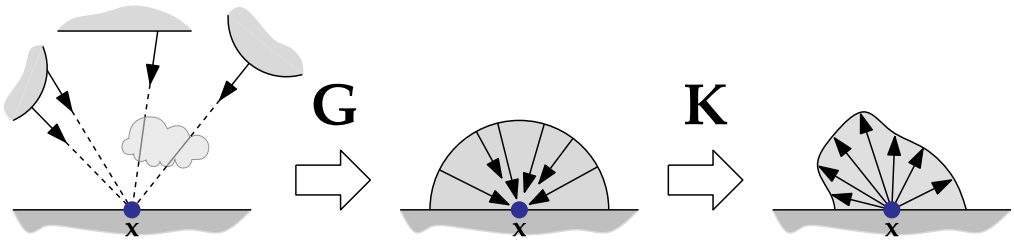


Figure 3.2: Illustration of the effects of **G** and **K** operators at points that lie on a surface.

3.2 Operator notation

Volumetric light transport can be seen as the alternation of two steps: *scattering* on a surface or within the medium, followed by *propagation* and *attenuation*. Analogous to [67] and [1], these two steps can be formulated as linear operators defined on the space of radiance functions. The goal of this section is to partition Equations (2.2) and (3.3) so that they can be expressed in this manner. We define the scattering operator **K** as one of the in-scattering integrals in Equations (2.2) or (3.3) dependent on whether or not $\mathbf{x} \in \mathcal{M}$:

$$(\mathbf{K}h)(\mathbf{x},\omega) := \begin{cases} \displaystyle\int_{S^2} f_s(\mathbf{x},\omega' \to \omega)\, h(\mathbf{x},\omega')\, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega'), & \mathbf{x} \in \mathcal{M} \\ \sigma_s(\mathbf{x}) \displaystyle\int_{S^2} f_p(\mathbf{x},-\omega' \to \omega)\, h(\mathbf{x},\omega')\, \mathrm{d}\sigma_{\mathbf{x}}(\omega'), & \text{otherwise} \end{cases} \tag{3.4}$$

On surfaces, **K** turns incident radiance into outgoing radiance. On the interior of the domain, it turns incident radiance into outgoing radiance per unit length.

In comparison, the transport operator **G** has a single definition on the whole domain. Its role is to transform outgoing radiance per unit length from the volume and outgoing radiance from surfaces into incident radiance.

$$\begin{aligned} (\mathbf{G}h)(\mathbf{x},\omega) := \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} & \tau(\mathbf{x} \leftrightarrow \mathbf{y})\, h(\mathbf{y},-\omega)\, \mathrm{d}\mathbf{y} \\ & + \tau(\mathbf{x} \leftrightarrow \mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega))\, h(\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega),-\omega) \end{aligned} \tag{3.5}$$

This leads to interesting differences in comparison to previous work: to match

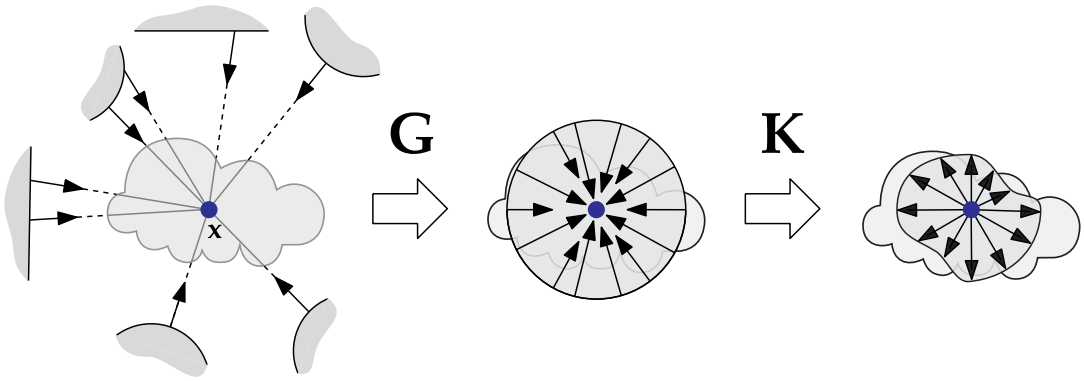


Figure 3.3: Illustration of the effects of **G** and **K** operators at points that lie in a volume.

equations (2.2) and (3.3), we arrive at an equilibrium equation of the form

$$L_i = \mathbf{G}\left(\mathbf{K}L_i + L_e\right) \tag{3.6}$$

whereas $L_o = L_e + \mathbf{KG}L_o$ was used by [67]. The differences arise since volume light transport forces us to deal with both radiance and radiance per unit length. To express the equilibrium condition purely in terms of radiance, the order of the **G** and **K** operators must be reversed. Assuming invertibility, the solution operator **S** can now be found:

$$\begin{aligned} & L_i = \mathbf{G}\left(\mathbf{K}L_i + L_e\right) \\ \Leftrightarrow\ & \left(\mathbf{I} - \mathbf{GK}\right) L_i = \mathbf{G}L_e \\ \Leftrightarrow\ & L_i = \underbrace{\left(\mathbf{I} - \mathbf{GK}\right)^{-1}\mathbf{G}}_{=:\,\mathbf{S}} L_e. \end{aligned}$$

When the following Neumann series converges in operator norm, the solution operator can also be expressed as

$$\mathbf{S} = \sum_{k=0}^{\infty} \left(\mathbf{GK}\right)^k \mathbf{G}. \tag{3.7}$$

3.3 Unified path integral formulation

Having defined the necessary operators, we can now proceed to find the associated path integral formulation. Note that in the following, we will often switch between parameterizing functions in terms of directions and positions. The arrow direction indicates the flow of light, which leads to the following notational conventions:

$$\begin{aligned}
L_e(\mathbf{x} \to \mathbf{y}) &:= L_e(\mathbf{x}, \overrightarrow{\mathbf{xy}}) \\
W_e(\mathbf{x} \leftarrow \mathbf{y}) &:= W_e(\mathbf{x}, \overrightarrow{\mathbf{xy}}) \\
f_s(\mathbf{x} \to \mathbf{y} \to \mathbf{z}) &:= f_s(\mathbf{y}, \overrightarrow{\mathbf{yx}}, \overrightarrow{\mathbf{yz}}) \\
f_p(\mathbf{x} \to \mathbf{y} \to \mathbf{z}) &:= f_p(\mathbf{y}, \overrightarrow{\mathbf{xy}}, \overrightarrow{\mathbf{yz}}) \\
(\mathbf{G}h)(\mathbf{x} \to \mathbf{y}) &:= (\mathbf{G}h)(\mathbf{x}, \overrightarrow{\mathbf{xy}})
\end{aligned}$$

Change of variables

When dealing with light transport on surfaces, it is often convenient to switch between integration over a sphere and integration over all surfaces of the scene, e.g.

$$\int_{S^2} f(\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega))\, \mathrm{d}\sigma^{\perp}_{\mathbf{x}}(\omega) = \int_{\mathcal{M}} f(\mathbf{y})\, \tilde{G}_{\text{surf}}(\mathbf{x} \leftrightarrow \mathbf{y})\, \mathrm{d}A(\mathbf{y}).$$

This change of variables involves the geometric term for surfaces [49]

$$\tilde{G}_{\text{surf}}(\mathbf{x} \leftrightarrow \mathbf{y}) = V(\mathbf{x} \leftrightarrow \mathbf{y}) \cdot \frac{\left|\cos\theta_{\mathbf{x}} \cos\theta_{\mathbf{y}}\right|}{\|\mathbf{x} - \mathbf{y}\|^2}$$

where $\theta_{\mathbf{x}}$ and $\theta_{\mathbf{y}}$ denote the angles that $\overrightarrow{\mathbf{xy}}$ makes with the surface normals $N(\mathbf{x})$ and $N(\mathbf{y})$, respectively, and V is a visibility function defined as

$$V(\mathbf{x} \leftrightarrow \mathbf{y}) := \begin{cases} 1, & \text{if } \{\alpha\mathbf{x} + (1-\alpha)\mathbf{y} \mid \alpha \in (0,1)\} \cap \mathcal{M} = \emptyset \\ 0, & \text{otherwise} \end{cases}$$

In the volume setting, a similar change of variables is possible: we consider a volume integral over rays radially emanating from a point $\mathbf{x}$. Fubini's theorem and some manipulation leads to

$$\begin{aligned}
&\int_{S^2} \int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} f(\mathbf{y}) \, \mathrm{d}\mathbf{y} \, \mathrm{d}\omega \\
&= \int_{S^2} \int_0^\infty f(\mathbf{x} + r\omega) V(\mathbf{x} \leftrightarrow \mathbf{x} + r\omega) \, \mathrm{d}r \, \mathrm{d}\omega \\
&= \int_0^\infty \frac{1}{r^2} \int_{S^2(\mathbf{x},r)} f(\mathbf{y}) V(\mathbf{x} \leftrightarrow \mathbf{y}) \, \mathrm{d}A(\mathbf{y}) \, \mathrm{d}t \\
&= \int_\Omega \frac{V(\mathbf{x} \leftrightarrow \mathbf{y})}{\|\mathbf{x} - \mathbf{y}\|^2} f(\mathbf{y}) \, \mathrm{d}V(\mathbf{y}).
\end{aligned}$$

To handle all possible combinations of volume and surface endpoints, we define the basic geometry term as follows:

$$\tilde{G}(\mathbf{x} \leftrightarrow \mathbf{y}) := V(\mathbf{x} \leftrightarrow \mathbf{y}) \cdot \frac{D_{\mathbf{x}}\left(\overrightarrow{\mathbf{xy}}\right) D_{\mathbf{y}}\left(\overrightarrow{\mathbf{yx}}\right)}{\|\mathbf{x} - \mathbf{y}\|^2}$$

where

$$D_{\mathbf{a}}(\omega) := \begin{cases} |N(\mathbf{a}) \cdot \omega|, & \mathbf{a} \in \mathcal{M} \\ 1, & \text{otherwise} \end{cases}$$

3.4 Path space measurement integral

Based on the generalized operators $\mathbf{G}$ and $\mathbf{K}$, we can proceed to find a path integral formulation of light transport similar to what is done in Chapter 8 of Eric Veach's thesis [67].

We begin by considering a measurement I_j, which is defined as the inner product of the importance $W_e^{(j)}$ emitted by sensor j and the incident radiance L_i:

$$\begin{aligned}
I_j &= \langle W_e^{(j)}, L_i \rangle \\
&= \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}).
\end{aligned} \tag{3.8}$$

For simplicity, we restrict the ray origins to the boundary, hence volumetric sensors are not supported[2].

The goal of the path space approach is to express the measurement integral (3.8) over the set of transport paths $\mathcal{P}$

$$I_j = \int_{\mathcal{P}} f_j(\bar{x}) \, \mathrm{d}\mu(\bar{x}),$$

where μ is a measure on $\mathcal{P}$, and f_j is a contribution weighting function specific to the measurement.

To render the above precise, let us define the path space $\mathcal{P}$ as the union of all fixed-length paths formed by concatenating vertices from $\mathcal{M}$ and Ω° according to a configuration vector $\mathbf{c} \in \{0,1\}^k$, i.e.

$$\mathcal{P} \coloneqq \bigcup_{k=1}^{\infty} \bigcup_{\mathbf{c} \in \{0,1\}^k} \mathcal{P}_k^{\mathbf{c}} \tag{3.9}$$

where $\mathcal{P}_k^{\mathbf{c}}$ is defined using a Cartesian product:

$$\mathcal{P}_k^{\mathbf{c}} \coloneqq \bigtimes_{i=1}^{k} \begin{cases} \mathcal{M}, & \text{if } \mathbf{c}_i = 0 \\ \Omega^\circ, & \text{if } \mathbf{c}_i = 1 \end{cases}$$

Using the Lebesgue measures for area and volume on $\mathcal{M}$ and Ω°, we can define a combined product measure on $\mathcal{P}$:

$$\mu(D) \coloneqq \sum_{k=1}^{\infty} \sum_{\mathbf{c} \in \{0,1\}^k} \mu_k^{\mathbf{c}}(D \cap \mathcal{P}_k^{\mathbf{c}}) \quad \text{where} \quad \mu_k^{\mathbf{c}}(D) \coloneqq \int_D \prod_{i=1}^{k} \begin{cases} \mathrm{d}A(\mathbf{x}_i), & \text{if } \mathbf{c}_i = 0 \\ \mathrm{d}V(\mathbf{x}_i), & \text{if } \mathbf{c}_i = 1 \end{cases}$$

To find the path space formulation, let us insert the operator form of the equilibrium equation (3.6) into the measurement integral (3.8):

$$I_j = \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega)(\mathbf{G}(\mathbf{K}L_i + L_e))(\mathbf{x}, \omega) \, \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}).$$

[2]But it would be straightforward to add by extending the definition of W_e appropriately and defining (3.8) as a sum of integrals over surfaces and volumes.

Expanding the $\mathbf{G}$ operator leads to

$$= \int_{\mathcal{M}} \int_{S^2} W_e^{(j)}(\mathbf{x}, \omega) \left[\int_{\mathbf{x}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{x},\omega)} \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \, [\mathbf{K}L_i + L_e](\mathbf{y}, -\omega) \, \mathrm{d}\mathbf{y} \right.$$
$$\left. + \, \tau(\mathbf{x} \leftrightarrow \mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega)) \, [\mathbf{K}L_i + L_e](\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega), -\omega) \right] \mathrm{d}\sigma_{\mathbf{x}}^{\perp}(\omega) \, \mathrm{d}A(\mathbf{x}).$$

Using a change of variables (see Section 3.3), we can turn the two summands into a volume and a surface integral, respectively:

$$= \int_{\mathcal{M}} \left[\int_{\Omega^\circ} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y}) \, \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \, \tilde{G}(\mathbf{x} \leftrightarrow \mathbf{y}) \, [\mathbf{K}L_i + L_e](\mathbf{y} \to \mathbf{x}) \, \mathrm{d}V(\mathbf{y}) \right.$$
$$\left. + \int_{\mathcal{M}} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y}) \tau(\mathbf{x} \leftrightarrow \mathbf{y}) \, \tilde{G}(\mathbf{x} \leftrightarrow \mathbf{y}) [\mathbf{K}L_i + L_e](\mathbf{y} \to \mathbf{x}) \, \mathrm{d}A(\mathbf{y}) \right] \mathrm{d}A(\mathbf{x}).$$

Using the fact that $\mathbf{K}L_i = \mathbf{K}\mathbf{S}L_e = \sum_{k=1}^{\infty} (\mathbf{K}\mathbf{G})^k L_e$, this can be rewritten as

$$= \sum_{k=0}^{\infty} \int_{\mathcal{M}} \left[\int_{\Omega^\circ} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y}) \, G(\mathbf{x} \leftrightarrow \mathbf{y}) \, [(\mathbf{K}\mathbf{G})^k L_e](\mathbf{y} \to \mathbf{x}) \, \mathrm{d}V(\mathbf{y}) \right.$$
$$\left. + \int_{\mathcal{M}} W_e^{(j)}(\mathbf{x} \leftarrow \mathbf{y}) \, G(\mathbf{x} \leftrightarrow \mathbf{y}) [(\mathbf{K}\mathbf{G})^k L_e](\mathbf{y} \to \mathbf{x}) \, \mathrm{d}A(\mathbf{y}) \right] \mathrm{d}A(\mathbf{x}). \quad (3.10)$$

where we have defined a new geometric term that also accounts for attenuation:

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) \coloneqq \tilde{G}(\mathbf{x} \leftrightarrow \mathbf{y}) \, \tau(\mathbf{x} \leftrightarrow \mathbf{y}).$$

In the above geometric sum, the operators $\mathbf{G}$ and $\mathbf{K}$ are reversed in comparison to previous encounters (e.g. Equation (3.7)). This makes it possible to perform additional simplifications: consider the concatenated operator $\mathbf{K}\mathbf{G}$ for $\mathbf{y} \in \mathcal{M}$:

$$(\mathbf{K}\mathbf{G}h)(\mathbf{y} \to \mathbf{x}) = \int_{S^2} f_s \left(\mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega') \to \mathbf{y} \to \mathbf{x} \right) \left[\int_{\mathbf{y}}^{\mathbf{x}_{\mathcal{M}}(\mathbf{y},\omega')} \tau(\mathbf{y} \leftrightarrow \mathbf{z}) \, h(\mathbf{z}, -\omega') \, \mathrm{d}\mathbf{z} \right.$$
$$\left. + \tau(\mathbf{y} \leftrightarrow \mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega')) \, h(\mathbf{x}_{\mathcal{M}}(\mathbf{y}, \omega'), -\omega') \right] \mathrm{d}\sigma_{\mathbf{y}}^{\perp}(\omega').$$

As before, we can perform a change of variables to volume and surface integrals:

$$= \int_{\Omega^\circ} f_s\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}V(\mathbf{z}) + \int_{\mathcal{M}} f_s\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}A(\mathbf{z})$$

For $\mathbf{y} \in \Omega^\circ$, we get

$$(\mathbf{KG}h)(\mathbf{y} \to \mathbf{x}) = \sigma_s \left[\int_{\Omega^\circ} f_p\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}V(\mathbf{z}) + \int_{\mathcal{M}} f_p\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}A(\mathbf{z}) \right]$$

To unify the two cases, we define the generalized scattering function $\bar{f}$ as

$$\bar{f}(\mathbf{z} \to \mathbf{y} \to \mathbf{x}) := \begin{cases} \sigma_s f_p(\mathbf{z} \to \mathbf{y} \to \mathbf{x}), & \mathbf{y} \in \Omega^\circ \\ f_s(\mathbf{z} \to \mathbf{y} \to \mathbf{x}), & \mathbf{y} \in \mathcal{M} \end{cases} \tag{3.11}$$

which leads to a single definition on the whole domain:

$$(\mathbf{KG}h)(\mathbf{y} \to \mathbf{x}) = \int_{\Omega^\circ} \bar{f}\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}V(\mathbf{z}) + \int_{\mathcal{M}} \bar{f}\left(\mathbf{z} \to \mathbf{y} \to \mathbf{x}\right) G(\mathbf{y} \leftrightarrow \mathbf{z})\, h(\mathbf{z} \to \mathbf{y})\, \mathrm{d}A(\mathbf{z}).$$

Inserting increasing powers of **KG** into Equation (3.10) leads to a nested integral. A cumbersome but straightforward rearrangement of its terms results in the following compact representation on path space:

$$I_j = \int_{\mathcal{P}} f_j(\bar{x})\, \mathrm{d}\mu(\bar{x}) \tag{3.12}$$

where

$$f_j(\bar{\mathbf{x}}) = f_j(\mathbf{x}_1 \cdots \mathbf{x}_n) = L_e(\mathbf{x}_1 \to \mathbf{x}_2) \left[\prod_{k=2}^{n-1} \bar{f}(\mathbf{x}_{k-1} \to \mathbf{x}_k \to \mathbf{x}_{k+1})\, G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k) \right] \cdot\ G(\mathbf{x}_{n-1} \leftrightarrow \mathbf{x}_n)\, W_e^{(j)}(\mathbf{x}_{n-1} \to \mathbf{x}_n). \tag{3.13}$$

Due to the definition of path space (3.9) and its associated measure, integrating f_j over $\mathcal{P}$ implies integration of f_j over all 2^k configurations for each $k = 1, \ldots, \infty$.

$$f_j(\mathbf{x}) = L_e(\mathbf{x}_1 \to \mathbf{x}_2)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)\, \bar{f}(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3)\, G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3)$$
$$\bar{f}(\mathbf{x}_2 \to \mathbf{x}_3 \to \mathbf{x}_4)\, G(\mathbf{x}_3 \leftrightarrow \mathbf{x}_4)\, W_e^{(j)}(\mathbf{x}_3 \to \mathbf{x}_4)$$

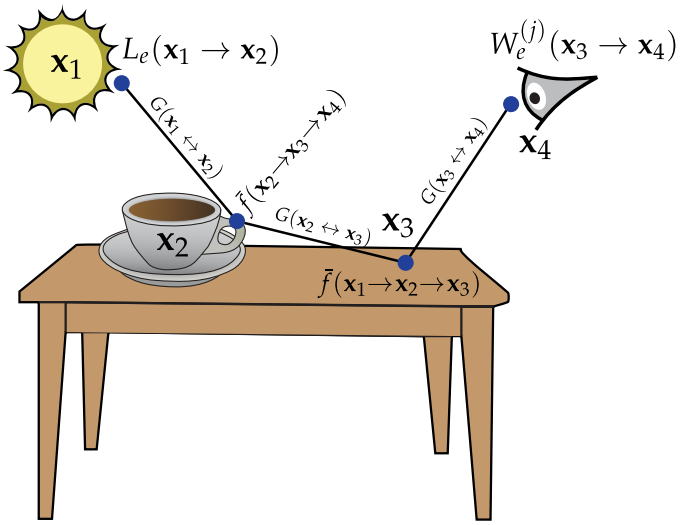


Figure 3.4: Illustration of the different components of the path-space contribution function f_j for a path with four vertices.

The reversed order of arguments to the importance function W_e is simply due to the convention that "$\to$" indicates the flow of light.

Based on this definition, there is a different f_j for each path length k that expresses its throughput as a function of the transport along edges and scattering at vertices. Figure 3.4 shows an example of its structure for a four-vertex path.

To summarize: this chapter derived an explicit expression for the value of a measurement as an integral over paths of different lengths and configurations. This provides the freedom that is needed to build more flexible sampling algorithms that directly operate on the space of paths involving both surface and volume scattering events. We use this path space as the foundation of extended implementations of Bidirectional Path Tracing, Metropolis Light Transport, and Energy Redistribution Path Tracing, as well as the Manifold Exploration technique proposed in this dissertation.

CHAPTER 4

PATH SPACE MANIFOLDS

The Fermat principle [19] states that light travels along paths having a stationary optical length with respect to small variations of the path. Consider the following simple two-dimensional scene, where a sensor measures the amount of light arriving from a light source viewed through a mirror:

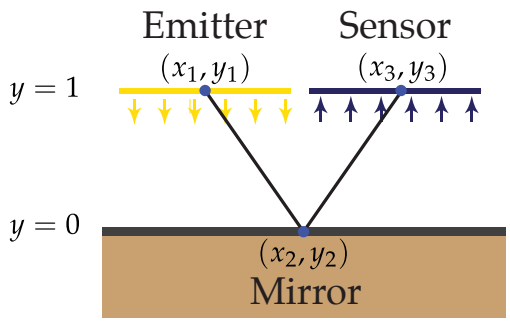


Figure 4.1: A simple flatland scene involving specular paths, with one example path highlighted.

Assuming that the vertical coordinates are fixed at $y_1 = y_3 = 1$ and $y_2 = 0$, the optical length as a function of the free coordinates is given as

$$\mathcal{L}(x_1, x_2, x_3) = \sqrt{1 + (x_1 - x_2)^2} + \sqrt{1 + (x_3 - x_2)^2},$$

which has the derivative

$$\mathcal{L}'(x_1, x_2, x_3) = \frac{x_2 - x_1}{\sqrt{1 + (x_1 - x_2)^2}} + \frac{x_2 - x_3}{\sqrt{1 + (x_3 - x_2)^2}}.$$

Solving for an extremum by setting $\mathcal{L}'$ to zero yields the relation

$$\mathcal{L}'(x_1, x_2, x_3) = 0 \;\Leftrightarrow\; x_2 = \frac{x_1 + x_3}{2}$$

This means that for a light-carrying paths, the position of x_2 is completely determined by x_1 and x_3[1]. The set of such paths thus has a lower dimension,

[1] Or, alternatively, the position of any vertex is given by the positions of the two other vertices.

and it is embedded in the ambient path space. We refer to this as the *specular manifold*—in this case it is simply a plane:

$$\mathcal{S} = \{(x_1, x_2, x_3) \in \mathcal{P} \mid 2x_2 = x_1 + x_3\}$$

In this section, we derive the geometry of such manifolds in a general setting and show how it is connected to the underlying light transport problem.

4.1 Prior work involving specular reflection geometry

Separate from work on global illumination algorithms, various research has examined the properties of specular reflection paths. Mitchell and Hanrahan [42] devised a method to compute irradiance from implicitly defined reflectors, using Fermat's principle with interval Newton's method to locate all reflection paths from a source to a point, and wavefront tracing methods from classical optics to compute the irradiance arriving along these paths. Walter et al. [75] proposed a related method that computes the singly scattered radiance within a refractive object with triangle mesh boundaries. Like these works, our method searches for specular paths. But because it does so within the *neighborhood* of a given path, it avoids the complexities and constraints entailed by a full global search. Another difference is that our manifold formalism can be used to build a fully general rendering system that is not limited to the specular paths that prompted its design.

The widely used method of tracing ray differentials to aid in filtering surface textures for antialiasing [23] also involves reasoning about the local structure of a set of reflected paths—in this case, paths from the eye. Igehy's approach requires elementary local differential information only, in the form of derivatives of surface normals, and does not require global surface descriptions as Mitchell

and Hanrahan's method does. Manifold exploration requires the same local geometric information as Igehy's approach, and can thus be implemented in most modern ray tracing systems.

The analysis of reflection geometry presented by Chen and Arvo [4, 5] is closest to the mathematics underlying our proposed methods. Their work relies on a characterization of specular paths via Fermat's Principle. Using Lagrange multipliers, the authors derive a *path Jacobian* and *path Hessian* with respect to perturbations of the endpoint of a path and use it to accelerate the interactive display of reflections on curved surfaces. The path Jacobian is related to the derivatives that we propose to use to define tangent spaces to the specular manifold while solving for path transitions. However, the use of this derivative and the goals of the research are entirely different: in their case, estimating changes to viewing paths, and in our case, tracking the evolution of specular paths in a very general context, as part of an unbiased rendering system.

The Fermat principle is a deep variational statement about the fundamental properties of light. Unfortunately, computations with it tend to become burdensome when dealing with longer paths that involve multiple specular interactions. While they appear very different in character, local[2] statements like Snell's law of refraction and the law of specular reflection can be shown to be equivalent to the Fermat principle. Thus, although the Fermat principle was used to motivate this chapter, we will from now on use a special half-vector formulation of these local laws, which considerably simplifies the derivation.

A less related but relevant idea is integrating over continuous paths in volume rendering applications, known as the path integral formulation of radiative transfer [62, 52]. This work, with its implications for the concentration of transport in path space, suggests the possibility of using MCMC to integrate

[2]These are called local, since they specify the behavior of light at the individual surfaces.

over multiple-scattering paths in volumes, as discussed in Chapter 7.

Manifold exploration is a technique for integrating the contributions of sets of specular or near-specular illumination paths to the rendered image of a scene. The general approach applies to surfaces and volumes and to ideal and non-ideal (glossy) specular surfaces. In this chapter we begin by examining the manifold defined by ideal specular reflection or refraction, in the setting of surfaces without participating media. In the following section we develop this theory into a rendering method for scenes combining ideal specular surfaces with fairly diffuse surfaces. We will then go on to generalize the method to glossy surfaces and finally extend it encompass participating media with both isotropic and highly directional scattering.

4.2 Motivating examples

As we have seen, the path space formulation of light transport provides a flexible foundation for the development of rendering algorithms. However, in the presence of ideal specular reflection, some difficulties arise, which are normally sidestepped in the transition from theory to algorithm, but which we prefer to confront directly. When some surface interactions are specular, as we saw in the simple example at the start of this chapter, the entire contribution to the path space integral is from paths that obey specular reflection or refraction geometry, and the set of such paths is lower in dimension than the full path space. For instance, consider a family of paths of the form LDSDE (in Heckbert's [20] notation) with one specular reflection vertex. These paths belong to the $\mathcal{P}_5$ component of $\mathcal{P}$, but the paths that contribute all have the property

$$\left(\overrightarrow{\mathbf{x}_3\mathbf{x}_2} + \overrightarrow{\mathbf{x}_3\mathbf{x}_4}\right) \parallel N(\mathbf{x}_3),$$

that is, the half-vector at $\mathbf{x}_3$ is in the direction of the normal. This places two constraints on the path, meaning that all contributing paths lie on a manifold $\mathcal{S}$ of dimension 8 embedded in $\mathcal{P}_5$, which is of dimension 10. The integral is more naturally expressed as an integral over the manifold $\mathcal{S}$, rather than as a singular integral over the whole path space.

To compute illumination due to the specular paths, we use a local parameterization of the manifold in terms of the positions of all nonspecular vertices on the path:

$$\iiiint_{\mathcal{S}} f(\mathbf{x}_1 \dots \mathbf{x}_5)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_4)\, \mathrm{d}A(d\mathbf{x}_5)$$

Note the missing integral over $\mathbf{x}_3$, the specular vertex. The contribution function f still has the same form, a product of terms corresponding to vertices and edges of the path, but the BSDF value at the specular vertex is replaced by a (unitless) specular reflectance value, and the geometry factors for the two edges involving the specular vertex are replaced by a single generalized geometry factor that we will denote $G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3 \leftrightarrow \mathbf{x}_4)$, i.e.:

$$\begin{aligned} = \iiiint_{\mathcal{S}} & L_e(\mathbf{x}_1 \to \mathbf{x}_2)\, G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) f(\mathbf{x}_1 \to \mathbf{x}_2 \to \mathbf{x}_3)\, \underline{G(\mathbf{x}_2 \leftrightarrow \mathbf{x}_3 \leftrightarrow \mathbf{x}_4)\, R} \\ & f(\mathbf{x}_3 \to \mathbf{x}_4 \to \mathbf{x}_5)\, G(\mathbf{x}_4 \leftrightarrow \mathbf{x}_5)\, W_e(\mathbf{x}_4 \to \mathbf{x}_5)\, \mathrm{d}A(\mathbf{x}_1)\, \mathrm{d}A(\mathbf{x}_2)\, \mathrm{d}A(\mathbf{x}_4)\, \mathrm{d}A(d\mathbf{x}_5). \end{aligned}$$

To find the right form of this term, recall that the standard geometry factor for a non-specular edge (Section 3.3) was a change of variables factor describing the derivative of projected solid angle at one vertex with respect to area at the other vertex. The generalized geometry factor is defined analogously: the derivative of solid angle at one end of the specular chain with respect to area at the other end of the chain, considering the path as a function of the positions of the endpoints. Figure 4.2 illustrates this for a more complex path involving a chain of three specular vertices. We will explain below how G can be easily

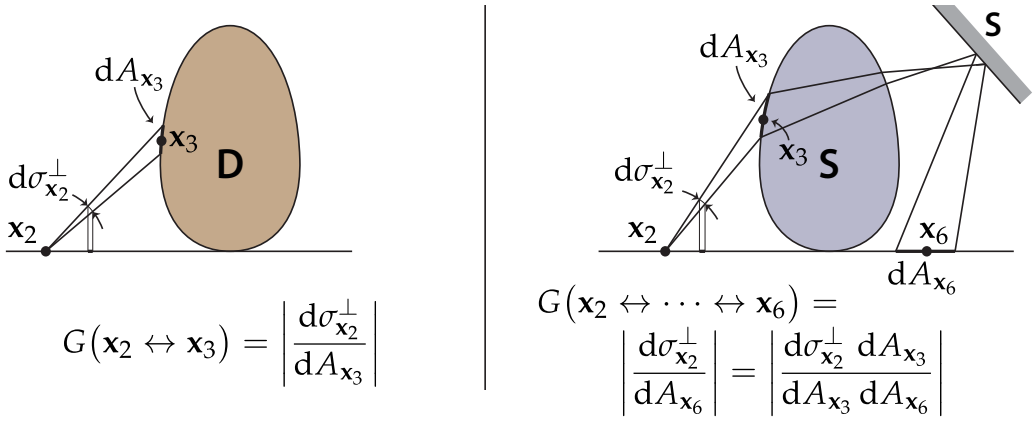


Figure 4.2: The geometry factor (left) and the generalized geometry factor (right) are both derivatives of projected solid angle at one end with respect to area at the far end.

computed from the differential geometry of the specular manifold, which is related to the differential geometry of the surfaces along the path.

4.3 Specular manifold geometry

In the general case, each path of length k belongs to a class in $\{D, S\}^k$ based on the classification of each of its vertices. (In this scheme point or orthographic cameras, and point or parallel lights, are denoted S, while finite-aperture cameras and area lights are D.) Each S surface vertex has an associated constraint that involves its position and the position of the preceding and following vertices:

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = \mathbf{0}. \tag{4.1}$$

The constraint function computes a half-vector at vertex i and projects it into the tangent space; the resulting 2-vector is zero when the half-vector is parallel to the normal. By making use of the generalized half-vector of Runge and Sommerfeld [59], both reflection and refraction can be handled by a single

constraint function[3]:

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = T(\mathbf{x}_i)^T h(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}}), \tag{4.2}$$

$$h(\mathbf{x}, \omega, \omega') = \frac{\eta(\mathbf{x}, \omega)\omega + \eta(\mathbf{x}, \omega')\omega'}{\|\eta(\mathbf{x}, \omega)\omega + \eta(\mathbf{x}, \omega')\omega'\|} \tag{4.3}$$

where $T(\mathbf{x})$ is a matrix whose columns form a basis for the shading tangent plane at $\mathbf{x}$, and $\eta(\mathbf{x}, \omega)$ denotes the refractive index associated with the ray $(\mathbf{x}, \omega)$. This generalized geometry factor is related to the "extended form factor" discussed by Sillion and Puech [58]. Note that the normalization factor in the denominator of (4.3) may appear superfluous, as the constraint equation (4.1) has the constant zero on its right hand side. The reasons for preserving this denominator will become clear at a later stage in Section 6.2, when glossy materials are considered.

"Specular" endpoint vertices also introduce constraints that additionally depend on their type. For instance, when the endpoint is associated with a directional light source (or an orthographic camera), the outgoing direction $\mathbf{x}_1 \to \mathbf{x}_2$ must remain fixed. To avoid creating special cases that would complicate a computer implementation, we found it easiest to introduce such a constraint similarly to (4.2) by setting

$$\mathbf{c}_1(\mathbf{x}_1, \mathbf{x}_2) = T(\mathbf{x}_1)^T \overrightarrow{\mathbf{x}_1\mathbf{x}_2} \quad (= 0)$$

where the tangent space $T(\mathbf{x}_1)$ contains two arbitrary linearly independent vectors that are perpendicular to the direction of the light source or camera. When $\mathbf{x}_1$ is a vertex on a point emitter (or on the aperture of a pinhole camera), the constraint only involves the position of the endpoint (i.e. $\mathbf{x}_1 = \text{const}$).

To make our constraints easier to formulate, we implicitly identify each vertex $\mathbf{x}_i$ with an associated point in $\mathbb{R}^2$ using local parameterizations of $\mathcal{M}$.

[3] This constraint is a computationally convenient way of simultaneously stating the law of reflection and Snell's law. It was introduced to graphics by Walter et al. [74].

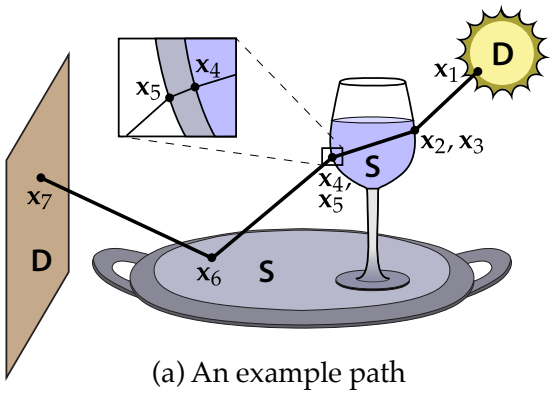
These parameterizations may be defined on arbitrarily small neighborhoods, since only their derivatives are relevant to what follows.

We focus on a path $\mathbf{x}_1, \ldots, \mathbf{x}_n$ with n vertices, of which p are specular. In this case, the constraints can be stacked together into a function $C : \mathbb{R}^{2n} \to \mathbb{R}^{2p}$ parameterized by a pair of local coordinates at each vertex, and the specular manifold is simply the set

$$\mathcal{S} = \{\bar{\mathbf{x}} \mid C(\bar{\mathbf{x}}) = \mathbf{0}\} \tag{4.4}$$

Expressing $\mathcal{S}$ using a constraint in this way makes it convenient to work with neighborhoods of a particular path. The Implicit Function Theorem [60] guarantees the existence of a parameterization of the manifold, in the neighborhood of any path $\bar{\mathbf{x}}$ that is nonsingular (in the sense explained below). This parameterization is a function $q : \mathbb{R}^{2(n-p)} \to \mathbb{R}^{2p}$ that determines the positions of all the specular vertices from the positions of all the nonspecular vertices. Furthermore, the derivative of q, which gives us the tangent space to the manifold at $\bar{\mathbf{x}}$, is simple to compute from the derivative of C.

For the specifics we restrict ourselves to the case of a single chain of specular vertices with non-specular vertices (surfaces, cameras, or light sources) at the ends. Paths with specular endpoints are handled with simple variations of this scheme. This suffices to cover all cases by considering multiple chains along the path separately. Number the vertices in the chain $\mathbf{x}_1, \ldots, \mathbf{x}_k$, with $\mathbf{x}_1$ and $\mathbf{x}_k$ being the (non-specular) endpoints of the segment and the remaining $k-2$ vertices being specular. In this case $C : \mathbb{R}^{2k} \to \mathbb{R}^{2(k-2)}$, and the derivative $\nabla C = \left(\frac{\partial C_i}{\partial x_j}\right)_{ij}$ is a matrix containing $k-2$ by k blocks, each of size 2 by 2. Each block contains the partial derivatives of one of the constraint functions with respect to the coordinates of one of the local surface parameterizations. Each manifold constraint (4.1) depends on three vertices, and as a result, this matrix

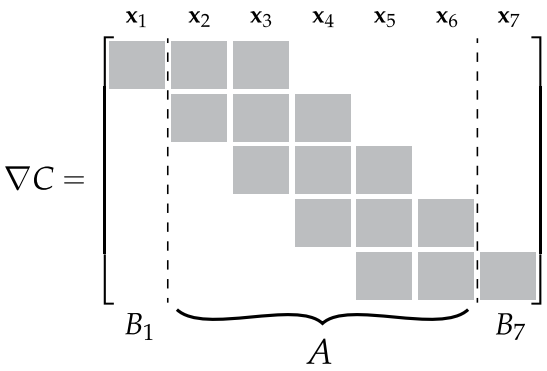


(a) An example path

$$C(\bar{\mathbf{x}}) = 0 \quad \text{where}$$

$$C = \begin{bmatrix} \mathbf{c}_2(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) \\ \vdots \\ \mathbf{c}_6(\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7) \end{bmatrix}$$

(b) Associated constraints



(c) Constraint Jacobian

$$\frac{\partial \begin{bmatrix} \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_6 \end{bmatrix}}{\partial \mathbf{x}_7} = -A^{-1}B_7$$

$$\frac{\partial \begin{bmatrix} \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_6 \end{bmatrix}}{\partial \mathbf{x}_1} = -A^{-1}B_1$$

(d) Tangent space

Figure 4.3: The linear system used to compute the tangent space to the specular manifold, also known as the derivative of a specular chain with respect to its endpoints.

has a block tridiagonal structure (Figure 4.3).

The Implicit Function Theorem gives us a parameterization of the manifold in terms of any two vertices, and if we pick $\mathbf{x}_1$ and $\mathbf{x}_k$ this simply says that the path, in a neighborhood of the current path[4], is a function of the two endpoints. Furthermore, it also tells us the derivative of that parameterization, which is to say, the derivative of all the specular vertices' positions with respect to the positions of the endpoints. If we partition the derivative ∇C, as shown in

[4]Because it is possible to have several separated specular paths joining two points, the parameterization cannot be global.

Figure 4.3 (c), into 2-column matrices B_1 and B_k for the first and last vertices and a square matrix A for the specular chain, then the tangent space to the manifold is

$$T_{\mathcal{S}}(\bar{\mathbf{x}}) = -A^{-1}\begin{bmatrix} B_1 & B_k \end{bmatrix}.$$

This matrix is $k-2$ by 2 blocks in size, and each block gives the derivative of one vertex (in terms of its own tangent frame) with respect to one endpoint.

In general, the specular chain $\mathbf{x}_1 \ldots, \mathbf{x}_k$ will not be globally unique in the sense that there may be other valid configurations having $\mathbf{x}_1$ and $\mathbf{x}_k$ as endpoints. The methods discussed in this thesis take special precautions to deal with this type of non-uniqueness.

In some cases, the chain may not even be locally unique, which occurs when one endpoint is on a caustic due to light emitted from the other endpoint. In this case, $\mathbf{x}_k$ receives infinite power per unit area, and the matrix A ceases to be invertible. This singular case will be discussed later.

We use $T_{\mathcal{S}}(\bar{\mathbf{x}})$ for two things: to navigate on the manifold and to compute the generalized geometry factor. The right two or left two columns of $T_{\mathcal{S}}(\bar{\mathbf{x}})$ are useful for updating the specular chain with $\mathbf{x}_1$ or $\mathbf{x}_k$ held fixed, respectively, and a Newton-like iteration using this derivative forms the basis of the algorithm discussed in the next section.

At the beginning of Section 4.2, we introduced a generalized geometry factor that became necessary when integrating over specular paths. The top-right or bottom-left block of $T_{\mathcal{S}}(\bar{\mathbf{x}})$ can be used to compute this factor as follows. Assuming orthonormal parameterizations[5], the determinant of the top-right block gives the ratio of an infinitesimal area at $\mathbf{x}_k$ to its reflection/refraction, as observed from $\mathbf{x}_1$, measured on the surface at $\mathbf{x}_2$. To convert this to a ratio

[5]If the parameterizations are not orthonormal, two additional determinants are required to account for the change in area.

of area at $\mathbf{x}_k$ to solid angle at $\mathbf{x}_1$, we multiply this determinant by the ordinary geometry factor $G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2)$; this product is the generalized geometry factor $G(\mathbf{x}_1 \leftrightarrow \cdots \leftrightarrow \mathbf{x}_k)$.

$$\begin{aligned} G(\mathbf{x}_1 \leftrightarrow \cdots \leftrightarrow \mathbf{x}_k) &= \left| P_2 A^{-1} B_k \right| G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) \\ &= \left| P_{k-1} A^{-1} B_1 \right| G(\mathbf{x}_{k-1} \leftrightarrow \mathbf{x}_k), \end{aligned} \tag{4.5}$$

where P_i is a 2 by $2(k-2)$ matrix that projects onto the two dimensions associated with vertex i. A simple example computation is shown below.

A useful property of this framework is its reliance on local information that is easily provided in ray tracing-based rendering systems. To compute the blocks of the A and B matrices, we must have access to the partial derivatives of position and shading normal with respect to any convenient parameterization of the surfaces, along with the refractive indices of all objects. These are exactly the same quantities also needed to trace ray differentials through refractive boundaries, which is part of many mature ray tracing-based rendering systems. A consequence of the simple form of the constraint (4.4) is that our technique works with any object that can provide such local information, including implicitly defined shapes or triangle meshes with shading normals.

When the shape associated with a vertex $\mathbf{x}_i$ uses shading normals that are distinct from its geometric normals, we define a shading tangent space $\tilde{T}(\mathbf{x}_i)$ that is found from $T(\mathbf{x}_i)$ using Gram-Schmidt orthogonalization with respect to the shading normal. This new tangent space, and its derivatives with respect to the parameterization, are then used in Equation (4.2) and ∇C.

A simple example: We now demonstrate how to compute the tangent space and generalized geometric term associated with a simple example path shown in Figure 4.4. This is a path with three vertices; $\mathbf{x}_1$ and $\mathbf{x}_3$ are planar endpoints

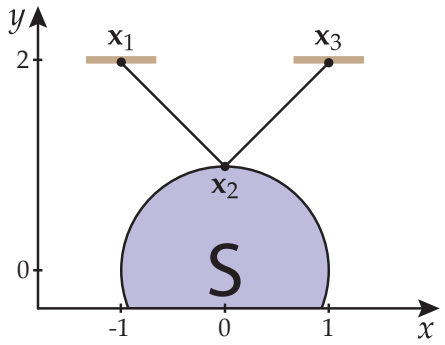


Figure 4.4: An example path with three vertices

used to parameterize the manifold, and $\mathbf{x}_2$ lies on a specularly reflecting cylinder that stretches out along the z-axis.

Assuming that this path is encountered in a ray tracer similar to PBRT [49] or Mitsuba [25], the rendering system will associate an intersection data record with each vertex containing (amongst other things) the position and surface normal, as well as derivatives thereof along some parameterization. Suppose that the information provided is as follows:

$$\begin{aligned}
\mathbf{x}_1 &= \big(-1,2,0\big), & \partial_u \mathbf{x}_1 &= \big(-1,0,0\big), & \partial_v \mathbf{x}_1 &= \big(0,0,1\big),\\
\mathbf{n}_1 &= \big(0,-1,0\big), & \partial_u \mathbf{n}_1 &= \big(0,0,0\big), & \partial_v \mathbf{n}_1 &= \big(0,0,0\big),\\
\mathbf{x}_2 &= \big(0,1,0\big), & \partial_u \mathbf{x}_2 &= \big(1,0,0\big), & \partial_v \mathbf{x}_2 &= \big(0,0,1\big),\\
\mathbf{n}_2 &= \big(0,1,0\big), & \partial_u \mathbf{n}_2 &= \big(1,0,0\big), & \partial_v \mathbf{n}_2 &= \big(0,0,0\big),\\
\mathbf{x}_3 &= \big(1,2,0\big), & \partial_u \mathbf{x}_3 &= \big(1,0,0\big), & \partial_v \mathbf{x}_3 &= \big(0,0,1\big),\\
\mathbf{n}_3 &= \big(0,-1,0\big), & \partial_u \mathbf{n}_3 &= \big(0,0,0\big), & \partial_v \mathbf{n}_3 &= \big(0,0,0\big)
\end{aligned}$$

Note that the parameterizations above are locally orthonormal—this is generally preferable, since it simplifies many computations involving ∇C. In practice, we can simply apply a suitable linear transformation to the tangents and normal derivatives to make them correspond to a locally orthonormal chart. Let us now compute the entries of ∇C.

Recall that the constraint associated with vertex $\mathbf{x}_2$ is

$$C(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_2) = T(\mathbf{x}_2)^T \left(\frac{\mathbf{x}_1 - \mathbf{x}_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|} + \frac{\mathbf{x}_3 - \mathbf{x}_2}{\|\mathbf{x}_3 - \mathbf{x}_2\|} \right) \Big/ \left\| \frac{\mathbf{x}_1 - \mathbf{x}_2}{\|\mathbf{x}_1 - \mathbf{x}_2\|} + \frac{\mathbf{x}_3 - \mathbf{x}_2}{\|\mathbf{x}_3 - \mathbf{x}_2\|} \right\|$$

As mentioned earlier, the normalization is technically unnecessary to define the manifold, but we include it for reasons that will be seen later. To compute derivatives of this expression, we can parameterize all needed ingredients to first order using the above data records, i.e.

$$\mathbf{x}_i(u_i, v_i) = \mathbf{x}_i + u_i(\partial_u \mathbf{x}_i) + v_i(\partial_v \mathbf{x}_i), \quad \mathbf{n}_i(u_i, v_i) = \mathbf{n}_i + u_i(\partial_u \mathbf{n}_i) + v_i(\partial_v \mathbf{n}_i)$$

$$T(u_2, v_2) = \begin{pmatrix} \partial_u \mathbf{x}_2 - \langle \partial_u \mathbf{x}_2, \mathbf{n}_2(u_2, v_2) \rangle \mathbf{n}_2(u_2, v_2) \\ \partial_v \mathbf{x}_2 - \langle \partial_v \mathbf{x}_2, \mathbf{n}_2(u_2, v_2) \rangle \mathbf{n}_2(u_2, v_2) \end{pmatrix}$$

Here, $u_i = v_i = 0$ corresponds to the current path. The rest is just a big nested application of the product rule. When differentiating C, we get two terms: one which accounts for changes of the tangent frame at $\mathbf{x}_2$, with h (Equation 4.3) held constant, and one which accounts for changes of h, while the tangent frame is held constant. For the former, we require derivatives of T, e.g:

$$\frac{\partial T}{\partial u_2} = \begin{pmatrix} -\langle \partial_u \mathbf{x}_2, \partial_u \mathbf{n}_2 \rangle \mathbf{n}_2 - \langle \partial_u \mathbf{x}_2, \mathbf{n}_2 \rangle \partial_u \mathbf{n}_2 \\ -\langle \partial_v \mathbf{x}_2, \partial_u \mathbf{n}_2 \rangle \mathbf{n}_2 - \langle \partial_v \mathbf{x}_2, \mathbf{n}_2 \rangle \partial_u \mathbf{n}_2 \end{pmatrix},$$

$$\frac{\partial T}{\partial v_2} = \begin{pmatrix} -\langle \partial_u \mathbf{x}_2, \partial_v \mathbf{n}_2 \rangle \mathbf{n}_2 - \langle \partial_u \mathbf{x}_2, \mathbf{n}_2 \rangle \partial_v \mathbf{n}_2 \\ -\langle \partial_v \mathbf{x}_2, \partial_v \mathbf{n}_2 \rangle \mathbf{n}_2 - \langle \partial_v \mathbf{x}_2, \mathbf{n}_2 \rangle \partial_v \mathbf{n}_2 \end{pmatrix}$$

and for the second term we can repeatedly apply the following vector calculus identity:

$$\frac{\partial}{\partial t} \frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|} = \frac{1}{\|\mathbf{z}(t)\|} \frac{\partial \mathbf{z}(t)}{\partial t} - \frac{\mathbf{z}(t)}{\|\mathbf{z}(t)\|^3} \left\langle \mathbf{z}(t), \frac{\partial \mathbf{z}(t)}{\partial t} \right\rangle$$

where $\mathbf{z}$ is a vector-valued function that depends on t. The resulting expression is quite messy and thus we only provide numerical values here. In particular,

∇C is given by

$$\nabla C = \begin{bmatrix} -\frac{1}{4} & 0 & -\frac{3}{2} & 0 & \frac{1}{4} & 0 \\ 0 & \frac{1}{2} & 0 & -1 & 0 & \frac{1}{2} \end{bmatrix}$$

and therefore

$$T_{\mathcal{S}}(\bar{\mathbf{x}}) = -A^{-1} \begin{bmatrix} B_1 & B_3 \end{bmatrix} = \begin{pmatrix} -\frac{1}{6} & 0 & \frac{1}{6} & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix}.$$

Since the parameterizations are orthonormal, the geometric term between $\mathbf{x}_1$ and $\mathbf{x}_3$ is simply

$$G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_3) = \left| P_2 A^{-1} B_3 \right| G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \begin{vmatrix} \frac{1}{6} & 0 \\ 0 & \frac{1}{2} \end{vmatrix} G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) = \frac{1}{48}.$$

For a C++ implementation that computes ∇C, please refer to the appendix A2.

CHAPTER 5

WALKING ON THE SPECULAR MANIFOLD

At their core, MCMC rendering techniques require an ability to make transitions between nearby contributing paths. In the previous chapter, we defined the specular manifold and described how to compute the tangent space associated with a given specular path. In this section, we develop a local parameterization of the manifold that makes extensive use of this tangent space information to find neighboring paths. In particular, we propose an algorithm that moves one of the endpoints of a specular chain and moves all the intermediate vertices to a valid new configuration. Later, in Chapter 6, we show how to apply this local parameterization as a key component of a MCMC rendering method.

To simplify the discussion, we will focus on the case where the position of a vertex $\mathbf{x}_n$ of a specular chain $\mathbf{x}_1, \dots \mathbf{x}_n$ is adjusted to a given new position $\mathbf{x}'_n$, while $\mathbf{x}_1$ is held fixed. We shall also briefly introduce the assumption that $\mathbf{x}_n$ is located on a planar surface of infinite extent.

Our manifold walking algorithm is based on two key insights:

1. The A and B matrices (Section 4.3) may be used to map an infinitesimal in-plane movement of $\mathbf{x}_n$ to displacements of the vertices $\mathbf{x}_2, \dots, \mathbf{x}_{n-1}$. We can use these displacements to approximate a finite change to the path simply by adding an offset to each vertex, but this will move the path off the specular manifold.

2. Ray tracing provides a deterministic means of projecting an off-specular path back onto the space of valid configurations. Given $\mathbf{x}_1$ and $\mathbf{x}_2$, we can trace a sequence of rays $\mathbf{x}_i \to \mathbf{x}_{i+1}$, at each step performing a specular reflection or refraction exactly as in normal ray tracing, and this leads to corrected positions $\mathbf{x}_2^+ \dots \mathbf{x}_n^+$.

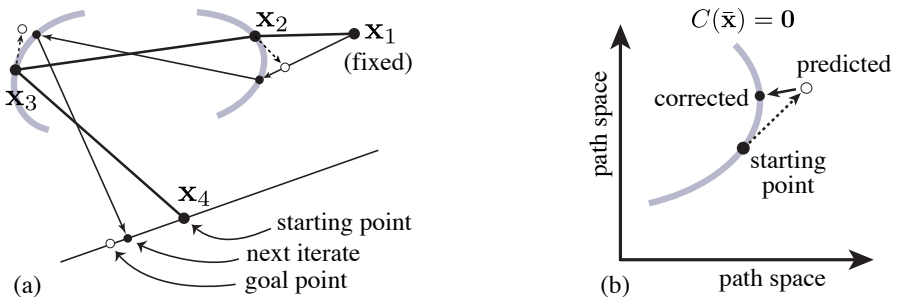


Figure 5.1: One iteration of updating a path using the specular manifold. **(a)** The path vertices are modified according to a local linear model, which **(b)** corresponds to a step along the tangent plane to the manifold, then **(a)** a nearby valid specular path is found, which **(b)** corresponds to projecting back onto the manifold.

By combining 1. and 2., we obtain a predictor-corrector type algorithm (Figure 5.1) that performs a step according to a local linear model, followed by a projection that restores the specular configuration, resulting in a new path $\mathbf{x}_1, \mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$, and these steps are repeated until convergence. As long as the prediction step solves the linear model and moves in the tangent space of the manifold, this iteration behaves like Newton's method, exhibiting quadratic convergence near the solution.

As with all Newton-like iterations, it is not guaranteed to converge when started far from the solution, since the linear model may not be accurate enough to make progress. But since the model is first-order accurate, the algorithm is *guaranteed* to make forward progress when the constraint function is differentiable and the partial steps are small enough. Our algorithm uses a simple heuristic to decrease the step size when progress is not made, then increase back to full steps to get quadratic convergence as it approaches the target configuration. This iteration is illustrated in Figure 5.1 and laid out in the following algorithm:

WalkManifold$(\mathbf{x}_1, \ldots, \mathbf{x}_n \rightsquigarrow \mathbf{x}'_n)$

1 Set $i = 0$ and $\beta = 1$

2 **while** $\|\mathbf{x}_n - \mathbf{x}'_n\| > \varepsilon L$

3 $\quad \mathbf{p} = \mathbf{x}_2 - \beta\, T(\mathbf{x}_2) P_2 A^{-1} B_n T(\mathbf{x}_n)^T (\mathbf{x}'_n - \mathbf{x}_n)$

4 $\quad$ Propagate the ray $\mathbf{x}_1 \rightarrow \mathbf{p}$ through all specular interactions, producing $\mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$.

5 $\quad$ **if** step 4 succeeded and $\|\mathbf{x}_n^+ - \mathbf{x}'_n\| < \|\mathbf{x}_n - \mathbf{x}'_n\|$

6 $\quad\quad \mathbf{x}_2, \ldots, \mathbf{x}_n = \mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$

7 $\quad\quad \beta = \min\{1, 2\beta\}$

8 $\quad$ **else**

9 $\quad\quad \beta = \frac{1}{2}\beta$

10 $\quad$ Set $i = i + 1$, and fail if $i > N$.

11 **return** $\mathbf{x}_2, \ldots, \mathbf{x}_{n-1}$

We now describe each line in more detail:

Line 1: The variable i records the number of iterations until a specified maximum N is reached, and β denotes a step size that is dynamically adjusted. The iteration begins with full-sized steps, i.e. $\beta = 1$.

Line 2: When the distance of vertex $\mathbf{x}_n$ and the target $\mathbf{x}'_n$ is small enough, the iteration stops. Here, ε is a relative error threshold to a scene-scale length L (we use $L = \max_i \|\mathbf{x}_i\|$ and $\varepsilon = 10^{-7}$).

Line 3: This step makes use of the differential geometry of the manifold via the components of the ∇C matrix, which are recomputed at every iteration.

Reading the expression from right to left, the vector from $\mathbf{x}_n$ to $\mathbf{x}'_n$ is first mapped into the tangent space at $\mathbf{x}_n$ using the 2x3 matrix $T(\mathbf{x}_n)^T$; this assumes that the plane parameterization of the vertex $\mathbf{x}_n$ is orthogonal.

Due to the manifold constraint, a displacement of the vertex $\mathbf{x}_n$ is accompanied by a corresponding displacement of all other vertices, which can be found to first order using the matrix $A^{-1}B_n$. For the purposes of this method, we are only interested in changes of the second vertex, which the matrix $T(\mathbf{x}_2)P_2$ extracts as a 3D vector in world space (Section 4.3 contains details on the matrices used).

Finally, the adjusted position of $\mathbf{x}_2$ is computed and stored as $\mathbf{p}$; the magnitude of the adjustment depends on whether full steps or sub-steps are being taken (i.e. whether $\beta = 1$ or $\beta < 1$).

Line 4: Because of the nature of the linear extrapolation in Line 3, The point $\mathbf{p}$ will generally not lie on any of the surfaces in $\mathcal{M}$. In this case we can find a tentative nearby vertex that does by tracing a ray from $\mathbf{x}_1$ to $\mathbf{p}$, i.e. computing $\mathbf{x}_2^+ = \mathbf{x}_{\mathcal{M}}(\mathbf{x}_1, \overrightarrow{\mathbf{x}_1\mathbf{p}})$. Since the second vertex was assumed to be specular, the local scattering law (i.e. the law of specular reflection, or Snell's law) also determines the next vertex, and so on, until a new tentative endpoint $\mathbf{x}_n^+$ is found.

Lines 5-10: If the local linear model is a good approximation of the manifold within the region of interest, the extrapolation and projection steps will succeed, and the adjusted endpoint will lie closer to the target position than the preceding one. In this case, the algorithm accepts the tentative set of vertices $\mathbf{x}_2^+, \ldots, \mathbf{x}_n^+$ and uses them as the starting point of the next iteration. Also, the step size is doubled up to a maximum value of one (corresponding to full steps). In all other cases, the step is retried using progressively smaller step sizes to ensure that the linear model eventually becomes accurate. When the number of iterations exceeds a certain threshold (we use $N = 20$), the iteration fails.

To make this algorithm usable for general specular chains, we must remove the previous assumption that the endpoint $\mathbf{x}_n$ is located on a plane. Inspecting the pseudocode reveals that the actual locations of $\mathbf{x}_n^+$ along the way from $\mathbf{x}_n$ to $\mathbf{x}'_n$ only play a minor role: they are used to check progress and ensure that $\mathbf{x}_n^+ \approx \mathbf{x}'_n$ at convergence. Hence, we can ignore the geometry of the last vertex and act as if it was located on a plane containing both $\mathbf{x}_n$ and $\mathbf{x}'_n$. We construct such a plane and modify Line 4 of WalkManifold so that the last propagation step computes an intersection against this virtual plane, ignoring the actual scene geometry. Once the algorithm converges, we must ensure that $\mathbf{x}_{n-1}$ and $\mathbf{x}_n$ are mutually visible before reporting the manifold walk as successful.

In our implementation, we choose the plane normal using the following symmetric orthogonalization procedure

$$\mathbf{n}_{\text{plane}} := \gamma\big(\gamma(\mathbf{n}+\mathbf{n}') - \overrightarrow{\mathbf{x}_n\mathbf{x}'_n}\langle\gamma(\mathbf{n}+\mathbf{n}'), \overrightarrow{\mathbf{x}_n\mathbf{x}'_n}\rangle\big)$$

where $\mathbf{n}$ and $\mathbf{n}'$ are the surface normals at $\mathbf{x}_n$ and $\mathbf{x}'_n$, and $\gamma(\mathbf{v}) := \mathbf{v}/\|\mathbf{v}\|$. This ensures that the same plane is used for walks $\mathbf{x}_n \rightsquigarrow \mathbf{x}'_n$ and $\mathbf{x}'_n \rightsquigarrow \mathbf{x}_n$.

To compute the matrix A, we derived symbolic expressions for the manifold constraints in C (Equation 4.4). A C++ implementation is given in the appendix. When solving the resulting linear system in step 3, it is beneficial to exploit the special structure of this matrix, which becomes important when processing specular chains with more than about ten vertices. We solve for $\mathbf{p}$ using a block tridiagonal LU factorization, and this reduces the time complexity from $O(n^3)$ to $O(n)$, n being the number of vertices in the chain.

There are several situations in which this algorithm may fail to converge: first, a specular path between $\mathbf{x}_1$ and $\mathbf{x}'_n$ need not exist at all. Secondly, WalkManifold usually cannot find paths that lie on a different connected component of the manifold. Thirdly, when the local structure of the manifold

is complex (e.g. due to high-frequency geometric detail of the reflectors and refractors) and $\|\mathbf{x}_n - \mathbf{x}'_n\|$ is large, the iteration may not converge to a solution.

Finally, the linear system may not be invertible, which happens when the last path lies at the fold of a wavefront, e.g. a caustic receiving infinite power per unit area; this is the locally non-unique case discussed in Chapter 4. Because the associated set of configurations has Lebesgue measure zero, this case only occurs rarely in simulations. During the $\sim 10^{12}$ manifold walks performed to produce the results of this dissertation, we detected $\sim 1.7 \cdot 10^5$ non-invertible linear systems. In the MCMC context it is not a problem for the iteration to fail occasionally, as will be explained in the next section.

The manifold walking algorithm works reliably for large chains with over 10 vertices, especially when it is used by the transition rule discussed in the next section, which only moves the endpoints of chains by a small amount. In our scenes, we observe between 92 and 98% successful walks, taking 2-3 iterations on average to converge to the tolerance $\varepsilon = 10^{-7}$. The failing 2-8% mainly contain cases where WalkManifold failed for good reasons, because it was asked to walk to a point for which there is no valid configuration on the manifold.

We have shown that it is possible to obtain a local parameterization of the neighborhood of a specular path using a simple iterative algorithm that is informed by the differential geometry of the specular manifold. Our approach works in a general setting and finds solutions reliably and efficiently. In the next chapter, we present a new transition rule that proposes steps in path space using manifold walks.

CHAPTER 6

MANIFOLD EXPLORATION FOR SURFACES

Recall that in the context of MCMC, a transition rule, or *perturbation*, is a random process that generates a *proposal* state conditioned on the current state of a Markov chain (Section 2.4.5). It provides the basic means of navigating through the state space, but to do this correctly the rule must satisfy two basic criteria: transitions must be *reversible* (i.e. return to the previous state with nonzero probability density), and the rule must also supply a function that computes the probability of proposals conditioned on the current state up to constant factors that are also shared with proposals in the reverse direction.

As discussed earlier, a transition rule should furthermore propose modifications of an appropriate scale in order to create an efficient sampling procedure. A rule that takes large steps will tend to leave local maxima of the target distribution π, and such steps are rejected with high probability. A rule that takes tiny steps will find most of them accepted, but it will not explore the state space well. Our perturbation is designed so that its scale naturally adapts to the scene, including the geometry and material properties.

The new perturbation supports general scenes and can be used both with the ERPT algorithm by Cline et al. and the path space MLT framework proposed by Veach and Guibas, where it replaces and generalizes the lens, caustic, and multi-chain perturbations. Depending on which combination is used, we call the resulting algorithm either *Manifold Exploration Path Tracing* (MEPT) or *Manifold Exploration Metropolis Light Transport* (MEMLT).

6.1 Manifold perturbation

Given an input path, the manifold perturbation finds a nearby path using a sequence of steps that can be grouped into *sampling* and *connection* phases (Figure 6.1). The sampling phase chooses a subpath to be modified that consists of three non-specular vertices which are potentially separated by specular chains. We shall denote these non-specular vertices as $\mathbf{x}_a, \mathbf{x}_b$, and $\mathbf{x}_c$. After establishing the type of perturbation to be performed, the sampling step generates a perturbed outgoing direction at the vertex $\mathbf{x}_a$ and propagates it through the specular chain between $\mathbf{x}_a$ and $\mathbf{x}_b$ (if any) until arriving at a new non-specular vertex $\mathbf{x}'_b$ in the neighborhood of $\mathbf{x}_b$. Then a manifold walk is used to update the specular chain (if any) between $\mathbf{x}_b$ and $\mathbf{x}_c$. If the configuration of the new path (i.e. the arrangement of specular and nonspecular vertices) is different in any way, the perturbation is rejected immediately.

Consider the setup shown in Figure 6.2: a glass egg focuses the sun onto a diffuse surface, where it forms a caustic that is visible to the camera through a mirror. Starting at the vertex $\mathbf{x}_a$ (the camera, in this example), we can slightly change the angle of the outgoing ray and propagate it through the mirror to obtain a perturbed vertex $\mathbf{x}'_b$ on the diffuse surface. To complete the perturbation, we must somehow connect this new vertex to the light source, but it is not

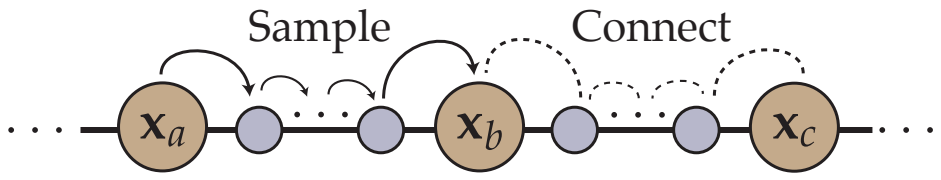


Figure 6.1: The manifold perturbation samples a perturbed outgoing direction from a vertex $\mathbf{x}_a$ and propagates it through a specular chain (if any) using ray tracing until arriving at a non-specular vertex $\mathbf{x}_b$. To connect the vertices $\mathbf{x}_b$ and $\mathbf{x}_c$, the perturbation performs a manifold walk to determine the positions of intermediate specular vertices (if any).

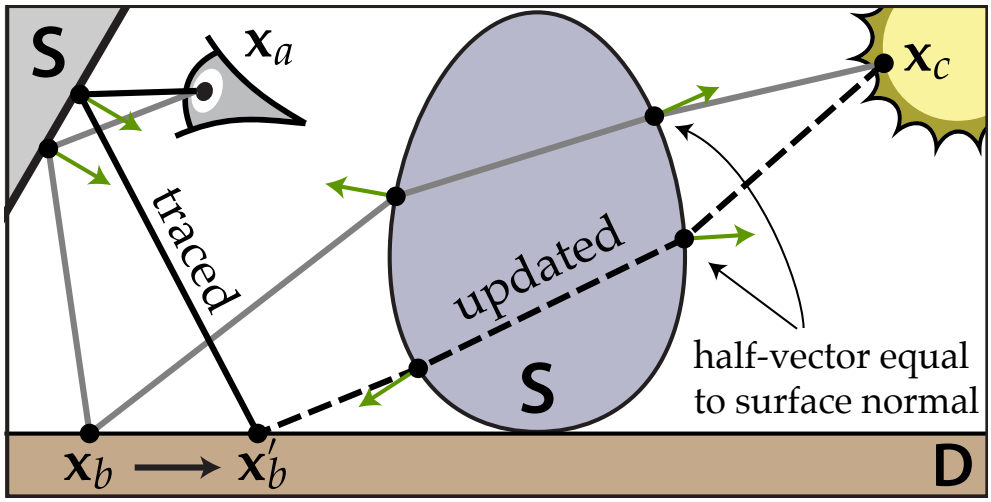


Figure 6.2: Manifold perturbation example: a slightly perturbed outgoing direction at $\mathbf{x}_a$ is propagated until it encounters the non-specular vertex $\mathbf{x}'_b$. Previously, it was not clear how to "connect" $\mathbf{x}'_b$ to $\mathbf{x}_c$ through multiple specular interactions. Our method can find this connection given knowledge about the previous path.

immediately clear how to do this because of the specular chain between the vertices $\mathbf{x}'_b$ and $\mathbf{x}_c$.

Up to this point, the proposed scheme is very similar to the set of perturbations proposed by Veach and Guibas. However, recall that in their work, perturbations must propagate through the path until arriving at a *pair* of adjacent non-specular vertices ("DD" in Heckbert's [20] notation) that can be used to establish a connection edge. Any attempt to connect two sampled subpaths that involves a specular vertex must fail, since the probability of creating a valid path in this manner is zero.

In comparison, our perturbation can stop at the vertex $\mathbf{x}_b$ and use the WalkManifold algorithm to solve for a valid configuration of the specular chain between $\mathbf{x}'_b$ and $\mathbf{x}_c$ (Figure 6.2). This seemingly subtle difference has major repercussions on the types of scenes that can be rendered efficiently. In particular, the resulting method can systematically explore large classes of specular paths instead of having to rely on random sampling alone.

In the following, we discuss the perturbation in more detail; first for the ideal specular case and then in a more general form that extends to rough surfaces.

Strategy sampling: Motivated by the desire to support a large range of different types of path modifications with high probability, the sampling step first chooses among the possible *perturbation strategies* for a given path by selecting three vertices as follows. Given a path $\mathbf{x}_1, \ldots, \mathbf{x}_k$, uniformly select a non-specular initial vertex $\mathbf{x}_a$, as well as a perturbation direction (i.e. towards the light source or towards the camera). Walk along the path in this direction until the first non-specular vertex is encountered, and continue until a second non-specular vertex is found. This path traversal may fail by walking past the end of the path, in which case the strategy sampling phase is simply restarted from scratch. This determines $\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$. For notational convenience, assume that $a < b < c$.

Perturbation sampling: With the overall strategy established, the sampling phase now perturbs the path segment $\mathbf{x}_{a+1}, \ldots, \mathbf{x}_b$. The goal here is to produce a new subpath $\mathbf{x}'_{a+1}, \ldots, \mathbf{x}'_b$ that is "nearby". When the vertex $\mathbf{x}_a$ denotes a surface scattering event with incident and exitant directions $\omega_i = \overrightarrow{\mathbf{x}_a \mathbf{x}_{a-1}}$ and $\omega_o = \overrightarrow{\mathbf{x}_a \mathbf{x}_{a+1}}$, the perturbation determines $\mathbf{x}'_{a+1}$ by tracing a ray in a direction ω'_o that is sampled from a suitable spherical distribution $D(\omega'_o)$ concentrated around ω_o. It is absolutely critical that this distribution generates direction changes of the appropriate scale: for instance, when $\mathbf{x}_a$ is a diffuse material, relatively large perturbations are in order. On the other hand, when $\mathbf{x}_a$ is a glossy material that only reflects into a small cone of directions, large perturbations will almost always be rejected, reducing performance.

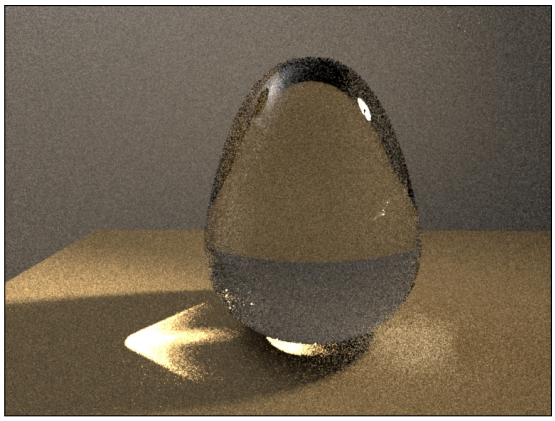
Observe that a useful hint about the right scale can be obtained directly from the scattering model at $\mathbf{x}_a$, in particular from the associated importance sampling

density $p(\omega_i \to \omega_o)$. When this sampling density is high, ω_o is likely located on a sharp peak of the scattering function, and small steps are appropriate.

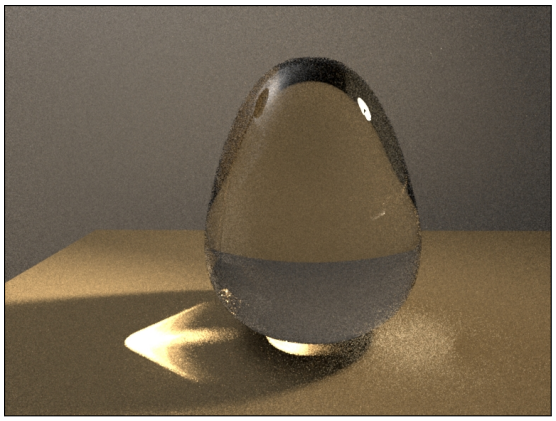
Our strategy is to sample from a distribution $D(\omega_o')$ centered at ω_o whose concentration is set so that $D(\omega_o)$ equals $\lambda^2 p(\omega_i \to \omega_o)$. For $D(\omega_o')$, we use the spherical von Mises-Fisher distribution. Note that this distribution can cause certain numerical difficulties during evaluation and sampling; information on how to avoid them is provided in the Appendix A3.

The parameter λ (generally set between 50 and 500) specifies how large the perturbations are relative to standard BSDF sampling. This is the main parameter of our technique, and it affects how far perturbations will move in path space. When λ is set to an inappropriately low or high value, the amount of noise present in the output renderings increases. In the first case, too few mutations are accepted, causing the chain to become "stuck" in certain paths for many iterations. In the latter case, the steps taken by the chain are too small to effectively explore path space, and this results in the typical coherent noise patterns that are known from other MLT-type algorithms. A comparison involving different settings is shown in Figure 6.3. We currently set this parameter manually to achieve a desired acceptance ratio, but this could in theory be automated using adaptive MCMC [13].

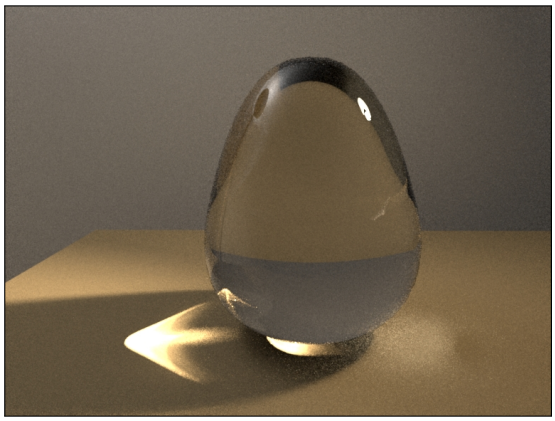
When $\mathbf{x}_a$ is a camera or light source, we choose a new outgoing direction in much the same way, but query the underlying model for the directional density of the associated sampling method (e.g. the density per solid angle corresponding to choosing pixels uniformly in screen-space). To further enlarge the space of possible perturbations, following Veach, we separate the emission and response profile of the camera and light sources into their spatial and directional components so that they can be sampled independently from one



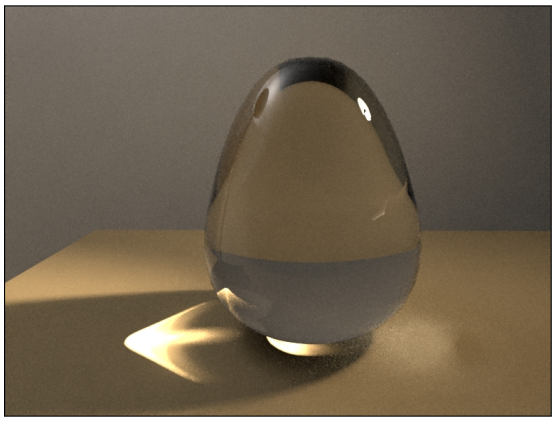
(a) $\lambda = 1$



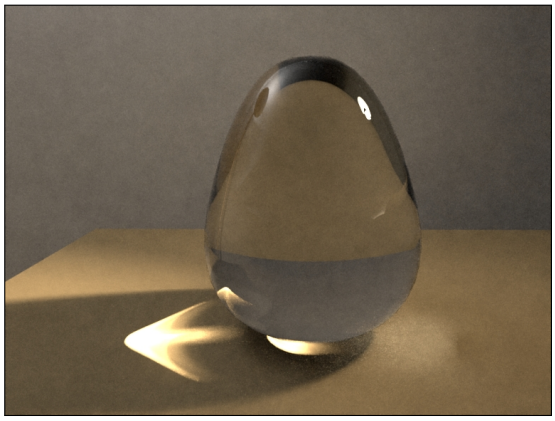
(b) $\lambda = 3$



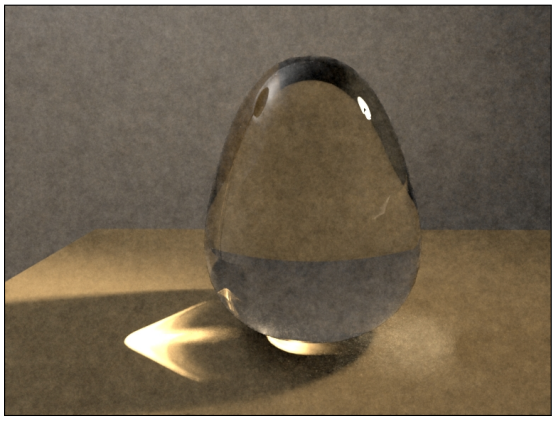
(c) $\lambda = 10$



(d) $\lambda = 30$



(e) $\lambda = 90$



(f) $\lambda = 270$

Figure 6.3: The effects of the λ parameter. This scene was rendered with relatively short Markov Chains (100 steps), hence the range of "good" parameter values is lower than in our other examples scenes. Modeled after a scene by Eric Veach.

another. In our implementation, we found it convenient to realize this approach by representing the camera and light source of a path using two vertices each, where one is a stateless pseudo-vertex and the other represents the position on the light source or camera (see Appendix A4 for details). When $\mathbf{x}_{a+1}$ is such a position vertex, we perturb its location on the aperture or light source by sampling a tangential displacement from a 2D normal distribution with variance $\rho/(2\pi\lambda^2)$, where ρ is the surface area. Being able to perturb both the position and outgoing direction at the path endpoints is valuable when rendering effects such as smooth shadows and out-of-focus blur.

After $\mathbf{x}'_{a+1}$ has been determined in this manner, the perturbation is propagated through the specular chain until reaching $\mathbf{x}'_b$. This process is deterministic.

Connection: When there is no specular chain between $\mathbf{x}'_b$ and $\mathbf{x}_c$, the connection step only entails checking that the vertices are mutually visible, and that their scattering models carry illumination along the connection edge. When there is a chain, we first set

$$\mathbf{x}'_{c-1}, \ldots \mathbf{x}'_{b+1} = \textsc{WalkManifold}(\mathbf{x}_c, \ldots \mathbf{x}_b \to \mathbf{x}'_b)$$

and then perform the same verification.

Recall that a key requirement of the Metropolis-Hastings algorithm discussed in Section 2.4.5 was that a nonzero transition probability $T(\mathbf{x}, \mathbf{x}') > 0$ also implies that $T(\mathbf{x}', \mathbf{x}) > 0$. This creates a potential issue when walking on the manifold, because WalkManifold can be *non-reversible*. It might succeed in moving from $\mathbf{x}$ to $\mathbf{x}'$ but fail to move from $\mathbf{x}'$ to $\mathbf{x}$. Even when the reverse iteration converges, the manifold can contain bifurcations so that it may converge to a different solution. Therefore, we always perform another manifold walk in the reverse direction and reject the perturbation if the path did not return to its

original configuration. In the example scenes, we observed between 0% and 0.4% non-reversible walks.

Transition probability computation Finally, the change in the contribution function is computed and, together with the transition probabilities, used to randomly accept or reject the proposal with probability

$$r(\bar{x}, \bar{x}') = \min \left\{ 1, \frac{f_j(\bar{\mathbf{x}}') T(\bar{\mathbf{x}}', \bar{\mathbf{x}})}{f_j(\bar{\mathbf{x}}) T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')} \right\} \tag{6.1}$$

where f_j is the contribution function (Equation 3.13) and $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$, denote the original and proposal path respectively. Note that many factors cancel in the above ratio, particularly all of those in f_j that are associated with the unchanged path segment, or common terms in the transition probability. For instance, the probability of choosing a particular sampling strategy cancels, since it only depends on the (unchanged) path configuration.

We require that $T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$ and $T(\bar{\mathbf{x}}', \bar{\mathbf{x}})$ express the density of forward and reverse proposals in a common measure so that it is valid to consider ratios of these probabilities. Since the measure used by our integral over specular manifold paths (Section 4.2) is the area product measure of the non-specular vertices, and because in this setting, the only non-specular vertex that changes during a perturbation is $\mathbf{x}'_b$, we must determine the area density at $\mathbf{x}'_b$ that results from the perturbation of ω_o. Observe that sampling an outgoing direction ω'_o from $D(\omega'_o)$ at $\mathbf{x}_a$, and propagating it through the first specular chain, produces area density $D^{\perp}(\omega'_o)\, G(\mathbf{x}_a \leftrightarrow \cdots \leftrightarrow \mathbf{x}'_b)$ on the surface at $\mathbf{x}'_b$ (where $D^{\perp}(\omega'_o) = D(\omega'_o)/|\cos(\mathbf{n}_a, \omega_o)|$ denotes probability with respect to the projected solid angle measure at $\mathbf{x}_a$). This is the needed transition probability $T(\bar{\mathbf{x}}, \bar{\mathbf{x}}')$.

Recall that the previous discussion made use of an importance function $W_e^{(j)}$ that modeled the sensitivity of a pixel j to illumination. However, in a practical

MCMC-based rendering system, we will generally want to sample paths based on their contribution to the *entire* image rather than to a single pixel, and then record their contributions to the individual pixel integrals that are affected. This is accomplished by replacing $W_e^{(j)}$ in (3.13) with an importance function that measures the overall luminance received on the image plane.

6.2 Extension to glossy materials

The method presented thus far can be used for scenes with both specular and non-specular transport, but the two classes are handled in fundamentally different ways. This is unfortunate, since a near-specular chain through an almost-smooth dielectric object would fall under the non-specular classification and hence be treated analogously to a group of diffuse interactions. As a result, such near-specular paths cannot be explored as effectively as perfectly specular ones. However, it turns out that a simple generalization of the specular case suffices to remove this "hard" classification and to encompass glossy materials that fall in between the two extremes.

Consider the motivating example in Figure 6.4: the scene shown on the upper left contains an ideally specular glass egg, and only a single valid light path joins the vertices $\mathbf{x}_b$ and $\mathbf{x}_c$ through it. In the schematic path space view on the right hand side, we can observe that the set of valid paths is a lower-dimensional subset of path space; all energy is concentrated on this manifold. The manifold perturbation can be seen to perform random steps on the zero level set of the function C.

In the glossy case (bottom left), this situation changes: the vertices $\mathbf{x}_b$ and $\mathbf{x}_c$ are now joined by an entire family of paths. Also, the path space integrand corresponding to a chain of glossy interactions has its energy concentrated in

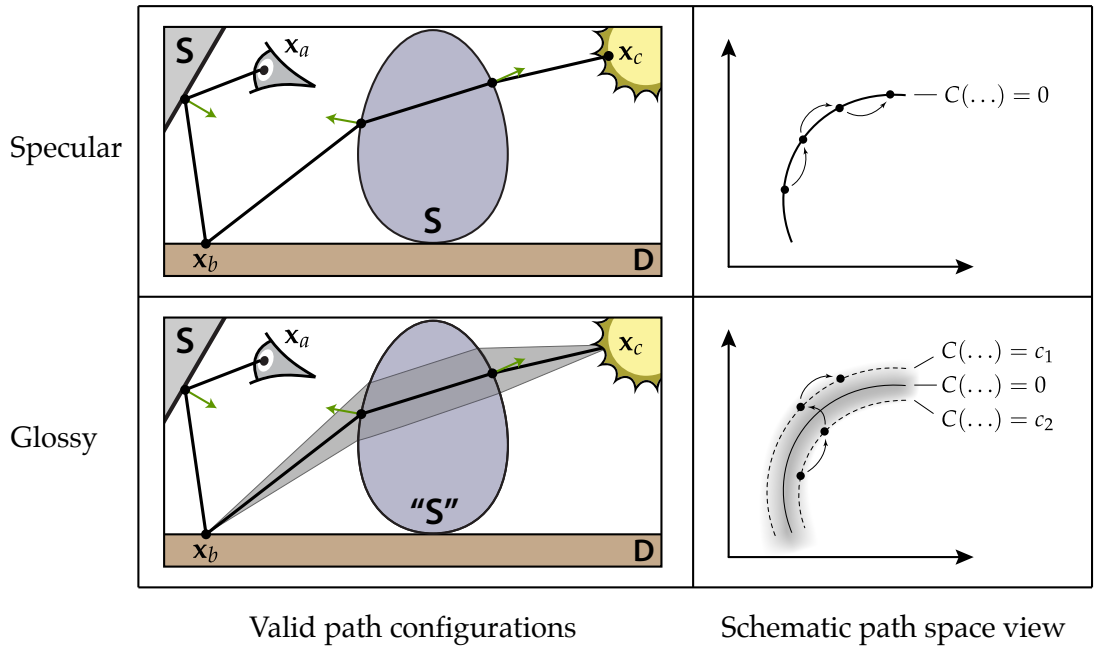


Figure 6.4: Generalizing from the specular to the glossy case

a thin "band" *near* the specular manifold, and the key idea of how we handle glossy materials is to take steps along a family of *offset manifolds* that are parallel to the specular manifold, so that path space near the specular manifold can be explored without stepping out of this thin band of near-specular transport. In this section, we add a simple extension that endows the perturbation with the ability to walk on offset manifolds and to recognize when this is appropriate.

For this, we first replace Equation (4.4) with the *offset manifold*

$$\mathcal{S}_{\mathbf{o}} = \{\bar{\mathbf{x}} \mid C(\bar{\mathbf{x}}) = \mathbf{o}\}\,, \tag{6.2}$$

where $\mathbf{o}$ captures the offset from ideal specular transport. Reinspecting the components of the constraint function C (Equation 4.2)

$$\mathbf{c}_i(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = T(\mathbf{x}_i)^T \underbrace{h(\mathbf{x}_i, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i-1}}, \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}})}_{=:\mathbf{m}_i} \quad (= \mathbf{o}_i)$$

then provides an intuitive explanation for the contents of the vector $\mathbf{o}$: the two entries associated with each vertex $\mathbf{x}_i$ record the x and y coordinates of the

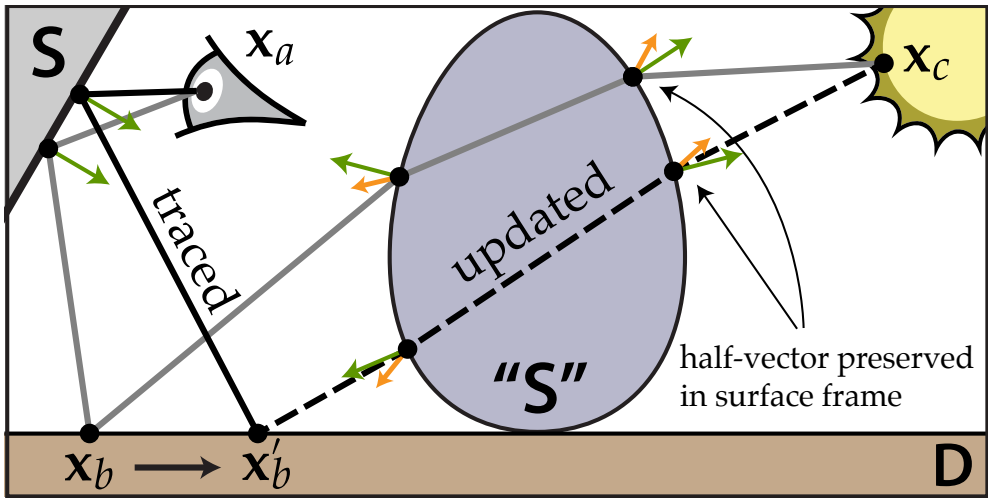


Figure 6.5: Perturbation of a path with near-specular surface interactions: instead of requiring that the half-vectors agree with the surface normals, their direction is preserved in the surface frame.

half-vector (henceforth referred to as $\mathbf{m}_i$) projected into the local tangent frame.

We like to interpret these half-vectors in the context of microfacet theory: recall that microfacet models describe the interaction of light with random surfaces composed of microscopic dielectric or conducting facets that are oriented according to a microfacet distribution. In this case, $\mathbf{m}_i$ (which is now different from the shading normal $\mathbf{n}_i$) is the normal of those microfacets that are responsible for the reflection or refraction along the subpath $\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}$. This finally explains the need for the normalization term in the denominator of h in Equation (4.3): with this term, it is possible to recover the microfacet normal $\mathbf{m}_i \in \mathbb{R}^3$ from its tangential projection stored in the manifold offset constant $\mathbf{o}_i \in \mathbb{R}^2$. Our extended perturbation then preserves the projection of this microgeometry normal $\mathbf{m}_i$ as an invariant during the manifold traversal[1].

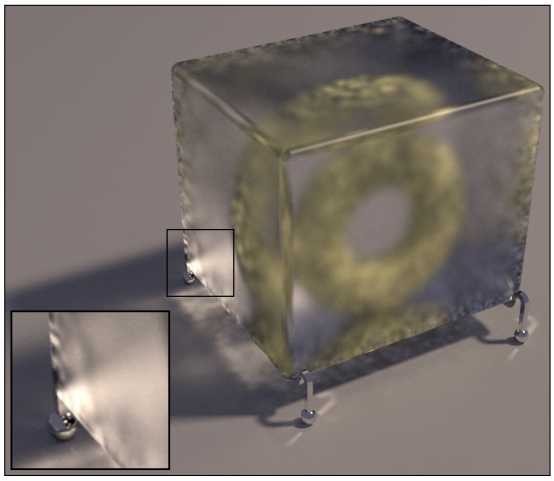
[1]For this to work well, the local surface parameterization should have continuous tangent vector fields within the region that can be reached by one MCMC step. When the tangent vector fields are discontinuous, or when they involve considerable rotation between nearby points, the perturbation takes larger steps that result in a lower acceptance rate.

Changes to the manifold perturbation: Since the differential geometry of the offset manifold is identical to that of the ordinary manifold (involving only different offset constants), the only required change in WalkManifold affects the ray tracing step on line four, where the algorithm now reflects and refracts using $\mathbf{m}_i$ instead of $\mathbf{n}_i$. Similarly, the deterministic phase of the manifold perturbation responsible for propagating the sampled direction at $\mathbf{x}_a$ to a position $\mathbf{x}'_b$ uses these normals instead. Note that it is straightforward to handle both cases, near-specular and specular perturbations, using the same implementation.

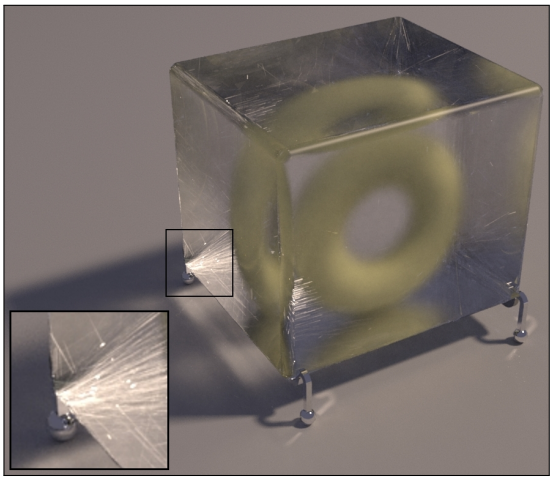
Recognizing near-specular transport: An important issue in the treatment of general scattering is the decision of whether the surface associated with a scattering event is "smooth enough" to be classified as part of a specular chain. We make this decision randomly by assigning a specular probability $\psi(\mathbf{x}_i)$ to each vertex that takes on values 0 and 1 when $\mathbf{x}_i$ is diffuse or specular, respectively, and values in $(0, 1)$ when $\mathbf{x}_i$ is at a rough interface. This avoids the issues of "hard" classifications that are commonly used in rendering algorithms. Figure 6.6 shows a comparison between $\psi(\mathbf{x}_i) = 0$, $\psi(\mathbf{x}_i) = 1$, and the specular probability function proposed in this thesis. For specifics on our choice of $\psi(\mathbf{x}_i)$, please refer to the Appendix A1.

Transition probability: In the purely specular case discussed earlier, the proposal distribution T and target distribution f_j were both supported on the same space $\mathcal{S}$. This permitted computing transition probabilities under an arbitrary projection (e.g. onto the vertex $\mathbf{x}_b$), since the determinant of the associated change of variables canceled when considering ratios of the form f_j/T.

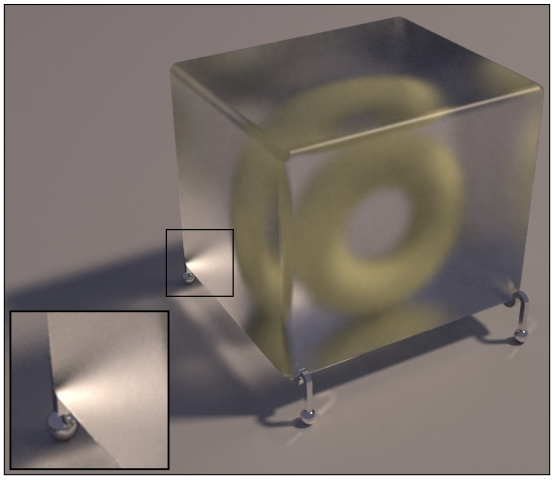
In the glossy case, this does not hold anymore: T is a distribution on an



(a) always non-specular



(b) always specular



(c) probabilistic

Figure 6.6: Consistently classifying glossy materials as non-specular or specular produces unsatisfactory results. Instead, our method makes this decision randomly whenever encountering a rough object (modeled after a scene by Cline et al.)

offset manifold $\mathcal{S}_{\mathbf{o}}$, whereas f_j is defined on the higher-dimensional space $\bigcup_{\mathbf{o}} \mathcal{S}_{\mathbf{o}}$. To compute forward and reverse transition probabilities that remain meaningful when they simultaneously occur in the acceptance ratio we must perform a change of variables that separates out all dimensions that are perpendicular to the current offset manifold (i.e. which remain invariant during a perturbation). We use a reparameterization of T using the parallel variable $\mathbf{x}_b$ (the vertex that

moves) and the perpendicular variables $\mathbf{o}_{i_1}, \ldots, \mathbf{o}_{i_k}$ (the preserved microfacet normals). The indices $i_1, \ldots, i_k \in \{a+1, \ldots, c-1\} \setminus \{b\}$ refer to the glossy vertices that were classified as specular. This change of variables causes a determinant to appear in the final transition probability:

$$T(\bar{\mathbf{x}}, \bar{\mathbf{x}}') = T_{\text{spec}}(\bar{\mathbf{x}}, \bar{\mathbf{x}}') \left| \frac{\partial \left[\mathbf{x}_b, \mathbf{x}_{i_1}, \ldots, \mathbf{x}_{i_k}\right]}{\partial \left[\mathbf{x}_b, \mathbf{o}_{i_1}, \ldots, \mathbf{o}_{i_k}\right]} \right| (1 - \psi(\mathbf{x}_b))(1 - \psi(\mathbf{x}_c))) \prod_{i=a+1, i\neq b}^{c-1} \psi(\mathbf{x}_i). \tag{6.3}$$

T_{spec} refers to the transition probability of the purely specular case. The terms involving ψ represent the discrete probability of the current (random) classification of the path in terms of specular and non-specular vertices. Since the function ψ depends on the roughness of the vertices, which may change during a perturbation, we must account for it here to maintain detailed balance.

The determinant in (6.3) is not hard to compute. Recall that the A-matrix (Section 4.3) maps perturbations of the vertex positions to changes of the half-vectors. Here, we seek the opposite: how all the non-specular vertices move as a function of $\mathbf{o}$ and $\mathbf{x}_b$. We therefore compute the matrix A over the vertex range $\mathbf{x}_{a+1}, \ldots, \mathbf{x}_{c-1}$ and make one small adjustment: the two rows associated with vertex $\mathbf{x}_b$ are set so that there is a 2×2 identity matrix on the diagonal and zeroes elsewhere (see Figure 6.7). We then invert this matrix and discard

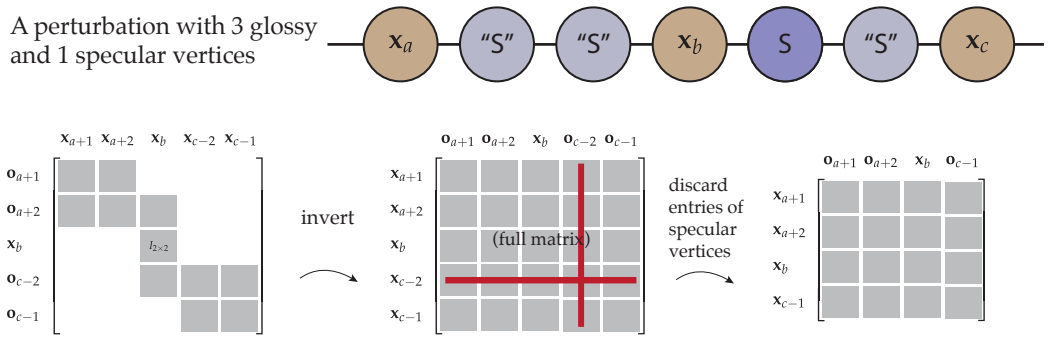


Figure 6.7: An illustration of the matrices involved in the glossy transition probability computation.

all rows and columns in this inverse that are associated with specular vertices; the determinant of the result is the change of variables factor we seek. Note that when the two chains only consists of glossy vertices, it is equivalent (and considerably faster) to compute the inverse of the determinant of the block tridiagonal matrix rather than doing a matrix inversion. When both chains only involve specular vertices, the computation reduces to the case discussed earlier in Section 6.1.

We have discussed a new perturbation rule that is able to explore the neighborhood of paths involving ideally specular and off-specular reflection and transmission in addition to diffuse interactions. Our rule handles the off-specular case by a random classification of material interactions into a diffuse-like and a specular-like case based on the roughness parameter of the underlying reflectance model. This framework is general enough to be extended to further kinds of interactions, and in the next chapter we will show how to apply it to volumetric scattering.

CHAPTER 7

MANIFOLD EXPLORATION FOR VOLUMES

Light transport in volumes tends to involve long multiple scattering processes; for instance, when a photon enters a high-albedo material like a glass of milk or a cloud, hundreds of scattering events may occur before it exits the volume elsewhere. Such diffusion processes cause light to lose its directionality—hence, there are no specular reflections in volumes *per se*.

On the other hand, each of the individual scattering events may be highly directional, which is e.g. the case for many common household materials [43]. From a purely mathematical standpoint, the phase function of a strongly forward-scattering volume is not unlike reflection from a rough mirror—this suggests that we may be able to use manifold exploration to facilitate rendering of such directionally peaked volume interactions as well.

Recall that the fast-varying part of a mirror BRDF is a function of the half-vector, and hence our method preserves it during manifold walks. In the medium case, we are interested in being able to handle highly peaked phase functions that vary rapidly with the scattering angle (i.e. the angle between the incident and outgoing directions). For this purpose we treat the scattering angle as analogous to the half vector, introducing the specular manifold constraint

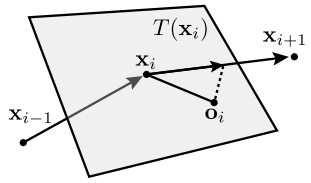


Figure 7.1: Medium constraint

$$\mathbf{c}(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) = T(\overrightarrow{\mathbf{x}_{i-1}\mathbf{x}_i})^T \overrightarrow{\mathbf{x}_i\mathbf{x}_{i+1}} \quad (= \mathbf{o}_i) \tag{7.1}$$

where $T(\mathbf{v})$ is a basis for the plane orthogonal to the direction $\mathbf{v}$ (Figure 7.1). Because medium vertices can move arbitrarily in space, this still leaves one degree of freedom per vertex, which we remove by preserving the distance to

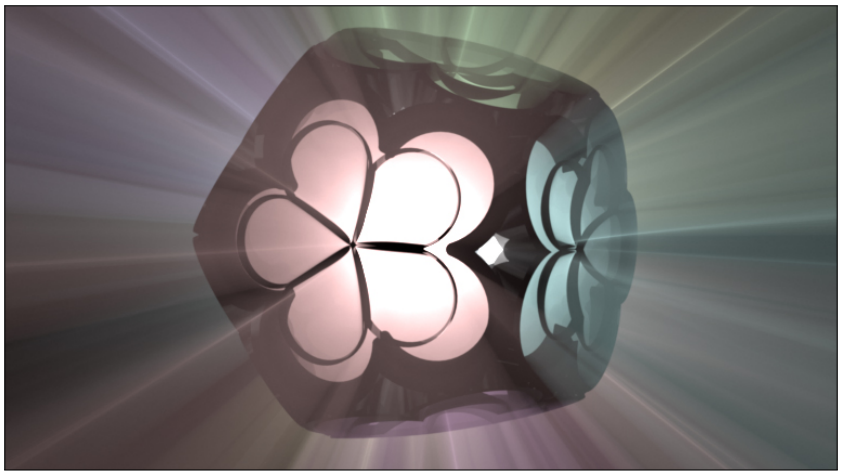


Figure 7.2: It is possible to extend the path space specular integration framework to volumes, for instance to render volume caustics from refractive objects, such as this dodecahedron-shaped metal luminaire with tinted glass inlays.

the scattering event (i.e. $\|\mathbf{x}_{i-1} - \mathbf{x}_i\| = \text{const.}$). While computing the entries of the constraint Jacobian ∇C, we use these two constraints in place of the previous definition (Section 4.3) whenever a vertex describes a medium interaction.

7.1 Medium manifold perturbation

From an algorithmic perspective, manifold exploration for volumes is almost identical to the surface case. Our implementation handles both cases jointly and works with specular chains that contain both surface and medium interactions.

Apart from the new type of constraint (7.1), the computation of offset manifold (6.2) tangent vectors is unchanged. In the ray tracing step 4 of WalkManifold, when encountering a vertex $\mathbf{x}_{i-1}$ that is followed by a medium interaction vertex $\mathbf{x}_i$, we set $\mathbf{x}'_i = \mathbf{x}_{i-1} + \|\mathbf{x}_{i-1} - \mathbf{x}_i\|\,\mathbf{d}$, where $\mathbf{d}$ is the outgoing direction at $\mathbf{x}_{i-1}$ (this enforces the length constraint mentioned earlier). Afterwards, the manifold offset $\mathbf{o}_i$ is transformed into an outgoing direction in the new frame at $\mathbf{x}'_i$ (Figure 7.1).

As in the glossy surface case, we require a criterion that clarifies when treating a medium vertex $\mathbf{x}_i$ as non-specular is in order, and when it is better handled by the manifold. Again, this decision is made probabilistically, based on a modified specular probability function $\psi(\mathbf{x}_i)$ described in the appendix.

CHAPTER 8

RESULTS

We have implemented the proposed technique and prior work as extension modules to the Mitsuba renderer [25]. All techniques operate on top of a newly added *path space abstraction layer* (details in Appendix A4) that exposes cameras, light sources, scattering models, and participating media as generalized path vertex and edge objects with a common basic interface. This greatly simplified the implementation effort, as bidirectional rendering algorithms can usually be stated much more succinctly in terms of operations on vertices that are oblivious to whether they contain, e.g., a camera model or a medium scattering event.

We compare the following algorithms:

- Primary sample space MLT by Kelemen et al., implemented on top of bidirectional path tracing (PSSMLT).
- Path space MLT by Veach and Guibas (MLT).
- An extended form of energy redistribution path tracing by Cline et al. (ERPT), which is seeded by bidirectional rather than unidirectional path tracing. The ERPT implementation shares the caustic, lens, and multi-chain perturbation with the previous algorithm. Since they introduce bias, we did not use the post-processing filters proposed in the original paper.
- Manifold exploration path tracing (MEPT), which is structured similarly to ERPT. We modified the original algorithm by replacing its highly specialized caustic, lens, and multi-chain perturbations with the manifold perturbation. Due to its general design, the new perturbation subsumes and extends the capabilities of the original set.

Scene	Seed generator		Perturb.	Mutations		Manifold walks			Manif. size	
	samples/px.	chains/px.	λ	total	accepted	total	conv.	avg. iter.	avg.	max.
Torus (MEPT)	32	2	100	1178 M	78.3%	1002 M	96.7%	2.3	3.4	7
Chandelier (MEPT)	64	3	160	1216 M	73.6%	975 M	97.6%	2.4	4.6	13
Chandelier (MEMLT)	—	—	160	3626 M	34.1%	376 M	98.4%	2.1	4.5	13
Table (MEPT)	32	1	300	1074 M	77.5%	868 M	95.5%	2.8	4.4	14
Table (MEMLT)	—	—	300	1921 M	35.9%	772.4 M	94.3%	3.2	4.4	14
GlassEgg (MEPT)	128	2	90	1533 M	72.4%	1246 M	92.2%	3.4	4.1	14
GlassEgg (MEMLT)	—	—	90	2774 M	40.3%	1101 M	92.5%	3.3	4.1	14

Table 8.1: Listing of seed generator and perturbation parameters, as well as performance statistics.

- Manifold Metropolis Light Transport (MEMLT), which corresponds to MLT with our perturbation (i.e. the bidirectional mutation and manifold perturbation, but none of the original perturbations from MLT).

Due to the aforementioned abstraction layer, all techniques transparently support participating media even if this was originally not part of their description. We found that MEPT generally performs better that MEMLT due to certain limitations of the bidirectional mutation that are discussed later.

The rendering of result images was conducted in the cloud using Amazon EC2 `cc1.4xlarge` instances, which, at that time, were eight-core Intel Xeon X5570 machines. A single machine was used per image. To exploit the local parallelism, our implementation runs a separate Markov chain on each core, and the resulting buffers are averaged together when exposing the image.

We have rendered three views of a challenging interior scene containing approximately 2 million triangles with shading normals and a mixture of glossy, diffuse, and specular surfaces and some scattering volumes. One hour of processing time was allocated to each rendering technique, and a comparison of the resulting images is shown in Figures 8.1, 8.2, and 8.3. The converged reference images were rendered in 48 hours. The one hour renderings are intentionally unconverged to permit a visual analysis of the convergence behavior.

Table 8.1 lists parameters and statistics collected during these renderings. The path generator columns refer to the seeding scheme used by ERPT and MEPT, which samples and subsequently resamples a number of paths per pixel before launching Markov Chains. The statistics include the total number of mutations and acceptance ratio, as well as the convergence behavior of the manifold walks and vertex count of encountered manifolds.

Chandelier: In this set of results, the poor performance of MLT is most apparent and is caused by the ineffectiveness of the bidirectional mutation in finding long specular paths. Because it must decide up front on the configuration of a path before generating it, most of the time the mutation fails, resulting in acceptance rates under 1%. Consequently, too few jumps between disjoint connected components of path space occur, causing parts of the image to have an incorrect relative brightness. This weakness is inherited by MEMLT, which also builds upon the bidirectional mutation. It is more successful at exploring some of the diffuse-specular-diffuse paths through the bulbs but overall does not work well on this scene. Densely seeding the same perturbations using paths obtained from bidirectional path tracing at each pixel does not suffer from this advantage. The resulting methods perform much better, as can be seen in the ERPT and MEPT renderings.

Table: This scene is lit by the chandelier, with its glass-enclosed sources, so all illumination is by specular paths. By reasoning about the geometry of the specular and offset specular manifolds for the paths it encounters, our perturbation rule is more successful at rendering paths—such as illumination that refracts from the bulbs into the butter dish, then to the camera (6 specular vertices)—that the other methods struggle with. The MLT rendering looks too dark, because it did not find enough of these paths and mainly captures

diffuse illumination from the walls. The noise in the ERPT result reveals that the underlying bidirectional path tracer encountered some of those paths but the Veach–Guibas perturbations are not able to explore path space around them effectively. The primary sample space MLT variant also has difficulties rendering this scene, because it has no knowledge about the underlying path geometry. MEMLT produces a clean result, but the relative brightness of different parts of the image is far from converged due to the bidirectional mutation's difficulty in performing sufficiently many jumps between them.

GlassEgg: In this scene, our technique's ability to create a specular chain containing both medium and surface interactions leads to fast convergence when rendering the homogeneous forward-scattering medium (Henyey-Greenstein phase function, $g = 0.8$) inside the glass egg. MLT and ERPT perform poorly here, since they do not have suitable perturbations for exploring this space. Because the MLT perturbations treat glossy and diffuse materials identically, they have difficulty rendering the near-specular tabletop, producing streak-like artifacts in the output rendering.

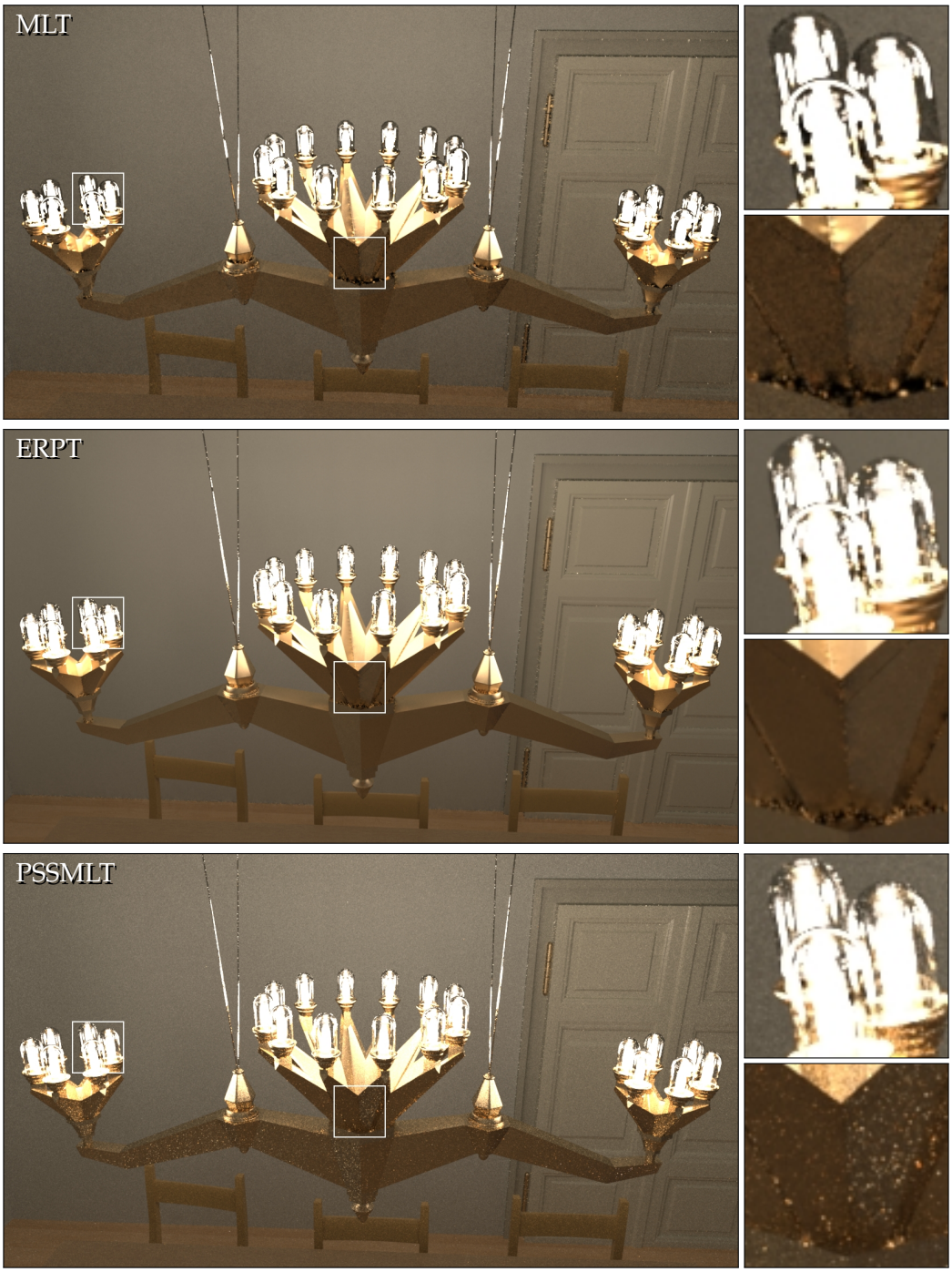


Figure 8.1: CHANDELIER: This view contains a brass chandelier with 24 light bulbs, each surrounded by a glass enclosure. The chandelier uses a realistic metal material based on microfacet theory and is attached to the ceiling using specular metal cylinders. This scene is challenging, as certain important light paths are found with low probability, particularly those involving interreflection between the bulbs and the body of the chandelier. In this and the following comparisons, one hour of processing time was allocated to each rendering technique.

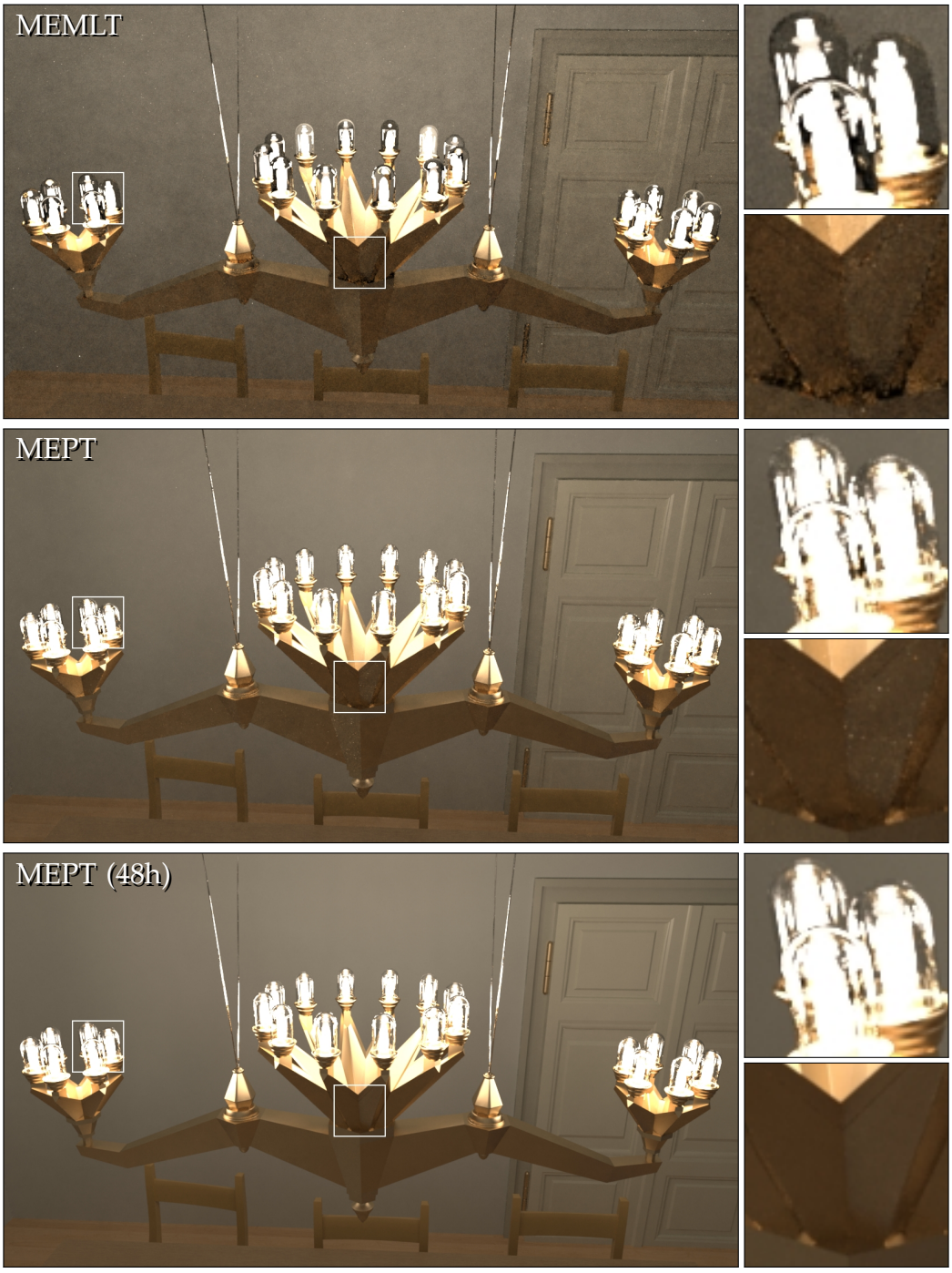


Figure 8.1: CHANDELIER (continued)

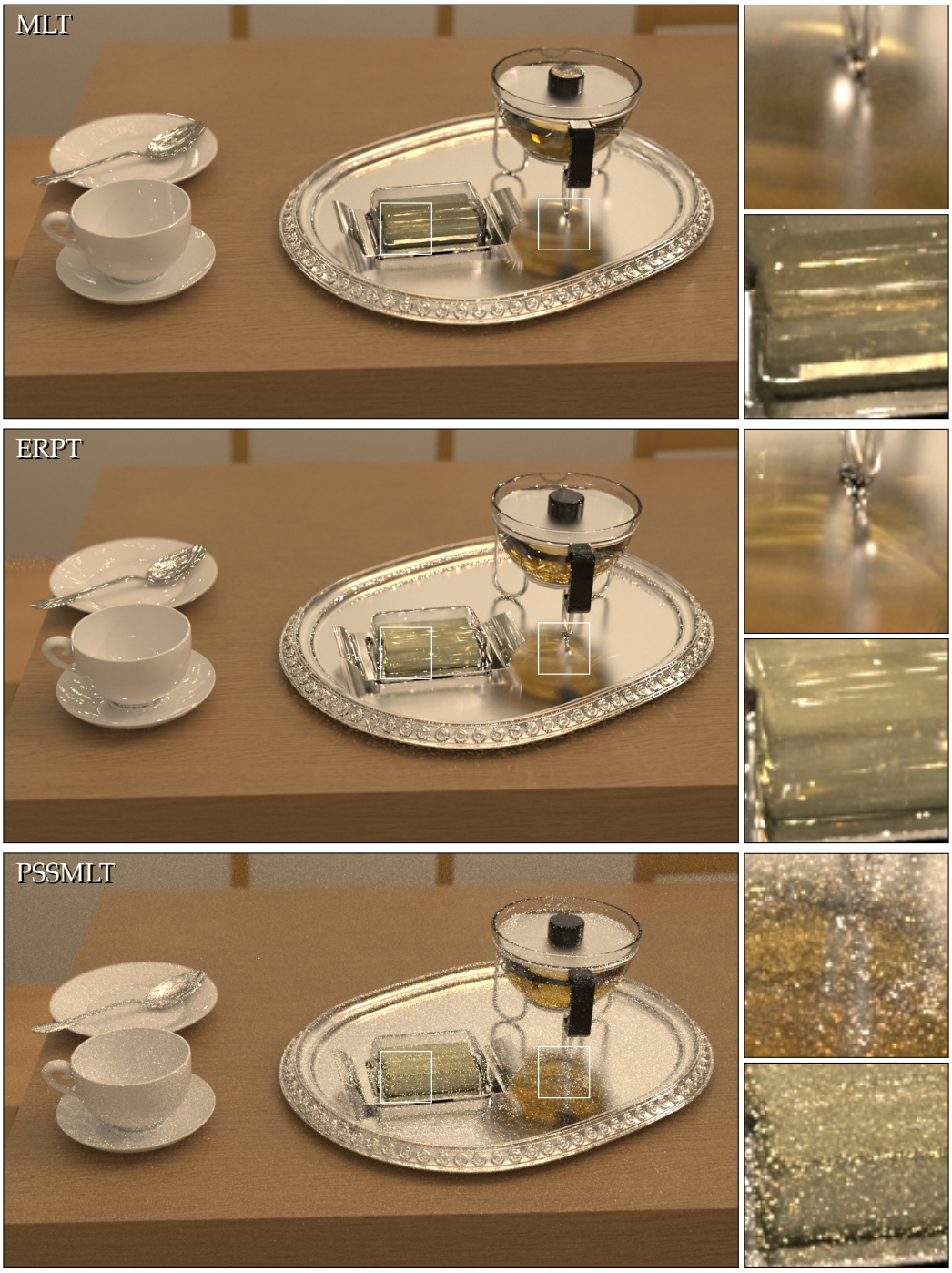


Figure 8.2: TABLE: This view of our room scene shows chinaware (using a BRDF with both diffuse and specular components), a teapot containing an absorbing medium, and a butter dish on a glossy silver tray. Illumination comes from the chandelier in Figure 8.1.

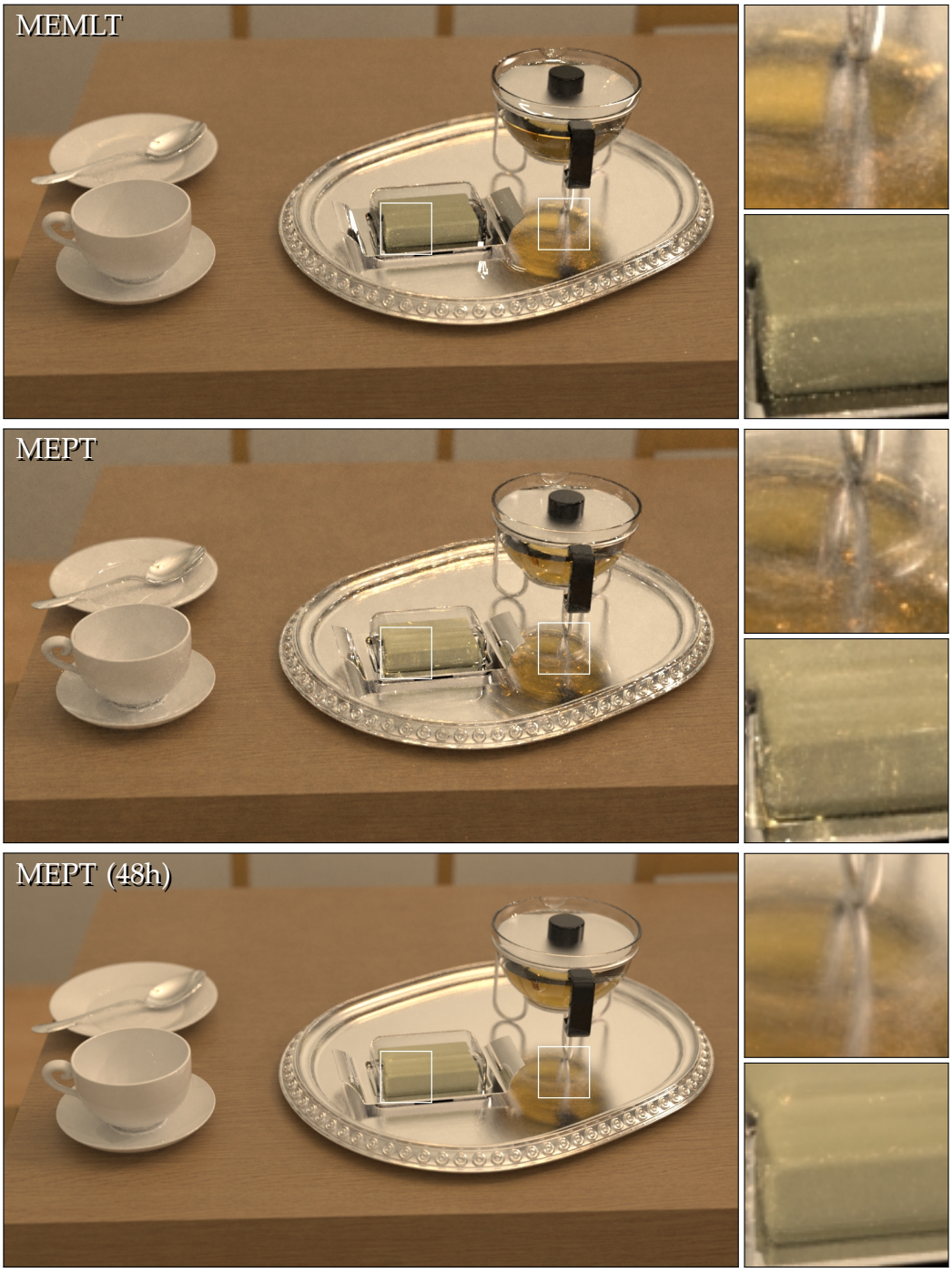


Figure 8.2: TABLE (continued)

Figure 8.3: GLASSEGG: This view of a different part of the room, now lit through windows using a spherical environment map surrounding the scene, contains a forward-scattering participating medium ($g = 0.8$) inside the glass egg.

Figure 8.3: GLASSEGG (continued)

CHAPTER 9

CONCLUSION

In this dissertation, we investigated difficulties that current unbiased rendering techniques encounter when rendering scenes involving certain types of "hard-to-find" specular light paths. We began with a self-contained derivation of the classical light transport operators and a path space integration framework that accounted for the effects of surfaces and volumes. This formalism did not make any special precautions for specular scattering; to begin to improve the behavior of unbiased methods, it was first necessary to obtain a better understanding of the joint behavior of chains of specular interactions on path space.

We therefore proposed a new theory of path-space light transport, which cleanly incorporated specular scattering into the standard rendering framework. In this theory, radiance measurements were conducted using nonsingular integrals over submanifolds of path space. We showed how to implicitly define these manifolds, how to specify the associated integrand using a generalized geometric term, and how to compute basic geometric properties, such as the manifold's tangent spaces. This led to a numerical method for moving around in the manifold using iterative root-finding, a useful building-block for exploring the neighborhood of a specular path. Due to its local nature, this method was able to avoid the complexities of performing a global search over specular paths. In practice, this meant that the search was fast, simple to implement, and that it made minimal assumptions about the underlying material type and surface representation.

Combining our results on specular light transport and the manifold walking algorithm, we proposed the manifold perturbation, a transition rule that explores the specular manifold to find the steady-state lighting distribution

of a scene as part of a Markov Chain Monte Carlo (MCMC) method. This rule was usable within the frameworks of Metropolis Light Transport (MLT) or Energy Redistribution Path Tracing (ERPT), producing rendering algorithms with support for specular paths fundamentally built in at the core.

At that point, our method could be used for scenes with both specular and non-specular transport, but the two classes were handled in fundamentally different ways. This could be undesirable—for instance, an almost-smooth dielectric or conductor would fall under the non-specular classification and hence be treated analogously to a diffuse material, preventing the associated paths from being explored effectively.

Unlike many methods for caustics and other specular phenomena, we showed that Manifold Exploration generalizes almost trivially to handle glossy surfaces and volumes that fall in between the two extremes of being diffuse and ideally specular. Similar refinements can let the same method handle perfectly anisotropic reflections, strongly oriented volume scattering media, and other kinds of problems with exactly or approximately constrained paths.

With minimal modifications to the implementation for specular surfaces, we thus also obtained a powerful new set of tools for rendering very challenging classes of light paths involving glossy materials. The end result was a Markov Chain Monte Carlo algorithm to compute lighting through very general families of paths that could involve arbitrary combinations of specular, near-specular, glossy, and diffuse surface interactions as well as isotropic or highly anisotropic volume scattering interactions. In equal-time comparisons on very challenging scenes, the methods proposed in this thesis compared favorably to previous work in Monte Carlo and MCMC rendering.

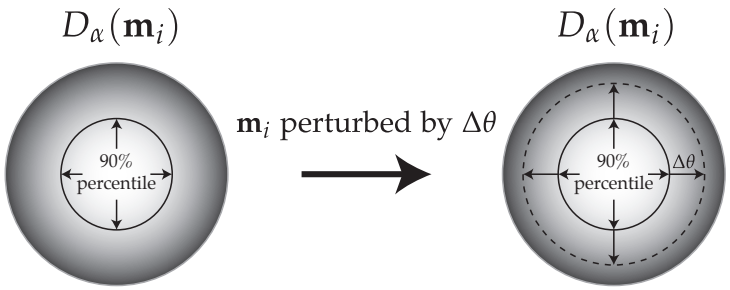
This new algorithm does still share certain limitations with its predecessors. Most importantly, it needs well distributed seed paths, because it can only explore connected components of the manifold for which seed paths are provided. Bidirectional Path Tracing is reasonably effective but still has trouble finding many components of path space, and this problem fundamentally becomes more and more difficult as their number increases. Ultimately, as the number of components exceeds the number of samples that can be generated, local exploration of path space becomes ineffective; future algorithms could be designed to attempt exploration only in sufficiently large path space components.

While MCMC rendering is a natural match for our methods of dealing with specular paths, their generality suggests interesting future applications, including purely deterministic ones. Other fields also depend on the ability to map out specular paths, for instance in the design of luminaires or optical systems, and the manifold walking algorithm may prove useful in this context.

APPENDIX A1: SPECULAR PROBABILITY FUNCTIONS

This appendix describes two specular probability functions introduced in Sections 6.2 and 7.

Specular probability function for surfaces: We found the following heuristic based on microfacet theory to work well: when the microsurface normals at $\mathbf{x}_i$ follow a distribution $D_\alpha(\mathbf{m}_i)$ with roughness parameter α, the BSDF at $\mathbf{x}_i$ will take on small values when $\mathbf{m}_i$ moves into a region where $D_\alpha(\mathbf{m}_i)$ has low density. We thus set $\psi(\mathbf{x}_i)$ by computing the expected probability that treating vertex $\mathbf{x}_i$ as non-specular during a manifold perturbation would move its microsurface normal $\mathbf{m}_i$ from a region of high density to one of low density, and we choose the 90th-percentile to classify the support of D_α into such regions.



To obtain the specular probability, our implementations must know the expected angular change $\Delta\theta$ of microsurface normals during a perturbation, which is found by briefly running the Markov chain before rendering starts. During rendering, $\psi(\mathbf{x}_i)$ is computed as the area ratio of the two highlighted regions on the sphere:

$$\psi(\mathbf{x}_i) = \frac{1 - \cos\theta_q(\alpha(\mathbf{x}_i))}{1 - \cos\left(\theta_q(\alpha(\mathbf{x}_i)) + \Delta\theta\right)}, \tag{9.1}$$

where θ_q is the aforementioned percentile (with q set to 0.9). For the Beckmann distribution, this is given by

$$\theta_q(\alpha) \coloneqq \tan^{-1}(-\alpha^2 \log(1-q)).$$

Implementation-wise, this heuristic requires material models to be able to compute their Beckmann distribution-equivalent roughness or provide a custom quantile function.

Specular probability function for participating media: In the medium case, we use the same probability (9.1), but now with a percentile that is suitable for volumetric scattering. For media using the Henyey-Greenstein phase function, this percentile is given by

$$\theta_q(g) = \cos^{-1} \frac{(1+|g|)^2 - 2(1+|g|)(1+g^2)q + 2|g|(1+g^2)q^2}{(1+|g|-2|g|q)^2}$$

where g is the *mean cosine* of the phase function, and q is set to 0.5. Other kinds of phase functions are handled by computing their associated mean cosine. Alternatively, it would also be possible to derive specific quantiles for them. In $\psi(\mathbf{x}_i)$ (Equation 9.1) we must also replace $\Delta\theta$ with the average change in scattering angle at medium vertices, again determined in a brief phase before rendering.

APPENDIX A2: DERIVATIVE COMPUTATION

Let us assume the existence of a `Vertex` data structure with the following contents:

```
struct Vertex {
    Point p;              // Vertex position
    Vector dpdu, dpdv;    // Tangent vectors
    Normal n;             // Normal vector
    Vector dndu, dndv;    // Normal derivatives
    float eta;            // Relative index of refraction
    Matrix2x2 A, B, C;    // Matrix blocks of ∇C in the row
                          // associated with the current vertex
};
```

Then the following C++ code computes the derivatives of ∇C associated with a single surface interaction vertex. It assumes that the function is called with a pointer into a list of vertices.

```
void computeDerivatives(Vertex *v) {
    /* Compute relevant directions and a few useful projections */
    Vector wi = v[-1].p - v[0].p;
    Vector wo = v[ 1].p - v[0].p;
    float ili = 1/wi.length();
    float ilo = 1/wo.length();
    wi *= ili; wo *= ilo;

    Vector H = wi + v[0].eta * wo;
    float ilh = 1/H.length();
    H *= ilh;

    float dot_H_n    = dot(v[0].n, H),
          dot_H_dndu = dot(v[0].dndu, H),
          dot_H_dndv = dot(v[0].dndv, H),
          dot_u_n    = dot(v[0].dpdu, v[0].n),
          dot_v_n    = dot(v[0].dpdv, v[0].n);

    /* Local shading tangent frame */
    Vector s = v[0].dpdu - dot_u_n * v[0].n;
    Vector t = v[0].dpdv - dot_v_n * v[0].n;
```

```
    ilo *= v[0].eta * ilh; ili *= ilh;

    /* Derivatives of C with respect to x_{i-1} */
    Vector
        dH_du = (v[-1].dpdu - wi * dot(wi, v[-1].dpdu)) * ili,
        dH_dv = (v[-1].dpdv - wi * dot(wi, v[-1].dpdv)) * ili;

    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].A = Matrix2x2(
        dot(dH_du, s), dot(dH_dv, s),
        dot(dH_du, t), dot(dH_dv, t));

    /* Derivatives of C with respect to x_i */
    dH_du = -v[0].dpdu * (ili + ilo) + wi * (dot(wi, v[0].dpdu) * ili)
                                     + wo * (dot(wo, v[0].dpdu) * ilo);
    dH_dv = -v[0].dpdv * (ili + ilo) + wi * (dot(wi, v[0].dpdv) * ili)
                                     + wo * (dot(wo, v[0].dpdv) * ilo);
    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].B = Matrix2x2(
        dot(dH_du, s) - dot(v[0].dpdu, v[0].dndu) * dot_H_n - dot_u_n * dot_H_dndu,
        dot(dH_dv, s) - dot(v[0].dpdu, v[0].dndv) * dot_H_n - dot_u_n * dot_H_dndv,
        dot(dH_du, t) - dot(v[0].dpdv, v[0].dndu) * dot_H_n - dot_v_n * dot_H_dndu,
        dot(dH_dv, t) - dot(v[0].dpdv, v[0].dndv) * dot_H_n - dot_v_n * dot_H_dndv);

    /* Derivatives of C with respect to x_{i+1} */
    dH_du = (v[1].dpdu - wo * dot(wo, v[1].dpdu)) * ilo;
    dH_dv = (v[1].dpdv - wo * dot(wo, v[1].dpdv)) * ilo;

    dH_du -= H * dot(dH_du, H);
    dH_dv -= H * dot(dH_dv, H);

    v[0].C = Matrix2x2(
        dot(dH_du, s), dot(dH_dv, s),
        dot(dH_du, t), dot(dH_dv, t));
}
```

APPENDIX A3: THE SPHERICAL VON MISES-FISHER DISTRIBUTION

The *von Mises-Fisher* (vMF) distribution [11, 39] is a popular distribution for statistical inference and many other applications involving directional data. On the 2-sphere, it is defined as

$$f_{\text{vMF}}(\omega) = \frac{\kappa}{4\pi \sinh \kappa} \exp(\kappa \mu^T \omega) \tag{9.2}$$

where $\mu \in S^2$ is the mean direction and κ denotes the *concentration* parameter ($\kappa \to 0$ approaching the uniform distribution). A recent application [16] of this distribution in computer graphics entailed fitting mixture models composed of vMF functions to arbitrary spherical data using the expectation maximization procedure. The manifold perturbation discussed in Chapter 6 relies on this distribution when sampling a perturbed outgoing direction at the vertex $\mathbf{x}_a$.

Unfortunately, many basic operations involving this distribution are prone to severe numerical issues when implemented in finite precision computer arithmetic. There is a surprising lack on information on how these can be circumvented, and hence the purpose of this appendix is to serve as a collection of numerically-well behaved recipes for common operations.

Evaluation

Evaluation of the vMF distribution easily overflows single precision arithmetic even for moderate concentration values (for instance, $\sinh 100 = 1.34406 \cdot 10^{43}$), and double precision fails shortly thereafter. The following expression derived using exponential function identities is equivalent to (9.2) and works reliably

over a much larger range of concentrations.

$$f_{\text{vMF}}(\omega) = \begin{cases} \dfrac{1}{4\pi}, & \kappa = 0 \\ \dfrac{\kappa}{2\pi(1-\exp(-2\kappa))}\, e^{\kappa(\mu^T\omega - 1)}, & \kappa > 0 \end{cases}$$

Sample generation

Several prior works have investigated how independent samples can be drawn so that they are distributed according to the vMF distribution [65, 77, 30]. The following is a brief summary of [30], which leads to a simple but numerically ill-behaved method:

Observe that the following random vector with mean direction $\mu = (0, 0, 1)$ is distributed according to f_{vMF} [65]:

$$\omega_\kappa = (\sqrt{1 - W^2}\, V, W)^T$$

where V and W are independent random variables, $V \in \mathbb{R}^2$ is a uniformly distributed vector on the unit circle, and $W \in [-1, 1]$ follows the density

$$f_W(w) = \frac{\kappa}{2\sinh\kappa} \exp(\kappa w).$$

All that is needed for a computer implementation is a way to generate realizations of W. Applying the inversion method results in

$$F_W^{-1}(\xi) = \kappa^{-1} \log\left(\exp^{-\kappa} + 2\,\xi \sinh\kappa\right) \tag{9.3}$$

To handle other values of μ, one can simply apply a rotation to directions obtained in this manner.

Numerically stable variant

Again, we can apply exponential function identities to arrive at an expression that is equivalent to (9.3) and avoids numerical issues for large values of κ:

$$F_W^{-1}(\xi) = 1 + \kappa^{-1}\left(\log \xi + \log\left(1 - \frac{\xi - 1}{\xi} e^{-2\kappa}\right)\right)$$

Finding κ such that $f_{\text{vMF}}(\mu) = c$

One very useful tool is the ability to create distributions that have a specified solid angle density into a certain direction. In the case of the von Mises-Fisher distribution, we can see that f_{vMF} takes on its maximum into direction μ, where

$$g(\kappa) := \frac{\kappa}{4\pi}(1 + \coth \kappa).$$

gives the maximum as a function of the concentration. Unfortunately, it is inconvenient to invert this expression analytically. However, note that

$$\coth \kappa = \frac{e^{2\kappa} + 1}{e^{2\kappa} - 1}$$

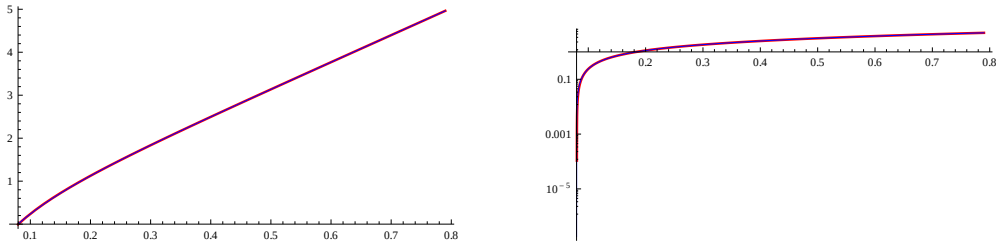
rapidly approaches 1. For instance, $\coth 5$ is already approximately equal to 1.0009. Assuming that there are no particularly stringent accuracy requirements on the inversion, we can use the following approximate scheme:

$$g^{-1}(x) \approx \begin{cases} 2\pi x, & x > g(5) \approx 0.795 \\ g_{\text{rat}}^{-1}(x), & \text{otherwise} \end{cases}$$

where we have approximated $\coth \kappa \approx 1$ for $\kappa > 5$ and make use the following rational interpolant elsewhere:

$$g_{\text{rat}}^{-1}(x) := \max\left\{10^{-5}, \frac{168.479x^2 + 16.4585x - 2.39942}{-1.12718x^2 + 29.1433x + 1}\right\}.$$

On the interval $[1/4\pi, g(5)]$, the function g_{rat}^{-1} has an absolute error of < 0.007



(a) Linear plot **(b)** Logarithmic plot

Figure A3.1: Fit of the rational function g_{rat}^{-1} (blue) to g^{-1} (red).

The relative error is infinite, as $g^{-1}(x) \to 0$ $(x \to 1/4\pi)$. Figure A3.1 shows an illustration of the fit.

Convolution

The convolution of two vMF distributions does not generally produce another vMF distribution. However, the result of this operation can be well-approximated by a vMF distribution with a suitably chosen value of κ. Mardia and Jupp [39] describe one approach to obtain this parameter, which entails approximating the distributions to be convolved by wrapped normal distributions, convolving them instead, and transforming the result back into a vMF distribution. A C implementation of this is given below:

```
float A3(float kappa) {
    return 1 / std::tanh(kappa) - 1 / kappa;
}

float dA3(float kappa) {
    float csch = 2.0f / (std::exp(kappa) - std::exp(-kappa));
    return 1 / (kappa*kappa) - csch*csch;
}

float A3inv(float y, float guess) {
    /* Initial guess */
    float x = guess, residual = 0;
```

```
    /* Invert using Newton's method */
    do {
        residual = A3(x)-y, deriv = dA3(x);
        x -= residual/deriv;
    } while (std::abs(residual) > 1e-5f);
    return x;
}

float convolve(float kappa1, float kappa2) {
    return A3inv(A3(kappa1) * A3(kappa2), std::min(kappa1, kappa2));
}
```

APPENDIX A4: IMPLEMENTING PATH-SPACE RENDERING ALGORITHMS

One of the advantages of path-space rendering algorithms is their ability to create paths in a large number of different ways due to the flexibility afforded by this framework. But this feature can also translate into difficulties when going from an abstract algorithm description to a concrete computer implementation.

Take for instance the innermost loop of a bidirectional path tracer, which is responsible for establishing connections between pairs of vertices on the light and camera subpaths. This is a simple operation when only surface interactions are involved—but in a complete implementation, we will want to be able to create every possible type of connection between pairs of surface interactions, participating medium interactions, positions on the camera, and positions on light sources. In each case, the space between the vertices may be empty or filled with a participating medium, requiring further special treatment. To make things more difficult, a single vertex may have several "identities". For example, a light source might reflect light in addition to emitting it, hence it can act both as a light source emission vertex at the end of a path, or as a surface reflection vertex somewhere in the middle of a path.

Finally, when rendering using path space, it is also important to consider that path vertices can be created in many different ways: for instance, a light source vertex can normally result from

1. independently sampling a position on a light source,
2. sampling a position on the light source given the position of some other vertex that should receive light (direct illumination sampling), or
3. intersecting a light source by chance during a random walk,

and similar holds true for the other vertex types. The available sampling strategies tend to have different density functions over their domain and must therefore be distinguished during computation. More involved algorithms like MLT and MEPT add further sampling strategies that operate by perturbing the state of an existing vertex. Given this plethora of possible vertex combinations, sampling strategies, and other special cases, it is clear why implementing such a system can be a daunting proposition.

While implementing prior work and our new method in Mitsuba, we found it crucial to design mutations and perturbation rules at a high level of abstraction to reduce the number of special cases to a minimum, and to allow the implementation code and algorithm pseudocode to look as similar as possible.

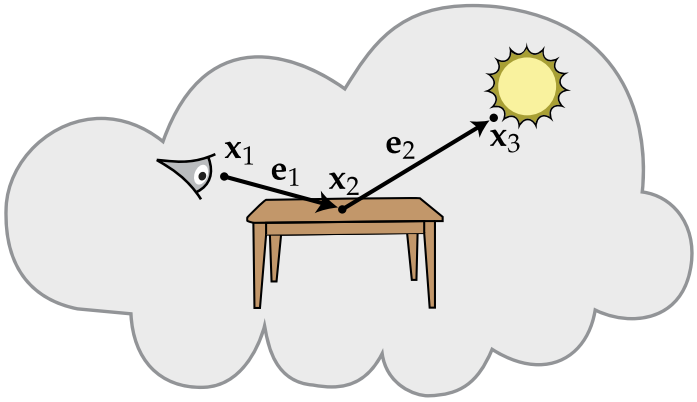
Consider the example of a mutation that generates a new vertex by tracing a ray $(\mathbf{x}_i, \omega)$ from a vertex $\mathbf{x}_i$ and intersecting it with the scene geometry, while keeping track of the amount of light throughput along the new path segment. In this case, there is no reason why the mutation's code should have to deal with the large number of possible cases, when this could also be moved into a generic object-oriented operation in the style of "$\mathbf{x}_{i+1} = \mathbf{x}_i.\textsc{Sample}(\omega)$" that is oblivious to whether $\mathbf{x}_i$ is, e.g., a camera or a medium interaction. For this reason, we prefer to move any such complexity into a special path space abstraction layer, which exposes the entire rendering system in the form of generic path edges and vertices. We reused this abstraction layer in our implementations of BDPT, PSSMLT, MLT, ERPT, MEPT, and MEMLT and found that this greatly simplified the effort of developing them.

In this system, the edges of a path represent transport, and the vertices represent both scattering interactions and the endpoints of the path. The data structures associated with edges and vertices store all relevant information that

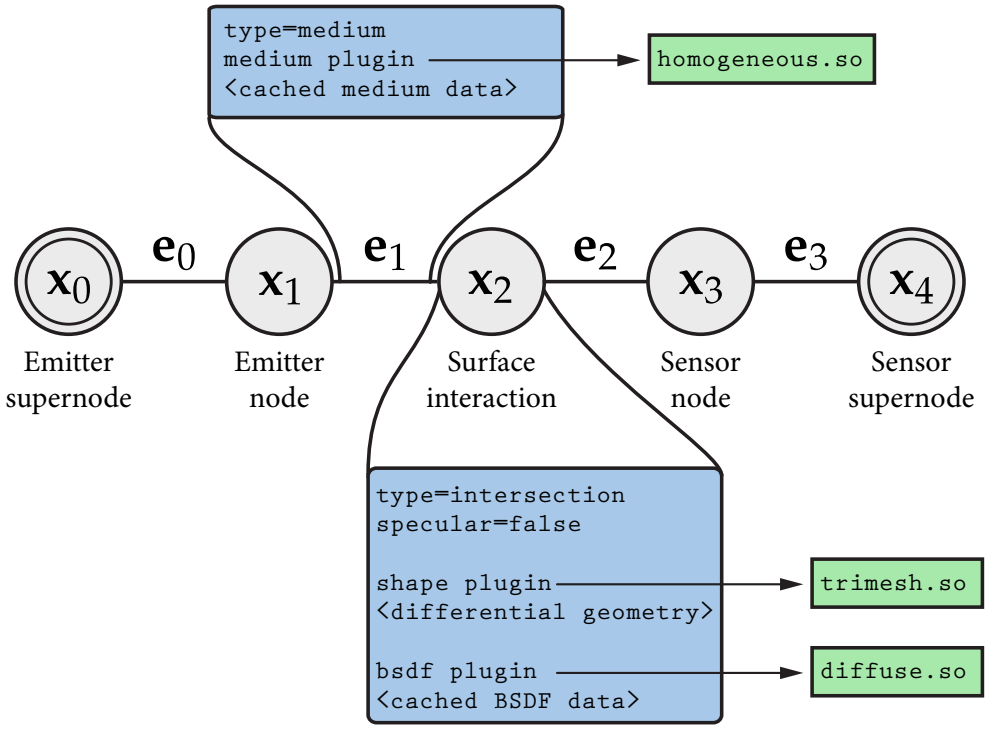
is needed to fully characterize their behavior at runtime—in particular, they record the associated terms of f_j (Figure 3.4) and cache other important data, such as information about the differential geometry of surfaces or the local medium properties.

Vertices have the capability of sampling a successor edge (e.g. by choosing a scattered direction from a material model), and edges have the capability of sampling a successor vertex (e.g. by finding an intersection with the scene geometry or generating a medium scattering event). A random walk, as it is needed for instance by BDPT, is then realized by an iteration that alternately samples edges and vertices.

Following Veach, we separate the emission and response profile of the camera and light sources into their spatial and directional components so that they can be sampled independently from one another. In our system, we found it convenient to implement this approach by representing the camera and light source using *two* vertices each. The first and last vertex of a path are stateless pseudo-vertices (referred to as "supernodes" in our system) that are not associated with any position in the domain—their only purpose is to encapsulate the operation of choosing a position on a light source or on the camera aperture in the form of a sampling operation that creates a successor vertex of a supernode. A further sampling operation on this successor vertex then selects the outgoing direction from the light source or camera. The smallest path that is possible in our system directly connects a light source to a camera and has four vertices. With these extra vertices, the light source and camera lose their special role in the implementation of mutators and perturbations; for the most part, they act just like any scattering interaction, which helps to further cut down on the number of special cases.



(a) Example path



(b) Representation in our system

Figure A4.1: This illustration shows a simple direct illumination path and the corresponding representation in our path space abstraction layer. The vertex and edge data structures (blue) store useful related information, such as the associated plugins (green), the current participating medium, or differential geometry information about an intersection.

We chose to build our system abstraction on top of the Mitsuba renderer [25], a modular physics-based renderer that was, however, not originally designed to accommodate path space rendering techniques. In Mitsuba, each "ingredient" of a scene, such as a camera, a shading model, or a geometric shape, is represented by an external plugin that is loaded at runtime. All plugins of the same type communicate with the renderer through a consistent interface, but this interface varies e.g. between BSDFs and phase functions. The abstraction layer thus has to keep track of which plugins are responsible for an edge or vertex and translate high level path space operations into the lower level operations provided by the individual plugins. Figure A4.1 relates an example path to its representation in our system. Altogether, the following types of vertices are used:

Emitter supernode: The emitter supernode is always the *first* vertex on a path. Generating a successor vertex causes the renderer to pick a light source and a position on it using the implemented importance sampling scheme, and the resulting data is stored in a newly created *emitter node*. This vertex, and the edge connecting it to the emitter node, are both stateless.

Sensor supernode: The sensor supernode is always the *last* vertex on a path. Generating a successor vertex causes the renderer to pick a position on the aperture of the camera, and the resulting information is stored in a newly created *sensor node*. This vertex, and the edge connecting it to the sensor node, are both stateless.

Emitter node: An emitter node corresponds to a point located on a light source. Generating a successor vertex causes the renderer to importance sample a direction with respect to emitted radiance, after which it traces a ray in the associated direction. Upon success, this either leads to a surface or a medium interaction vertex.

Sensor node: A sensor node corresponds to a point located the aperture of a camera. Generating a successor vertex causes the renderer to importance sample a direction with respect to emitted importance, after which it traces a ray in the associated direction. Upon success, this either leads to a surface or a medium interaction vertex.

Surface interaction: This vertex stores all information pertaining to a scattering event located on a surface. Generating a successor vertex causes the renderer to pick a direction according to the surface's BSDF, after which it either generates another surface or a medium interaction vertex.

Medium interaction: This vertex stores all information pertaining to a scattering event located somewhere inside a participating medium. Generating a successor vertex causes the renderer to pick a direction according to the medium's phase function, after which it either generates another medium or a surface interaction vertex.

Our abstraction supports the following key methods on the vertices:

1. $\mathbf{e}_i,\ \mathbf{x}_{i+1} = \textsc{SampleNext}(\mathbf{x}_{i-1} \xrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i)$

   Given two preceding vertices and an edge between them, this operation samples a successor edge and vertex. Internally, this operation invokes the importance sampling scheme associated with the scattering model, camera response profile, or light source emission profile that underlies the vertex $\mathbf{x}_i$.

   When the new edge $\mathbf{e}_i$ passes through a participating medium, the medium's importance sampling code determines whether the vertex $\mathbf{x}_{i+1}$ is a volume scattering interaction or a point on a surface.

2. $\textsc{Eval}(\mathbf{x}_{i-1} \xrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i \xrightarrow{\mathbf{e}_i} \mathbf{x}_{i+1})$

   Given three adjacent vertices, this function evaluates the part of the measurement contribution function (3.13) that is associated with the middle vertex $\mathbf{x}_i$. A similar function also exists for edges.

3. $\textsc{EvalPdf}(\mathbf{x}_{i-1} \xrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i \xrightarrow{\mathbf{e}_i} \mathbf{x}_{i+1})$

   This function computes the area density that SampleNext produces on the vertex $\mathbf{x}_{i+1}$ when invoked with the path segment $\mathbf{x}_{i-1} \xrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i$.

4. $\textsc{Perturb}(\mathbf{x}_{i-1} \xrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i \xrightarrow{\mathbf{e}_i} \mathbf{x}_{i+1}, \omega)$

   This function perturbs the outgoing direction along the path segment $\mathbf{x}_i \to \mathbf{x}_{i+1}$. Similar perturbation operations also exist to adjust the position on the light source or camera aperture, or the length of medium edges.

5. $\mathbf{x}'_i = \textsc{Cast}(\mathbf{x}_i, \langle\text{desired type}\rangle)$

   Sometimes, vertices must be cast into a different type. Consider the hypothetical example that the vertex $\mathbf{x}_2$ in Figure A4.1 lies on a light source that emits and reflects light. A bidirectional mutation [70] might cut out $\mathbf{x}_1$ from the path and reconnect the two resulting subpaths, in which case $\mathbf{x}_2$ shifts to the position 1 and becomes the emitter associated with the current path. In this case, this function is used to cast the vertex into an emitter node, which fails when such a reinterpretation is not possible. Cast is internally used by the Connect method whenever necessary.

6. $\mathbf{e}_i = \textsc{Connect}(\mathbf{x}_{i-1} \xleftrightarrow{\mathbf{e}_{i-1}} \mathbf{x}_i \dashleftarrow\dashrightarrow \mathbf{x}_{i+1} \xleftrightarrow{\mathbf{e}_{i+1}} \mathbf{x}_{i+2})$

   This function joins two disconnected subpaths created using arbitrary sampling techniques. It verifies that there is nonzero throughput between $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ and returns a new connection edge. Any cached data in the vertices that may have changed due to the connection is also updated.

In bidirectional rendering algorithms that sample light paths starting both from the light source and from the camera, it is important to keep track of certain non-symmetries in scattering models, which Veach [66] discusses in detail. All of the above functions are aware of these non-symmetries: depending on the indicated direction of transport, they appropriately query or sample the standard version, or the adjoint of the associated scattering models.

With this framework in place, most path space rendering methods turn into simple sequences of these operations. For instance, our implementation of BDPT creates two subpaths that initially only contain endpoint supernodes, and repeatedly calls the SampleNext operation to perform two random walks. Following this, the Connect function is invoked on every pair of vertices from the two subpaths, and their contribution is recorded upon success. This involves querying the measurement contribution function via Eval and the sampling density via EvalPdf to compute multiple importance sampling weights.

Currently, our system simulates surface and volumetric scattering but does not support some other types of transport like Bidirectional Surface Scattering Distribution Functions (BSSRDFs) [46]; these are a popular way of summarizing the aggregate effects of volumetric scattering that occurs in an object. Simulating subsurface scattering using BSSRDFs is often considerably faster than doing so using the radiative transfer equation (usually also involving some loss of accuracy). However, such forms of scattering could be incorporated into our system by adding another type of edge that describes subsurface transport. We leave such extensions for future work.

The abstraction layer, as well as our implementations of prior work and the proposed methods are available in Mitsuba as of October 1, 2012. It is hoped that they will facilitate future research involving path space rendering techniques.

BIBLIOGRAPHY

[1] James Richard Arvo. *Analytic methods for simulated light transport*. PhD thesis, Yale University, 1995.

[2] K.M. Case and P.F. Zweifel. *Linear transport theory*, volume 29. Addison-Wesley Reading, Mass., 1967.

[3] Jiating Chen, Bin Wang, and Jun-Hai Yong. Improved stochastic progressive photon mapping with Metropolis sampling. *Computer Graphics Forum*, 30(4):1205–1213, 2011.

[4] Min Chen and J. Arvo. Perturbation methods for interactive specular reflections. *IEEE Transactions on Visualization and Computer Graphics*, 6(3):253–264, July/September 2000.

[5] Min Chen and James Arvo. Theory and application of specular path perturbation. *ACM Transactions on Graphics*, 19(4):246–278, October 2000.

[6] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tacing. *ACM Transactions on Graphics*, 24(3):1186–1195, August 2005.

[7] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, pages 137–145, July 1984.

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

[9] A. Doucet, A.M. Johansen, and V.B. Tadic. On solving integral equations using Markov Chain Monte Carlo methods. *Applied Mathematics and Computation*, 2010.

[10] Philip Dutré, Kavita Bala, Philippe Bekaert, and Peter Shirley. *Advanced Global Illumination*. AK Peters Ltd, 2006.

[11] N.I. Fisher, T. Lewis, and B.J.J. Embleton. *Statistical analysis of spherical data*. Cambridge University Press, 1993.

[12] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 84)*, pages 213–222, 1984.

[13] Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, pages 223–242, 2001.

[14] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Transactions on Graphics*, 28(5), December 2009.

[15] Toshiya Hachisuka and Henrik Wann Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Transactions on Graphics*, 30(5):114:1–114:11, October 2011.

[16] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency domain normal map filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007)*, 26(3):28:1–28:12, 2007.

[17] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[18] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. *ACM Transactions on Graphics*, 26(3), July 2007.

[19] Eugene Hecht. *Optics (4th Edition)*. Addison Wesley, 4 edition, August 2001.

[20] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, pages 145–154, August 1990.

[21] Lukas Hosek and Alexander Wilkie. An analytic model for full spectral sky-dome radiance. *ACM Transactions on Graphics*, 31(4):95:1–95:9, July 2012.

[22] Matthias B. Hullin, Johannes Hanika, Boris Ajdin, Hans-Peter Seidel, Jan Kautz, and Hendrik P. A. Lensch. Acquisition and analysis of bispectral bidirectional reflectance and reradiation distribution functions. *ACM Transactions on Graphics*, 29(4):97:1–97:7, July 2010.

[23] Homan Igehy. Tracing ray differentials. In *Computer Graphics (Proceedings of SIGGRAPH 99)*, pages 179–186, August 1999.

[24] Akira Ishimaru. *Wave propagation and scattering in random media*, volume 2. Academic Press, New York, 1978.

[25] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[26] Wenzel Jakob, Adam Arbree, Jonathan T. Moon, Kavita Bala, and Steve Marschner. A radiative transfer framework for rendering materials with anisotropic structure. *ACM Transactions on Graphics*, 29(4):53:1–53:13, 2010.

[27] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum*, 27(2):557–566, April 2008.

[28] Henrik Wann Jensen. Global illumination using photon maps. In *Eurographics Rendering Workshop 1996*, pages 21–30, June 1996.

[29] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *Computer Graphics (Proceedings of SIGGRAPH 98)*, pages 311–320, July 1998.

[30] Sungkyu Jung. Generating von Mises Fisher distribution on the unit sphere (s^2). Technical report, University of Pittsburgh, 10 2009.

[31] James T. Kajiya. The rendering equation. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, pages 143–150, August 1986.

[32] C. Kelemen, L. Szirmay-Kalos, G. Antal, and F. Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Computer Graphics Forum*, 21(3):531–540, 2002.

[33] Alexander Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 49–56. ACM Press/Addison-Wesley Publishing Co., 1997.

[34] Shinya Kitaoka, Yoshifumi Kitamura, and Fumio Kishino. Replica exchange light transport. *Computer Graphics Forum*, 28(8):2330–2342, December 2009.

[35] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of Compugraphics 93*, pages 145–153, Alvor, Portugal, 1993.

[36] Yu-Chi Lai, Shao Hua Fan, Stephen Chenney, and Charcle Dyer. Photorealistic image rendering with population Monte Carlo energy redistribution. In *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, pages 287–296, June 2007.

[37] Edward W Larsen and Richard Vasques. A generalized linear Boltzmann equation for non-classical particle transport. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 112(4):619–631, 2011.

[38] Jun S. Liu. *Monte Carlo strategies in scientific computing*. Springer, 2001.

[39] K.V. Mardia and P.E. Jupp. *Directional statistics*. Wiley, 2000.

[40] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6), 1953.

[41] Michael I. Mishchenko, Larry D. Travis, and Andrew A. Lacis. *Multiple scattering of light by particles*. Cambridge University Press, 2006.

[42] Don P. Mitchell and Pat Hanrahan. Illumination from curved reflectors. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, pages 283–291, July 1992.

[43] Srinivasa G. Narasimhan, Mohit Gupta, Craig Donner, Ravi Ramamoorthi, Shree K. Nayar, and Henrik Wann Jensen. Acquiring scattering properties of participating media by dilution. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 1003–1012. ACM, 2006.

[44] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of BRDF models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226. Eurographics Association, 2005.

[45] Duc Quang Nguyen, Ronald Fedkiw, and Henrik Wann Jensen. Physically based modeling and animation of fire. *ACM Transactions on Graphics*, 21(3):721–728, 2002.

[46] United States. National Bureau of Standards and Fred Edwin Nicodemus. *Geometrical considerations and nomenclature for reflectance*, volume 160. US Department of Commerce, National Bureau of Standards Washington, D. C, 1977.

[47] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Rendering Techniques 2000: 11th Eurographics Workshop on Rendering*, pages 11–22, June 2000.

[48] Matt Pharr. Monte Carlo Ray Tracing, Course notes, chapter 9. In *ACM SIGGRAPH*, 2003.

[49] Matt Pharr and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[50] A. J. Preetham, Peter Shirley, and Brian Smits. A practical analytic model for daylight. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 91–100. ACM Press/Addison-Wesley Publishing Co., 1999.

[51] R.W. Preisendorfer. *Hydrologic optics*. US Department of Commerce, 1976.

[52] Simon Premoze, Michael Ashikhmin, and Peter Shirley. Path integration for light transport in volumes. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, June 2003.

[53] G.O. Roberts, A. Gelman, and W.R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, 1997.

[54] B. Segovia, J.C. Iehl, and B. Péroche. Metropolis instant radiosity. *Computer Graphics Forum*, 26(3):425–434, September 2007.

[55] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R. Marschner, Mark Horowitz, Marc Levoy, and Hendrik P. A. Lensch. Dual photography. *ACM Transactions on Graphics*, 24(3):745–755, July 2005.

[56] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte Carlo techniques for direct lighting calculations. *ACM Transactions on Graphics*, 15(1):1–36, 1996.

[57] Peter S. Shirley, Bretton Wade, Philip Hubbard, David Zareski, Bruce Walter, and Donald P. Greenberg. Global illumination via density estimation. In *Eurographics Rendering Workshop 1995*, pages 219–231, June 1995.

[58] François X. Sillion and Claude Puech. A general two-pass method integrating specular and diffuse reflection. In *Computer Graphics (Proceedings of SIGGRAPH 89)*, pages 335–344, July 1989.

[59] Arnold Sommerfeld and J Runge. Anwendung der vektorrechnung auf die grundlagen der geometrischen optik. *Annalen der Physik*, 340(7):277–298, 1911.

[60] Michael Spivak. *Calculus on manifolds*. Addison-Wesley, 1965.

[61] Jos Stam. Diffraction shaders. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 101–110. ACM Press/Addison-Wesley Publishing Co., 1999.

[62] Jerry Tessendorf. Angular smoothing and spatial diffusion from the feynman path integral representation of radiative transfer. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 112(4):751 – 760, 2011. 2009 International Conference on Mathematics and Computational Methods (M&C 2009).

[63] L. Tierney. A note on Metropolis-Hastings kernels for general state spaces. *Annals of Applied Probability*, 8(1):1–9, 1998.

[64] Kenneth E Torrance and Ephraim M Sparrow. Theory for off-specular reflection from roughened surfaces. *JOSA*, 57(9):1105–1112, 1967.

[65] G. Ulrich. Computer generation of distributions on the m-sphere. *Applied Statistics*, pages 158–163, 1984.

[66] Eric Veach. Non-symmetric scattering in light transport algorithms. In *Proceedings of the eurographics workshop on rendering techniques '96*, pages 81–90, London, UK, UK, 1996. Springer-Verlag.

[67] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1997.

[68] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Fifth Eurographics Workshop on Rendering*, June 1994.

[69] 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, pages 419–428. ACM, 1995.

[70] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Computer Graphics (Proceedings of SIGGRAPH 97)*, pages 65–76, August 1997.

[71] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional lightcuts. *ACM Transactions on Graphics*, 25(3):1081–1088, July 2006.

[72] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a scalable approach to illumination. *ACM Transactions on Graphics*, 24(3):1098–1107, July 2005.

[73] Bruce Walter, Philip M. Hubbard, Peter S. Shirley, and Donald F. Greenberg. Global illumination using local linear density estimation. *ACM Transactions on Graphics*, 16(3):217–259, July 1997.

[74] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Rendering Techniques 2007: 18th Eurographics Workshop on Rendering*, pages 195–206, June 2007.

[75] Bruce Walter, Shuang Zhao, Nicolas Holzschuch, and Kavita Bala. Single scattering in refractive media with triangle mesh boundaries. *ACM Transactions on Graphics*, 28(3):92:1–92:8, July 2009.

[76] Turner Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, June 1980.

[77] A.T.A. Wood. Simulation of the von Mises Fisher distribution. *Communications in statistics-simulation and computation*, 23(1):157–164, 1994.