Informed Choices in Primary Sample Space

Benedikt Bitterli

Master Thesis
August 2015

Advised by
Wenzel Jakob, Wojciech Jarosz, Jan Novak

Overseen by Markus Gross

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

igl

Disney Research

Abstract

In this thesis, we investigate mappings from primary sample space to path space and their inverses. We give special consideration to how these mappings interact with Markov Chain Monte Carlo rendering methods operating in primary sample space. In particular, we show that such methods perform uncontrolled changes to light transport paths in the presence of geometric and material discontinuities, discrete choices and multiple sampling techniques. We also show that ignoring internals of these mappings can lead to poor noise distribution on the image plane and low acceptance rates for large steps. Our contributions are three-fold: We describe how to construct inverses of several path sampling techniques employed in graphics in order to robustly turn transport paths back into the random numbers that produced them, and use these inverses to create two new perturbations for Multiplexed Metropolis Light Transport; we introduce Multiple Correlated-Try Metropolis to Markov Chain Monte Carlo rendering and apply it to create a new large step mutation based on bidirectional connections; and we show how annotating dimensions of primary sample space with information about how they are employed in the path sampling process can make Markov Chain Monte Carlo rendering methods more robust, and give two applications of such an approach.

Zusammenfassung

In dieser Arbeit untersuchen wir Funktionen vom Raum der Zufallszahlen zum Raum der Lichtpfade, sowohl deren Inversen. Insbesondere untersuchen wir wie diese Funktionen mit Hinsicht auf Renderingverfahren basierend auf Markov-Ketten Monte Carlo. Wir zeigen, dass solche Verfahren unkontrollierte Veränderungen am Lichtpfad ausführen, wenn diskrete Entscheidungen, Geometrie- oder Materialdiskontinuitäten oder mehrere Samplingtechniken involviert sind. Wir zeigen auch, dass das Ignorieren der Funktionsweise dieser Funktionen zu einer unschönen Verteilung des Rauschens im Bild und zu geringen Akzeptanzwahrscheinlichkeiten von *Large Steps* führen kann. Wir fassen die Beiträge dieser Arbeit wie folgt zusammen: Wir beschreiben, wie man Inversen diverser Samplingmethoden für Lichtpfade konstruieren kann, um Lichtpfade zurück in Zufallszahlen zu transformieren, und beschreiben zwei neue *perturbations* für Multiplexed Metropolis Light Transport basierend auf diesen Inversen; wir beschreiben Multiple Correlated-Try Metropolis in Hinsicht auf Markov-Ketten Monte Carlo und benutzen es, um einen neuen Large Step basierend auf zweiseitigen Verbindungen zu konstruieren; und wir zeigen, wie zusätzliche Information über die Funktionsweise der Samplingmethoden und ihrer Verwendung der Dimensionen des Raums der Zufallszahlen benutzt werden kann, um Markov-Ketten Monte Carlo Methoden mehr robust zu machen, und beschreiben zwei Anwendungen dessen.

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my advisors for their continuous support, supervision and advice during this thesis; Wojciech Jarsoz, who continued to devote his time even in the middle of his move across continents and time zones; Jan Novak, who did not even let a vacation get in the way of reviewing my work; and Wenzel Jakob, who agreed to join this project more than halfway through, and without whom this thesis would have gone entirely differently.

My sincere thanks also go to Aaron Griffith for answering my misguided math questions, and Andrew Chin for his support in setting up last-minute render jobs.

My deepest appreciation also goes to ETH for supporting me with their generous Master Scholarship Award, as well as all of the people who made that possible.

Finally, none of this would have been possible without the unwavering love and patience of my family, friends and loved ones, and my heart-felt gratitude goes out to them.

Contents

List of Figures

CHAPTER 1

Introduction

The synthesis of images depicting virtual scenes – so-called renderings – has a long-standing tradition in the field of computer graphics. The ever-increasing demand for visual fidelity and realism of these renderings and their pervasive use in the entertainment industry have been the main driving forces behind rendering research in both industry and academia, and it remains an important research topic to this day.

Physically based rendering, which strives to simulate light and the physical laws (or approximations thereof) that govern its interaction with matter, is one particular subfield of rendering that has seen a rise in popularity over the past decade. Such an approach to rendering, firmly rooted in physically motivated principles rather than *ad hoc* artist-driven models, promises to faithfully reproduce many of the important lighting effects we can observe in the real world. However, physically-based techniques also come with great computational demand, and only the steady increase in computational resources over the past few decades has made pursuit of this approach feasible for movie and video game production.

Light transport is one aspect of physically based rendering which studies light and its interactions with the virtual world. Given a description of a virtual scene and the materials comprising it, light transport algorithms are capable of simulating light as it propagates through the scene and arrives at the virtual camera. Research in this field has spawned a wealth of different algorithms that attempt to solve the light transport problem. Interestingly, these algorithms do not necessarily differ in the fidelity of the images they are able to compute – indeed, for a given scene, most of them will produce the same solution if run long enough – but rather the time they require to produce a visually acceptable result. This is of special practical relevance, since we only ever have a finite amount of time available to render an image.

In this thesis, we are concerned with one particular class of light transport algorithms, referred to as *Markov Chain Monte Carlo* methods. These algorithms are currently

among the most robust unbiased rendering algorithms available, and have been a popular research topic in recent years. In methods based on Monte Carlo integration, random numbers form a natural part of the process that generates the image. In particular, Monte Carlo methods utilize deterministic mappings from random numbers to points in a different space, the *path space*, to obtain part of the solution. These mappings are normally thought of in a forward sense only – from random numbers to points in path space. However, in this thesis we will show how inverses of these mappings can be constructed, and we will subsequently utilize such inverses in a Markov Chain process to produce a viable rendering algorithm. We also show how a small amount of additional information about these mappings can help Markov Chain rendering algorithms produce more visually pleasing results.

1.1 Thesis Overview

This thesis consists of seven chapters. In Chapter 2 we will review the fundamentals of light transport and describe the light transport problem. In Chapter 3 we describe practical solution techniques to the light transport problem and review the related work relevant to this thesis. These two chapters will also introduce the notation used throughout the remainder of the text. In Chapter 4 we highlight some of the shortcomings of previous Markov Chain Monte Carlo methods operating on random numbers and show how they can be fixed by exposing some information about the mapping from random numbers to path space. In Chapter 5 we show how mappings to path space can be inverted and show two applications of such an inverse in a Markov Chain Monte Carlo rendering method. We compare our proposed techniques to previous work in Chapter 6 and conclude the thesis in Chapter 7.

CHAPTER 2

Fundamentals of Light Transport

The main purpose of light transport algorithms in computer graphics is to synthesize realistic images of virtual scenes. Given a full description of the surface geometry of a scene as well as the physical properties of these surfaces, light transport captures the behaviour of light as it is emitted, transported, scattered, and measured, when it ultimately meets with the camera. To produce an image, light transport algorithms must simulate the physical laws governing the interactions of light with the virtual world.

Several models of the physics of light exist today, differing in their complexity and ability to model physical effects. These models include geometric optics, wave optics, electromagnetic optics and quantum optics, listed in order of increasing completeness. Most of computer graphics is built within the framework of geometric optics, which is the simplest of the models mentioned here. Geometric optics makes several simplifying assumptions and considers light to travel in straight lines, which makes efficient simulation tractable. Certain physical phenomena, such as interference or diffraction, cannot be captured by such a model. However, for most scenes encountered in practical rendering, the visual impact of these phenomena is not considered significant enough to warrant a more expensive computational model.

In geometric optics, the quantity of interest is the *radiance*. For a given point $\mathbf{x}$ and direction ω, the radiance $L(\mathbf{x}, \omega)$ measures the amount of energy arriving from direction ω at a small (hypothetical) surface patch $\mathrm{d}A$ at $\mathbf{x}$, oriented perpendicularly to ω.

It is often convenient to also define the more intuitive *incident radiance* and *exitant radiance*. When we talk about surfaces, the incident radiance $L_i(\mathbf{x}, \omega)$ describes the radiance arriving at the surface point $\mathbf{x}$ from direction ω, whereas the exitant radiance $L_o(\mathbf{x}, \omega)$ describes the radiance leaving the same surface point towards direction ω. Intuitively, these two quantities differ only in the sign of ω and we can think of them in terms of the radiance as $L_i(\mathbf{x}, \omega) = L(\mathbf{x}, \omega)$ and $L_o(\mathbf{x}, \omega) = L(\mathbf{x}, -\omega)$. We refer the

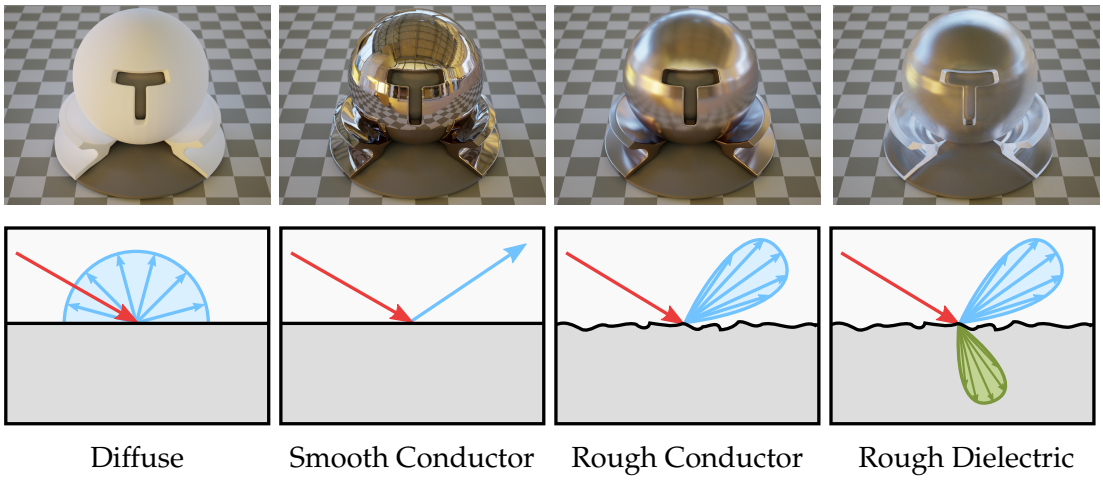


Figure 2.1: *Renderings and illustrations of four different BSDFs. The incident beam of light is marked in red; reflected and transmitted light is shown in blue and green, respectively.*

reader to Veach [Vea98, Section 3.5] for a more rigorous definition.

Of main interest for this thesis is the interaction of light with surfaces, as described in this section. For a more general overview of light transport on surfaces and participating media, we refer the reader to Pharr and Humphreys [PH10] and Jarosz [Jar08].

2.1 The BSDF

In order to describe the appearance of surfaces under varying illumination, we require a model of surface reflectance. Nicodemus et al. [NRH+77] describe a formal framework to reason about such reflectance functions, and introduce the bidirectional reflectance distribution function (BRDF) to model reflection from arbitrary surfaces. A straightforward generalization of their work to transmissive surfaces is the *bidirectional scattering distribution function* (BSDF), which we will use in this thesis. The BSDF is a function of surface location $\mathbf{x}$, incident direction ω_i and exitant direction ω_o, and describes the fraction of light arriving from direction ω_i that is scattered towards direction ω_o. Intuitively, if we were to shine a flashlight at $\mathbf{x}$ from direction ω_i, the BSDF would tell us how bright the surface at $\mathbf{x}$ would appear when viewed from direction ω_o. More precisely,

$$f_S(\mathbf{x}, \omega_o, \omega_i) = \frac{\mathrm{d}L_o(\mathbf{x}, \omega_o)}{L_i(\mathbf{x}, \omega_i)|N(\mathbf{x}) \cdot \omega_i|\mathrm{d}\sigma(\omega_i)}\,. \tag{2.1}$$

For many surfaces, the BSDF is the dominating feature governing its appearance under illumination. We illustrate a few example BSDFs in Figure 2.1.

In Equation 2.1, we used a previously undiscussed term, the *foreshortening factor*, $|N(\mathbf{x}) \cdot \omega_i|$. Remember that we defined the radiance $L(\mathbf{x}, \omega)$ with respect to a hypothetical surface patch oriented perpendicular to ω. However, in Equation 2.1, the actual surface patch on which $\mathbf{x}$ is located is oriented perpendicular to the *surface normal* $N(\mathbf{x})$, which may be different from the incident direction ω_i. This incurs an additional term describing the decrease in radiance as the surface is tilted away from the incident direction. In order to explain the intuition behind this factor, we can picture a scenario where we shine a flashlight perpendicularly onto a surface, illuminating a circular area. As we tilt the flashlight away from the perpendicular orientation, the circle illuminated by the light source is stretched along one direction and illuminates an increasingly larger area. The same amount of energy is emitted from the flash light, but it is distributed onto a larger surface area, causing a decrease in radiance received by each illuminated differential surface element. This decrease in radiance is proportional to the foreshortening factor $|N(\mathbf{x}) \cdot \omega_i|$.

2.2 The Rendering Equation

The radiance leaving a surface point can be described as the sum of two terms: The *emitted radiance* L_e, and the *scattered radiance* L_s:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + L_s(\mathbf{x}, \omega_o)\,. \tag{2.2}$$

The emitted radiance describes the self-emission of the surface at $\mathbf{x}$. This term allows us to model light sources in the scene, including naturally occurring emitters such as the sun or artificial emitters such as candles or light bulbs.

In turn, the scattered radiance describes the light that is received by $\mathbf{x}$ from other surfaces and reradiated towards direction ω_o:

$$L_s(\mathbf{x}, \omega_o) = \int_{\mathcal{S}^2} f_S(\mathbf{x}, \omega_o, \omega_i) L_i(\mathbf{x}, \omega_i) |N(\mathbf{x}) \cdot \omega_i| \mathrm{d}\sigma(\omega_i)\,. \tag{2.3}$$

The integrand consists of three terms of interest: The incident radiance $L_i(\mathbf{x}, \omega_i)$ describes the amount of light that arrives at $\mathbf{x}$ from the incident direction ω_i; the foreshortening term $|N(\mathbf{x}) \cdot \omega_i|$ describes how much of the incident radiance is received by the surface oriented with normal $N(\mathbf{x})$; and the BSDF $f_S(\mathbf{x}, \omega_o, \omega_i)$ describes how much of the light received is scattered towards the outgoing direction ω_o. The integral

over all incident directions then gives the total amount of radiance scattered towards ω_o.

Here, we used the sphere of directions $\mathcal{S}^2$ as the integration domain; we will denote $\mathrm{d}\sigma$ as the *solid angle measure* on this domain. We can rewrite Equation 2.3 more succinctly using the *projected solid angle measure*, $\mathrm{d}\sigma^\perp(\mathbf{x}, \omega) = |N(\mathbf{x}) \cdot \omega| \mathrm{d}\sigma(\omega)$:

$$L_s(\mathbf{x}, \omega_o) = \int_{\mathcal{S}^2} f_S(\mathbf{x}, \omega_o, \omega_i) L_i(\mathbf{x}, \omega_i) \mathrm{d}\sigma^\perp(\mathbf{x}, \omega_i) \,. \tag{2.4}$$

By inserting Equation 2.4 into Equation 2.2, we arrive at the following energy balance equation:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathcal{S}^2} f_S(\mathbf{x}, \omega_o, \omega_i) L_i(\mathbf{x}, \omega_i) \mathrm{d}\sigma^\perp(\mathbf{x}, \omega_i) \,. \tag{2.5}$$

This equation is called the *rendering equation*, originally introduced by Kajiya [Kaj86]. Solving this equation is the light transport problem.

2.2.1 Surface Area Formulation

Equation 2.5 is formulated in terms of an integral over the sphere of directions, which is referred to as the (hemi)spherical form of the rendering equation. Sometimes it is more convenient to reformulate it as an integral over surfaces instead.

In the following, we denote the union of all surfaces in the scene as $\mathcal{M}$. We will perform integration in this domain with respect to the *surface area measure* $\mathrm{d}A$.

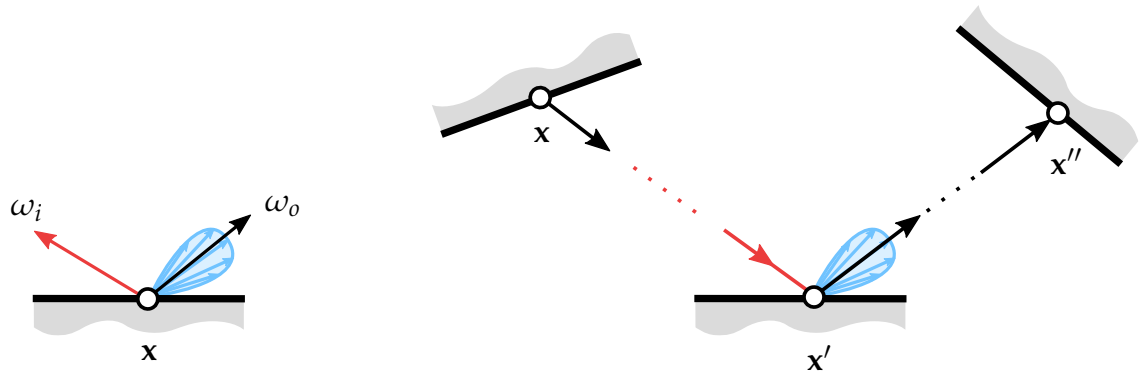
Following Kajiya [Kaj86] and Veach [Vea98], we rewrite the quantities appearing in the rendering equation in order to eliminate the directions ω_o and ω_i and define them purely in terms of positions. We first write the radiance in terms of two surface points in the scene, $\mathbf{x}, \mathbf{x}' \in \mathcal{M}$:

$$L(\mathbf{x} \to \mathbf{x}') := L\left(\mathbf{x}, \frac{\mathbf{x}' - \mathbf{x}}{||\mathbf{x}' - \mathbf{x}||}\right) , \tag{2.6}$$

where arrows denote the direction of light flow; $L(\mathbf{x} \to \mathbf{x}')$ reads as "the radiance leaving $\mathbf{x}$ toward $\mathbf{x}'$". Similarly, using a third surface point $\mathbf{x}'' \in \mathcal{M}$ we can rewrite the BSDF as

$$f_S(\mathbf{x} \to \mathbf{x}' \to \mathbf{x}'') := f_S\left(\mathbf{x}, \frac{\mathbf{x}'' - \mathbf{x}'}{||\mathbf{x}'' - \mathbf{x}'||}, \frac{\mathbf{x} - \mathbf{x}'}{||\mathbf{x} - \mathbf{x}'||}\right) . \tag{2.7}$$

It is worth noting that incident direction ω_i is reversed compared to the flow of light. Figure 2.2 illustrates the two different conventions.



(a) Directional Form **(b)** Three-point form

Figure 2.2: *The directional (a) and three-point form (b) of the BSDF. In the directional form, the directions always point away from the surface. In the three-point form, the directions align with the direction of light flow, effectively reversing the incident direction compared to (a).*

Moving from integration with respect to projected solid angle measure to area measure incurs a Jacobian determinant [Kaj86],

$$\mathrm{d}\sigma^{\perp}(\mathbf{x},\omega) = \frac{|N(\mathbf{x})\cdot\omega||N(\mathbf{x}')\cdot\omega|}{||\mathbf{x}'-\mathbf{x}||^2}\mathrm{d}A(\mathbf{x}')\,. \tag{2.8}$$

We are now ready to write down the surface area formulation of Equation 2.5:

$$L(\mathbf{x}'\to\mathbf{x}'') = L_e(\mathbf{x}'\to\mathbf{x}'') + \int_{\mathcal{M}} f_S(\mathbf{x}\to\mathbf{x}'\to\mathbf{x}'')L(\mathbf{x}\to\mathbf{x}')G(\mathbf{x}\leftrightarrow\mathbf{x}')\mathrm{d}A(\mathbf{x})\,. \tag{2.9}$$

This is also called the *three-point form* of the rendering equation. Here, we used the *geometry factor* $G(\mathbf{x}\leftrightarrow\mathbf{x}')$,

$$G(\mathbf{x}\leftrightarrow\mathbf{x}') = V(\mathbf{x}\leftrightarrow\mathbf{x}')\frac{|N(\mathbf{x})\cdot\omega||N(\mathbf{x}')\cdot\omega|}{||\mathbf{x}'-\mathbf{x}||^2}\,. \tag{2.10}$$

It contains the familiar terms arising from the change of variables, as well as the *visibility function* $V(\mathbf{x}\leftrightarrow\mathbf{x}')$, which is a simple indicator function that returns 1 if $\mathbf{x}$ and $\mathbf{x}'$ are mutually visible and 0 otherwise.

2.3 The Measurement Equation

At the beginning of this chapter, we motivated light transport with the main purpose of rendering images. So far, we only described how to model light as it interacts with the scene. In order to produce images, we also need to be able to describe a virtual camera and how it interacts with light.

Similar to physical cameras, we can reason about virtual cameras in terms of an aperture that allows light to enter, as well as a sensor that measures the incoming radiance and turns it into pixel values. We can think of the camera sensor as being divided into M sensor elements arranged in a regular grid, each element associated with a pixel $j \in \{1, \dots, M\}$ in the image. Typically, M is large (on the order of 10^6).

The response of the sensor to light may vary with respect to the position and direction at which light strikes the sensor. We characterize the sensor response using the *importance*, denoted as $W_e(\mathbf{x}, \omega)$. Here, the point $\mathbf{x}$ lies on the aperture, which is modelled as a physical surface in the scene.

Typically, the sensor response also differs among the sensor elements, and we denote the sensor response associated with the j-th pixel as $W_e^{(j)}$. The relation between the response of the whole sensor and the response of the individual sensor elements is defined in terms of the *reconstruction filter* h_j as

$$W_e^{(j)}(\mathbf{x}, \omega) = h_j(\mathbf{x}, \omega) W_e(\mathbf{x}, \omega)\,. \tag{2.11}$$

We can think of each of the pixels in the image being the result of a *measurement* of the incident radiance by one of the sensor elements. We denote the value of the j-th pixel as the measurement I_j, defined as the inner product of the incident radiance and the sensor response,

$$I_j = \int_{\mathcal{M}} \int_{\mathcal{S}^2} W_e^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) \mathrm{d}\sigma^{\perp}(\mathbf{x}, \omega) \mathrm{d}A(\mathbf{x})\,. \tag{2.12}$$

This equation is called the *measurement equation*.

2.3.1 Path Integral Formulation

In both the (hemi)spherical and the surface area form of the rendering equation, the unknown quantity L appears on both sides. This recursive formulation is intuitive and some light transport methods can be expressed very naturally within this framework. We can think of this formulation as the "local" view of the light transport problem.

There also exists an alternative "global" view, originally introduced by Veach [Vea98], which allows expressing the measurement equation directly as a *path integral*. Unlike the local formulation, the path integral provides an explicit expression for the value of a measurement. Any unbiased rendering method can be expressed within this framework.

A full derivation of the path integral formulation is outside the scope of this thesis. We will review the most important concepts required for the rest of this thesis below and refer the reader to Veach [Vea98] and Jakob [Jak13] for full discussion of path space for surfaces and participating media.

The path integral can be derived by repeatedly expanding the surface area form of the measurement equation and rearranging the resulting terms into an infinite sum of integrals, each term integrating over a different number of surface points. This sum can be expressed in a unified integral,

$$I_j = \int_{\mathcal{P}} f_j(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) \,. \tag{2.13}$$

In the path integral formulation, the measurement is directly computed as an integral over *light transport paths* $\bar{\mathbf{x}}$. A transport path is a series of vertices

$$\bar{\mathbf{x}} = \mathbf{x}_0 \mathbf{x}_1 \dots \mathbf{x}_k \,, \tag{2.14}$$

where each of the vertices is a surface point $\mathbf{x}_i \in \mathcal{M}, i = 0, \dots, k$ and k is the path length.

All transport paths of length k form a space $\mathcal{P}_k$,

$$\mathcal{P}_k = \underbrace{\mathcal{M} \times \mathcal{M} \times \dots \times \mathcal{M}}_{k+1 \text{ times}} \,. \tag{2.15}$$

The union of these spaces forms *path space* $\mathcal{P}$, the space of all transport paths of arbitrary length:

$$\mathcal{P} = \bigcup_{k=1}^{\infty} \mathcal{P}_k \,. \tag{2.16}$$

The integrand in Equation 2.13 is referred to as the *measurement contribution function* $f_j(\bar{\mathbf{x}})$ and is defined as

$$\begin{aligned} f_j(\mathbf{x}_0 \dots \mathbf{x}_k) = L_e(\mathbf{x}_0 \to \mathbf{x}_1) W_e^{(j)}(\mathbf{x}_{k-1} \to \mathbf{x}_k) \\ G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \prod_{i=1}^{k-1} f_S(\mathbf{x}_{i-1} \to \mathbf{x}_i \to \mathbf{x}_{i+1}) G(\mathbf{x}_i \leftrightarrow \mathbf{x}_{i+1}) \,. \end{aligned} \tag{2.17}$$

We will also sometimes make use of the path contribution f to the entire sensor, not just one specific measurement, signaled by dropping the subscript j. It is defined similarly as the measurement contribution function in Equation 2.17, only replacing $W_e^{(j)}$ with the importance W_e.

CHAPTER 3

Solving the Light Transport Problem

Solutions to the light transport problem have been the subject of major research efforts over the past 30 years. Both the rendering equation and the measurement equation do not generally lend themselves to analytic solutions, and various numerical solution techniques have been proposed, starting with ray tracing [Whi80] and finite element methods such as radiosity [GTGB84].

In this chapter, we will review a few solution methods to surface light transport that are relevant to the work presented in this thesis.

3.1 Monte Carlo Methods

Rendering methods based on Monte Carlo integration were first introduced to graphics by Cook et al. [CPC84] in order to render distribution effects such as motion blur, depth of field and soft shadows. Kajiya [Kaj86] applied the Monte Carlo method to the rendering equation to render global illumination effects.

Consider the following definite integral of a function f over some domain Ω:

$$I = \int_\Omega f(x) \mathrm{d}\mu(x) \,. \tag{3.1}$$

Monte Carlo integration is a method to approximate the value of this integral by replacing it with a random variable with a specially crafted expected value. Let X be a random variable in Ω with probability density function (pdf) of $p_X(x)$. The expected value of a function g of X is

$$E\left[g(X)\right] = \int_\Omega g(x) p_X(x) \mathrm{d}\mu(x) \,. \tag{3.2}$$

We now define g in such a way that the expected value of $g(X)$ is the integral we intend to solve. Let

$$g(x) = \frac{f(x)}{p_X(x)} \tag{3.3}$$

with expected value

$$\begin{aligned} E\left[g(X)\right] &= \int_\Omega \frac{f(x)}{p_X(x)} p_X(x) \mathrm{d}\mu(x) && (3.4)\\ &= \int_\Omega f(x) \mathrm{d}\mu(x) && (3.5)\\ &= I\,. && (3.6) \end{aligned}$$

Estimating the expected value of a random variable is a well-known problem, and we can easily obtain an estimator of $E\left[g(X)\right]$: Let $X_1, \ldots, X_N$ be N independent realizations of X, distributed with density p_X. Then an estimate of I can be obtained with

$$I^{(N)} = \frac{1}{N} \sum_{i=1}^{N} g(X_i) = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p_X(X_i)}\,. \tag{3.7}$$

As $N \to \infty$, $I^{(N)}$ is guaranteed to converge to the expected value I by the law of large numbers, as long as $p_X(x) > 0$ whenever $f(x) \neq 0$.

In the context of rendering, we can apply Monte Carlo integration to obtain an estimate of the pixel measurement I_j. As long as we have a way of sampling paths $\bar{\mathbf{x}}$ with probability $p(\bar{\mathbf{x}})$ non-zero wherever $f_j(\bar{\mathbf{x}})$ non-zero, we can obtain an arbitrarily accurate approximation of I_j using the estimator in Equation 3.7.

3.1.1 Variance of the Monte Carlo Estimator

The Monte Carlo estimator given in Equation 3.7 is itself a random variable. Of special practical interest is the variance of this random variable: The lower the variance, the more accurate the estimate of the integral will be on average for a fixed number of samples.

The variance of a random variable is defined as

$$V\left[X\right] = E\left[X^2\right] - \left(E\left[X\right]\right)^2\,. \tag{3.8}$$

Inserting the Monte Carlo estimator $I^{(N)}$ into Equation 3.8 and rearranging the terms yields

$$V\left[I^{(N)}\right] = \frac{1}{N} V\left[\frac{f(X)}{p_X(X)}\right]\,. \tag{3.9}$$

This immediately shows the problem of Monte Carlo integration: The variance of the estimate $I^{(N)}$ only decreases linearly with respect to N, and therefore the standard deviation only decreases proportionally to $\sqrt{N}$. In other words, to decrease the expected integration error by a factor of two, we must increase N by a factor of four, a relatively poor convergence rate.

It is worth noting that the convergence rate of the Monte Carlo estimator is independent of the dimensionality of the integration problem, unlike other numerical integration techniques such as quadrature. This makes it suitable to integration problems in high-dimensional domains such as path space.

The variance of the Monte Carlo estimator greatly depends on the choice of sampling density. In general, the closer the sampling density approximates the integrand, the lower the variance; this is a technique called *importance sampling*. To demonstrate, we can compute the variance of the estimator using the "perfect" sampling density, which is proportional to the integrand. Consider $p^*(x) = c\, f(x)$. Then

$$V\left[\frac{f(X)}{p^*(X)}\right] = V\left[\frac{f(X)}{c\, f(X)}\right] = V\left[\frac{1}{c}\right] = 0\,. \tag{3.10}$$

In other words, the perfect sampling density leads to a variance of 0, and the exact value of I can be obtained taking only a single sample. Unfortunately, computing the correct proportionality factor c involves first solving the integral of interest – that is, to compute the solution of the integral using the perfect Monte Carlo estimator, we must first compute the solution of the integral. As such, we are not usually able to perform perfect importance sampling.

However, it is still possible to importance sample part of the target function. In particular, the measurement contribution function is a product of a number of terms, and although importance sampling their product may be difficult, it is still possible to importance sample the terms individually. This is also called *local importance sampling* and will be explored in the next section in more detail.

How effective the probability density of transport paths is at approximating the path contribution function is the main factor determining the efficiency of a rendering algorithm, and all of the rendering techniques presented in this chapter only differ in their choice of sampling scheme.

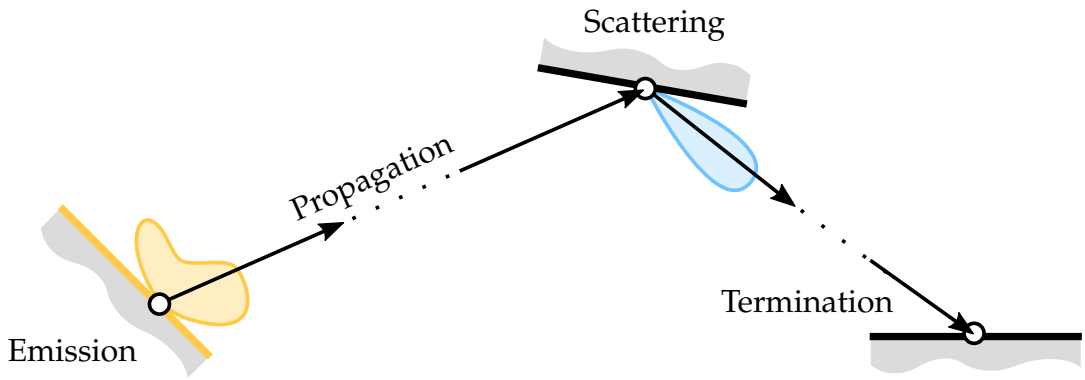


Figure 3.1: *Illustration of four different events on an example random walk*

3.2 Random Walks

Core to all Monte Carlo ray tracing methods is the sampling of transport paths. As mentioned at the end of the previous section, the probability density with which these transport paths are generated should ideally be proportional to the measurement contribution function of the path. As it is not generally possible in practice to jointly generate all vertices of a path with density proportional to the path contribution, paths are sampled incrementally instead, in a process referred to as a *random walk* through the scene.

In a random walk, we incrementally add more vertices to the path by importance sampling the different terms of the path contribution function in sequence. We describe these terms and how to sample them in the next few paragraphs. We also show an illustration of an example random walk in Figure 3.1.

Random walks can begin at either end of the transport path – that is, at a light source or at the camera. In order to use the same notation for both directions, we refer to paths as $\bar{\mathbf{y}} = \mathbf{y}_0 \dots \mathbf{y}_k$ in this section. Depending on the sampling direction, $\mathbf{y}_0$ can either refer to a vertex on an emitter or a vertex on the camera aperture. The discussion of importance sampling remains largely the same for both directions.

Camera Sampling To begin a random walk, we need to determine the location of the first vertex on the path, $\mathbf{y}_0$. When the random walk starts at the camera, a good choice is to importance sample the sensor response. In most cases, the sensor response is invariant with respect to points on the aperture, and $\mathbf{y}_0$ can be chosen uniformly on the camera aperture. To obtain the direction ω_0 leaving the first vertex, we need to importance sample the sensor response with respect to directions. We can either select

a single pixel estimate I_j to which the path should contribute, in which case we should importance sample its associated reconstruction filter [ESG06]; or we can importance sample the sensor response W_e over the entire image plane, allowing the traced path to contribute to all pixel estimates at once by splatting it to the image.

Emitter Sampling When the random walk starts on a light source, we should obtain $\mathbf{y}_0$ and ω_0 by importance sampling the emitted radiance. In the simplest case, the emitted radiance is piecewise constant with respect to positions and is completely diffuse, i.e. the directional part of the emitted radiance only depends on the angle between ω_0 and the surface normal, allowing for a straight-forward importance sampling scheme. However, some light sources may exhibit spatially varying emission (e.g. textured light sources), directionally varying emission (e.g. IES lights [Ill91]) or both (e.g. environment maps [Deb98]) and require special treatment. We refer the reader to Pharr and Humphreys [PH10] for a complete discussion.

Propagation Given a vertex on the path $\mathbf{y}_i$ and the outgoing direction ω_i at that vertex, the next vertex on the path can be easily obtained using the *ray cast function*, $\mathbf{y}_{i+1} = \mathbf{x}_{\mathcal{M}}(\mathbf{y}_i, \omega_i)$. The ray cast function returns the surface point closest to $\mathbf{y}_i$ along the direction ω_i. It is defined in terms of a utility function $d_{\mathcal{M}}(\mathbf{x}, \omega)$, which returns the smallest distance $d > 0$ such that $\mathbf{x} + \omega\, d \in \mathcal{M}$. If no such distance exists, $d_{\mathcal{M}}(\mathbf{x}, \omega)$ returns ∞. The ray cast function is then

$$\mathbf{x}_{\mathcal{M}}(\mathbf{x}, \omega) = \mathbf{x} + \omega\, d_{\mathcal{M}}(\mathbf{x}, \omega)\,. \tag{3.11}$$

Scattering In order to obtain the outgoing direction ω_i at surface vertex $\mathbf{y}_i$ with $i > 0$, we should importance sample the terms appearing in the scattered radiance, i.e. the product of BSDF and foreshortening term, $f_S(\mathbf{x}, -\omega_{i-1}, \omega_i)|N(\mathbf{x}) \cdot \omega_i|$. For some materials, such as perfectly diffuse reflectors, simple sampling routines exist [DHM$^+$01] that can perfectly importance sample this product, but in the general case, the sampling distribution only approximately matches the integrand of the scattered radiance at that point. Indeed, certain BSDFs such as microfacet models [WMLT07] can significantly contribute to the variance in the rendered image.

Termination Random walks terminate naturally if the path leaves the scene – that is, the path stops at vertex $\mathbf{y}_i$ if $||\mathbf{y}_i - \mathbf{y}_{i-1}||$ is not finite. Additionally, the rendering algorithm itself sometimes imposes a maximum path length, and the random walk is al-

lowed to terminate after a certain number of scattering events. However, in closed environments, the path may not leave the scene, and the random walk never terminates if no maximum path length is imposed. This is problematic, since non-terminating random walks cause the surrounding rendering algorithm to grind to a halt and never produce an image. A simple way of achieving finite path lengths without imposing an artificial maximum path length is to employ *Russian roulette* [AK90], which probabilistically terminates the random walk. At every vertex, we choose an arbitrary continuation probability w_i. With probability w_i, the random walk continues after vertex $\mathbf{y}_i$ and its pdf is decreased by a factor of w_i; with probability $1 - w_i$ however, the random walk is terminated at $\mathbf{y}_i$. Using Russian roulette, paths of any length still have a positive probability of being sampled, but the random walk always terminates in finite time if the continuation probabilities are chosen carefully. Decreasing the pdf by the appropriate factor also ensures that the Monte Carlo estimate still converges to the correct solution when Russian roulette is used.

Path probability density The path $\bar{\mathbf{y}}$ sampled by the random walk has an associated pdf, $p(\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k)$, which can be viewed as a joint probability density function over all vertices. Because the path is sampled incrementally, it reduces to the product

$$p(\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_k) = p(\mathbf{y}_0) \prod_{i=1}^{k} p(\mathbf{y}_i | \mathbf{y}_{i-1}, \mathbf{y}_{i-2}, \dots, \mathbf{y}_0) \,. \tag{3.12}$$

For our purposes, the pdf of the first vertex is usually proportional to either the emitted radiance or the sensor response, whereas the pdfs of the subsequent vertices are proportional to some of the terms in the geometry factor and the BSDF. We will make use of this fact when we review different path sampling techniques in the following sections. Note how the pdfs of all vertices but the first are conditioned on the path that precedes them. This is because the sampling distribution at a vertex usually adapts to the preceding vertices; for example, most BSDF sampling strategies take the incoming direction into account when sampling an outgoing direction, and the resulting sampling distribution for a vertex $\mathbf{y}_i$ depends on the two vertices that precede it. More sophisticated sampling strategies that are conditioned on longer prefixes of the path are also possible [GKH+13].

3.3 Path Tracing

Arguably one of the simplest Monte Carlo methods for global illumination is *path tracing*, introduced by Kajiya [Kaj86]. Path tracing is also sometimes called unidirectional path tracing due to its mode of operation, which we illustrate in Figure 3.2 (a).

In path tracing, transport paths are sampled using a random walk starting at the camera, resulting in a camera path $\bar{\mathbf{z}} = \mathbf{z}_0\mathbf{z}_1 \dots \mathbf{z}_k$ with $\mathbf{z}_0$ on the camera aperture. From this single camera path, we can construct k full transport paths of different lengths with $\mathbf{z}_i \dots \mathbf{z}_1\mathbf{z}_0, i = 1, \dots, k$ – that is, any emitting surface encountered on the camera path contributes to the image. The contribution F of the sampled camera path is then simply

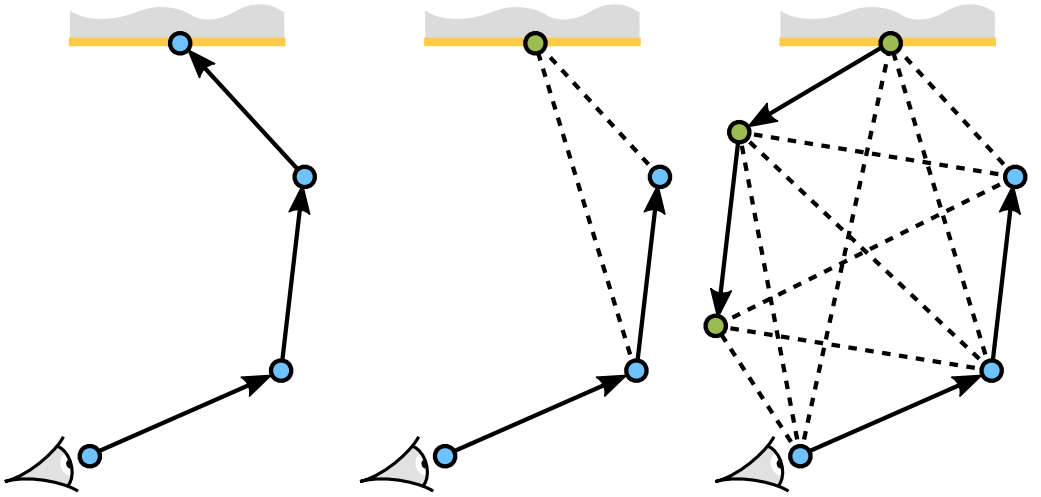
$$F = \sum_{i=1}^{k} \frac{f(\mathbf{z}_i \dots \mathbf{z}_1\mathbf{z}_0)}{p(\mathbf{z}_0\mathbf{z}_1 \dots \mathbf{z}_i)}. \tag{3.13}$$

Many terms in Equation 3.13 appear in both the numerator and the denominator and cancel out, in particular the geometry factors. The remaining terms are simple to compute, and the sum can be be computed incrementally as the path is being traced, making path tracing conceptually simple and easy to implement. Computing the sample contribution incrementally also removes the need to keep the full path in memory at once and enables interesting optimization strategies targeted at tracing a large number of rays simultaneously [LKA13, ENSB13].

3.3.1 Next Event Estimation

Tracing paths unidirectionally from the camera using locally importance sampled random walks includes all factors of the measurement contribution in the sampling density except the emitted radiance L_e. Unfortunately, not importance sampling this term can cause excessive variance in most scenes: We expect L_e to be zero for the majority of the surfaces in the scene, and very large for a small subset of $\mathcal{M}$ (the light sources). Unidirectional path tracing relies on BSDF sampling to find surfaces with large emission, which is inadequate for most scenes (Figure 3.3 (a)). Something as simple as an outdoor scene lit by a far-away sun can result in unrecognizably noisy images when naive path tracing is used.

To remedy this issue, we can employ a technique called *next event estimation*. In next event estimation, we sample a "one vertex path" $\bar{\mathbf{y}} = \mathbf{y}_0$ starting on a light source



(a) Path Tracing **(b)** Next Event Estimation **(c)** Bidirectional Path Tracing

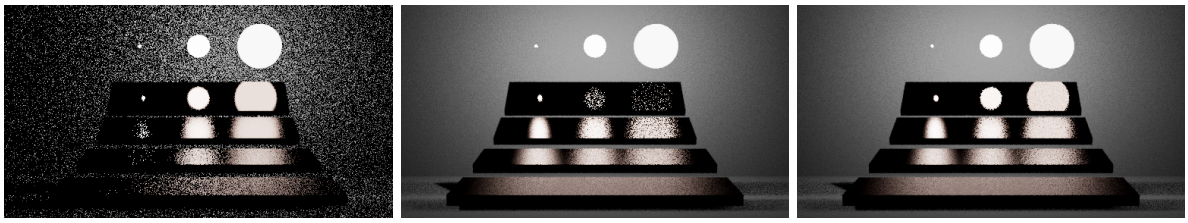
Figure 3.2: *Illustration of the different path sampling strategies in path tracing, next event estimation and bidirectional path tracing*

in addition to the camera path used in naive path tracing. We then obtain the transport paths $\mathbf{y}_0\mathbf{z}_i \dots \mathbf{z}_0, i = 0, \dots, k$ by connecting the one vertex on the emitter with all vertices on the camera path. Figure 3.2 (b) illustrates this sampling procedure for an example light path. The resulting contribution F can then be written as

$$F = \sum_{i=0}^{k} \frac{f(\mathbf{y}_0\mathbf{z}_i \dots \mathbf{z}_0)}{p(\mathbf{z}_0 \dots \mathbf{z}_i)p(\mathbf{y}_0)}. \tag{3.14}$$

Different to the naive path tracing algorithm, the emitted radiance that appears inside f in the numerator is likely to be matched with a proportional factor $p(\mathbf{y}_0)$ in the denominator. This can greatly improve variance in scenes where the light sources are small and difficult to hit with BSDF sampling alone. However, unlike the naive path tracing algorithm, the BSDF at $\mathbf{z}_i$ as well as the geometry factor between the last and second last vertex on the full transport path are no longer importance sampled. Because of this, next event estimation can perform poorly in some cases, especially when the BSDF at $\mathbf{z}_i$ is very peaked and L_e is not (Figure 3.3 (b)).

It is worth noting that this section describes a simplified version of next event estimation; in practice, next event estimation is normally only employed with a few additional optimizations. In particular, paths of length one (i.e. with only two vertices) are normally handled with unidirectional sampling, since variance from these paths is low. Additionally, one would normally sample a new vertex on the light source for



(a) Path Tracing **(b)** Next Event Estimation **(c)** MIS of (a) and (b)

Figure 3.3: *Renderings using path tracing (a), next event estimation (b) and MIS of the two (c). Both (a) and (b) show sampling deficiencies for certain material-emitter combinations. MIS is robust over all combinations. Modeled after a scene by Eric Veach.*

each connection with the camera path to reduce correlation. For this discussion, we will stick with the simplified version, as it is better comparable to fully bidirectional sampling methods.

3.3.2 Multiple Importance Sampling

Naive path tracing and next event estimation are two different sampling techniques that produce transport paths with different probability densities. Either technique in isolation works well in some cases and poorly in others. Rather than using only one technique or the other, a better idea is to use both at the same time and weight them in a special way such that we will prefer a technique if it samples a path much better than the other technique and vice-versa.

This is a technique referred to as *multiple importance sampling* (MIS) and can be formalized using a multi-sample estimator [VG95]. If we have N sampling techniques at our disposal, each with sampling density $p_i, i = 1, \ldots, N$ respectively, and we draw one sample x_i from each technique, then the multi-sample estimator of a function f in its simplest form is

$$F = \sum_{i=1}^{N} w_i(x_i) \frac{f(x_i)}{p_i(x_i)} . \tag{3.15}$$

Here, w_i are the *MIS weights* of the techniques. As long as the MIS weights form a partition of unity and $w_i(x_i) > 0$ only when $p_i(x_i) > 0$, F will converge to the correct result. We are free to choose any MIS weights that fulfill these conditions. Veach proposes the *balance heuristic*, a provably good choice of MIS weights, defined as

$$w_i(x) = \frac{p_i(x)}{\sum_{j=1}^{N} p_j(x)} . \tag{3.16}$$

(a) Path tracing, 32 paths/pixel **(b)** Bidirectional path tracing, 32 paths/pixel

Figure 3.4: *Comparisons of renderings produced by path tracing with next event estimation and MIS (a) and bidirectional path tracing (b) at an equal number of paths per pixel. Bidirectional path tracing has many more sampling techniques at its disposal, leading to lower variance compared to path tracing. Modeled after a scene by Eric Veach.*

The intuition behind the balance heuristic is that a large probability density is usually a good indicator when a technique does well at generating a sample. Therefore, the balance heuristic will give a sampling technique a large weight if its sampling density is large for a given sample compared to the other techniques.

We can apply multiple importance sampling to path tracing with the purpose of obtaining a robust combination of next event estimation and naive path tracing (Figure 3.3 (c)). A path $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$ can be sampled in two different ways: With unidirectional path tracing, $\bar{\mathbf{x}}_{\text{PT}} = \mathbf{z}_k \dots \mathbf{z}_0$, and with next event estimation, $\bar{\mathbf{x}}_{\text{NEE}} = \mathbf{y}_0\mathbf{z}_{k-1} \dots \mathbf{z}_0$. This allows us to insert the probability densities of either technique into Equation 3.16 to directly compute MIS weights with which to combine the techniques.

3.4 Bidirectional Path Tracing

In the previous section, we discussed two possible sampling techniques for transport paths of a given length: Full unidirectional path tracing, or next event estimation. *Bidirectional path tracing* (BDPT) [LW93, VG94, LW96] supplements these with a whole fam-

ily of sampling techniques: Paths of length k can be constructed with $k+2$ different sampling techniques in BDPT, as opposed to the 2 sampling techniques discussed previously.

To do this, BDPT obtains an *emitter subpath* $\bar{\mathbf{y}} = \mathbf{y}_0 \dots \mathbf{y}_{N_E}$ and a *camera subpath* $\bar{\mathbf{z}} = \mathbf{z}_0 \dots \mathbf{z}_{N_C}$ using two random walks started from the light sources and the camera, respectively. BDPT then constructs full transport paths $\bar{\mathbf{x}}_{s,t} = \mathbf{y}_0 \dots \mathbf{y}_{s-1}\mathbf{z}_{t-1} \dots \mathbf{z}_0$ by connecting prefixes of the subpaths with each other, which we illustrate in Figure 3.2 (c). Here N_E and N_C are the lengths of the emitter- and camera subpaths, and s and t are the number of vertices on the emitter- and camera subpaths used for the connection.

We can see that a path $\bar{\mathbf{x}}$ of length k can be generated by $k+2$ different sampling techniques in BDPT, depending on how many vertices are taken from the emitter subpath and how many from the camera subpath. Similar to before, it is a good idea to combine the different sampling techniques using multiple importance sampling, since all of them have sampling deficiencies for certain types of paths. However, having $k+2$ sampling techniques at our disposal rather than just 2 increases the chances of there being at least one sampling technique that samples a given path well. The resulting estimate for a pair of subpaths is then

$$F = \sum_{s=0}^{N_E} \sum_{t=0}^{N_C} w_{s,t}(\bar{\mathbf{x}}_{s,t}) \frac{f(\bar{\mathbf{x}}_{s,t})}{p_s(\bar{\mathbf{x}}_{s,t})} \,. \tag{3.17}$$

Here, we used the path probability $p_s(\bar{\mathbf{x}})$, which is the probability density of generating $\bar{\mathbf{x}}$ using the s-th technique. It is defined as

$$p_s(\mathbf{x}_0 \dots \mathbf{x}_k) = p(\mathbf{x}_0 \dots \mathbf{x}_{s-1}) p(\mathbf{x}_k \dots \mathbf{x}_s) \,. \tag{3.18}$$

That is, it is the product of probability densities of generating the two subpaths individually. p_s also allows to easily define the MIS weights $w_{s,t}$ computed with the balance heuristic:

$$w_{s,t}(\bar{\mathbf{x}}) = \frac{p_s(\bar{\mathbf{x}})}{\sum_{i=0}^{s+t} p_i(\bar{\mathbf{x}})} \,. \tag{3.19}$$

Other choices of weighting heuristic are possible [Vea98], but we will stick with the balance heuristic throughout this thesis for simplicity.

Figure 3.4 demonstrates the advantage of bidirectional path tracing over path tracing in a rendering of an indoor scene at equal sample count. Figure 3.5 shows the individual sampling techniques comprising Figure 3.4 (b), both with and without MIS.

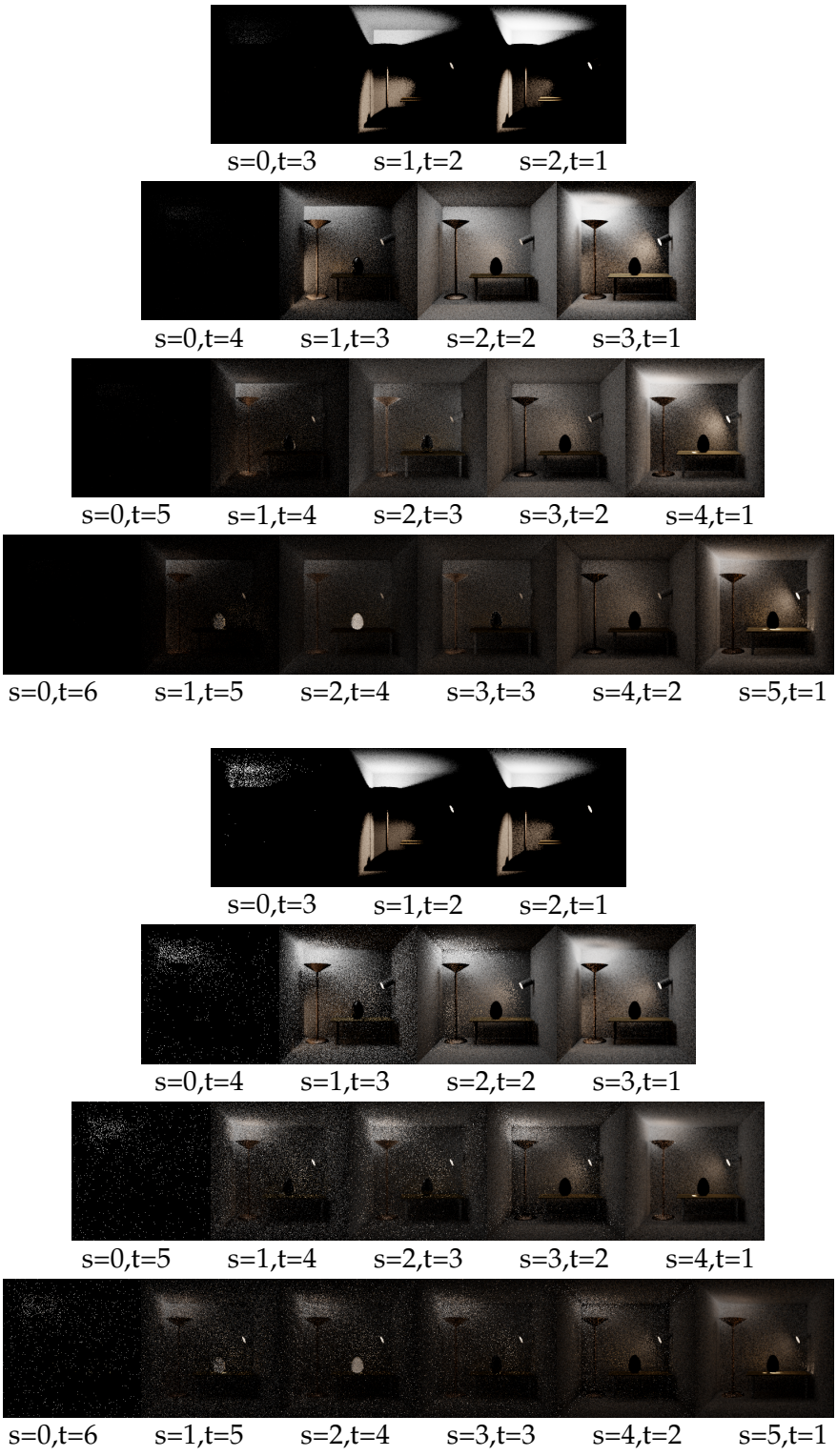


Figure 3.5: *Top: The individual sampling techniques used to generate image Figure 3.4 (b), weighted with multiple importance sampling. Bottom: The same sampling techniques, but this time* without *multiple importance sampling. Note how all techniques have deficiencies for certain types of paths.*

3.5 Markov Chain Monte Carlo Methods

An interesting deviation from the Monte Carlo methods we've seen so far is the field of *Markov Chain Monte Carlo* (MCMC) methods. Unlike the Monte Carlo methods discussed previously, MCMC methods make use of random samples that are not statistically independent.

A sequence of random variables $X_1, X_2, X_3, \ldots$ is said to form a *Markov chain* if the probability of the realization $X_{i+1} = x_{i+1}$ only depends on the state of the previous random variable in the sequence, X_i. More formally,

$$Pr(X_{i+1} = x_{i+1} | X_1 = x_1, X_2 = x_2, \ldots, X_i = x_i) = Pr(X_{i+1} = x_{i+1} | X_i = x_i) \tag{3.20}$$

$$= q(x_i \to x_{i+1}) \,. \tag{3.21}$$

We will refer to the function $q(x_i \to x_{i+1})$ as the *transition distribution*. If we wanted to simulate a Markov process, that is, to obtain a particular realization of the Markov chain, the transition distribution would inform us how to obtain the next state x_{i+1} given the current state x_i.

Under certain conditions, the sequence of states $x_1, x_2, x_3, \ldots$ will converge to a *stationary distribution* that is uniquely defined by the transition distribution. The core idea of MCMC is to construct a Markov chain that has a stationary distribution proportional to an arbitrarily chosen target function. The distribution of states visited by the Markov chain will then in the limit be proportional to the target function - in other words, the states of such a Markov chain importance sample the target function.

Remember that a Markov chain (and its stationary distribution, if it exists) is defined by its transition distribution. The Metropolis-Hastings algorithm [MRR+53, Has70] describes how to turn a transition distribution that does not result in the desired stationary distribution into one that does. Given a *proposal distribution* $T(\mathbf{x} \to \mathbf{y})$ and a *target distribution* π, the Metropolis-Hastings rule obtains the next state $\mathbf{y}$ from the current state $\mathbf{x}$ in the following steps:

1. Sample the *proposal state* $\mathbf{x}'$ from the proposal distribution $T(\mathbf{x} \to \mathbf{x}')$
2. Compute $r(\mathbf{x} \to \mathbf{x}') := min\left\{1, \frac{\pi(\mathbf{x}')T(\mathbf{x}' \to \mathbf{x})}{\pi(\mathbf{x})T(\mathbf{x} \to \mathbf{x}')}\right\}$
3. $\mathbf{y} = \begin{cases} \mathbf{x}', & \text{with probability } r \\ \mathbf{x}, & \text{otherwise} \end{cases}$

The quantity $r(\mathbf{x} \to \mathbf{x}')$ is known as the *acceptance probability* and is what reshapes the

stationary distribution of the Markov chain to the desired target distribution. With probability r, the proposal state $\mathbf{x}'$ is accepted as the new state of the Markov chain, and with probability $1 - r$ it is rejected and the current state is repeated.

Let us now review how to use such a Markov chain to integrate a target function. Assume we wanted to integrate a function $f(x)$ over some domain Ω. We can use the Metropolis-Hastings algorithm to define a Markov chain with state space Ω and stationary distribution f. Running this Markov chain for N steps yields the states $x_1, \ldots, x_N$ visited by the chain. These are realizations of a random variable, and inserting into the Monte Carlo estimator yields

$$\int_\Omega f(x) \mathrm{d}\mu(x) \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \,. \tag{3.22}$$

In order to compute an estimate of the integral using samples produced by a Markov chain, we need to compute the currently unknown quantity $p(x_i)$. The Metropolis-Hastings algorithm guarantees us that the states are distributed proportionally to the target function f, i.e. $p(x_i) = c\, f(x_i)$. The proportionality factor c is required to make sure p is a valid probability density (i.e. it integrates to 1) and is defined as

$$c = \frac{1}{\int_\Omega f(x) \mathrm{d}\mu(x)} \,. \tag{3.23}$$

Unfortunately, computing this proportionality factor requires us to solve the desired integral before we can use Metropolis-Hastings to solve the desired integral, making it a pointless endeavour. Indeed, straight-forward applications of MCMC to integration problems only work if the probability density cancels out (when computing expected values, for example).

Fortunately, rendering an image requires us to solve not just one, but many measurement integrals I_j, and the majority of the integrand is shared among them. Recall the path integral

$$I_j = \int_{\mathcal{P}} f_j(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) \,. \tag{3.24}$$

Using the definition of the importance and the measurement contribution function, this can be rewritten as

$$I_j = \int_{\mathcal{P}} h_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) \,. \tag{3.25}$$

We can see that all measurements share the same evaluation of the path contribution to the entire image. Let us now define a Markov chain with state space $\mathcal{P}$ and target

distribution $f(\bar{\mathbf{x}})$. The Markov chain constributes to the estimates of *all* measurement integrals at the same time; given states $\bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_N$ visited by the Markov chain, the resulting estimates will be

$$I_j \approx \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}_i)}{p(\bar{\mathbf{x}}_i)} = \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}_i)}{c\, f(\bar{\mathbf{x}}_i)} = \frac{1}{N} \sum_{i=1}^{N} \frac{h_j(\bar{\mathbf{x}})}{c}\,. \tag{3.26}$$

Here, the proportionality factor is

$$c = \frac{1}{\int_{\mathcal{P}} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}})}\,. \tag{3.27}$$

This Markov chain contributes to M pixel estimates at once. While it still requires computing a proportionality factor, this factor is shared among all estimates, making it feasible to approximate the proportionality factor using a secondary Monte Carlo estimator (e.g. bidirectional path tracing) with a large number of samples. The estimation of the proportionality factor is amortized over all M pixels; usually, M is large (on the order of 10^6), and the computation time spent on the proportionality factor is insignificant compared to the time spent running the Markov chain.

This results in a conceptually simple algorithm that applies MCMC sampling to the light transport problem, proceeding as follows

1. Estimate $c^{-1} = \int_{\mathcal{P}} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}})$ using a secondary estimator, such as BDPT.
2. Choose $\bar{\mathbf{x}}_1$ arbitrarily
3. For $i = 1, \ldots, N$:
   a) Sample $\bar{\mathbf{y}}$ from $T(\bar{\mathbf{x}}_i \to \bar{\mathbf{y}})$
   b) $\bar{\mathbf{x}}_{i+1} = \begin{cases} \bar{\mathbf{y}}, & \text{with probability } \frac{f(\bar{\mathbf{y}}) T(\bar{\mathbf{y}} \to \bar{\mathbf{x}}_i)}{f(\bar{\mathbf{x}}_i) T(\bar{\mathbf{x}}_i \to \bar{\mathbf{y}})} \\ \bar{\mathbf{x}}_i, & \text{otherwise} \end{cases}$
   c) Splat c^{-1}/N to the image at $\bar{\mathbf{x}}_{i+1}$

An interesting problem is the choice of initial state $\bar{\mathbf{x}}_1$. We follow Veach and Guibas [VG97], who propose an initialization technique that chooses the initial state while computing the normalization factor, simultaneously ensuring unbiasedness of the MCMC estimator. The reader is referred to their work for full discussion.

The MCMC method described above operates on a scalar target distribution $f(\bar{\mathbf{x}})$; however, we are more likely to encounter an RGB-valued contribution function in practice. A common solution is to use a scalar importance function $f^*(\bar{\mathbf{x}})$ that attempts to match all color channels as the target distribution of the Markov chain. Frequently, the luminance of $f(\bar{\mathbf{x}})$ is used, although other choices are possible [HH10]. In the MCMC

method outlined above, the quantity splatted to the image plane then only needs to be multiplied by an additional $f(\bar{\mathbf{x}})/f^*(\bar{\mathbf{x}})$ to support color information as well.

The remaining issue is the choice of proposal distribution $T(\bar{\mathbf{x}}_i \to \bar{\mathbf{y}})$. Similar to importance sampling, MCMC methods benefit from proposal distributions that approximate the target distribution well, and finding good proposal distributions has been the main focus of research on MCMC for the light transport problem. Indeed, most MCMC rendering algorithms only differ in how they generate proposals.

3.6 Metropolis Light Transport

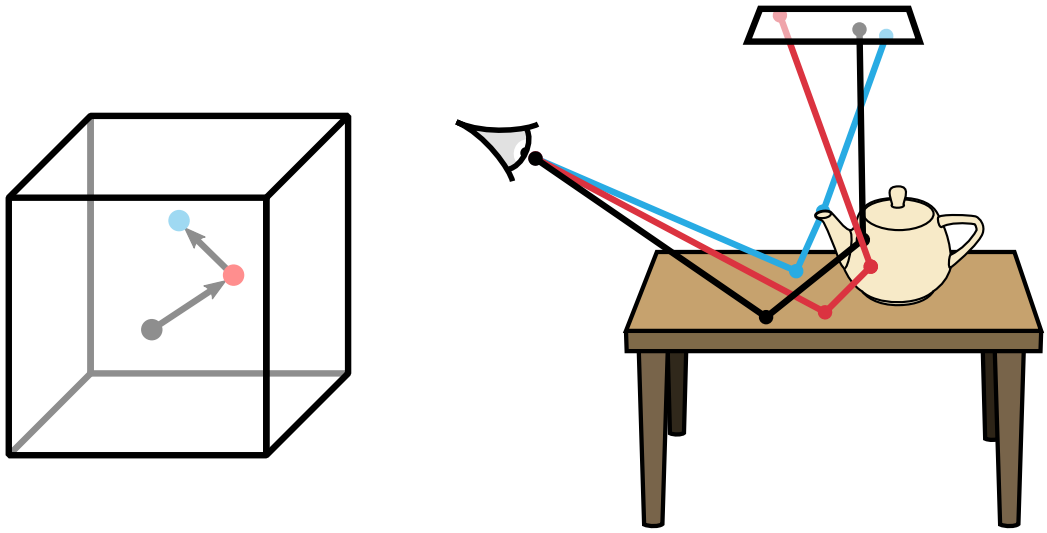
Markov Chain Monte Carlo was first applied to the light transport problem by Veach [VG97] in the *Metropolis Light Transport* (MLT) algorithm, which is an MCMC method that operates directly in path space.

MLT contains several ways of obtaining proposals, which are classified into *perturbations* that keep the structure of the light path fixed while changing the vertex positions by a small amount, and *mutations* that perform large changes to the path and are allowed to change its structure.

Intuitively, we expect the path contribution function to be at least somewhat coherent, such that similar paths have similar contribution. Perturbations locally explore regions of path space, producing many similar paths with (hopefully) similar contribution that are likely to be accepted. On the other hand, high contribution regions tend to form "islands" in path space surrounded by low-contribution regions, and perturbations may have trouble jumping from one island to the next, since they only move in small steps through path space. It is the task of mutations to propose large changes to the path in order to fully explore path space. Such mutations usually suffer from low acceptance rates, but are required in order to account for all light transport.

3.7 Primary Sample Space Metropolis Light Transport

A significantly simpler version of MLT, referred to as *Primary Sample Space Metropolis Light Transport* (PSSMLT), was proposed by Kelemen et al. [KSKAC02]. The main idea behind PSSMLT is that unbiased rendering algorithms can be understood in terms of a deterministic *path sample function* $S(\mathbf{u})$ that maps points $\mathbf{u}$ in the space of random numbers, the *primary sample space* $\mathcal{U} = [0,1]^\infty$, to path space (Figure 3.6). The path



(a) Primary Sample Space **(b)** Path Space

Figure 3.6: *Primary Sample Space MLT runs a Markov chain in primary sample space (a) to obtain samples in path space (b)*

integral can then be written as an integral over primary sample space via a change of variables

$$\int_{\mathcal{P}} f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}) = \int_{\mathcal{U}} f(S(\mathbf{u})) \left| \frac{\mathrm{d}S^{-1}(\mathbf{u})}{\mathrm{d}\mathbf{u}} \right| \mathrm{d}\mathbf{u} \tag{3.28}$$

$$= \int_{\mathcal{U}} f(S(\mathbf{u})) \frac{1}{p(S(\mathbf{u}))} \mathrm{d}\mathbf{u}\,. \tag{3.29}$$

Notably, the Jacobian determinant arising from the change of variables is simply the probability density of generating a path from a set of random numbers. A Monte Carlo estimator of Equation 3.29 is given by

$$I_j \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f_j(S(\mathbf{u}_i))}{p(S(\mathbf{u}_i))}\,, \tag{3.30}$$

where the $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ are uniformly distributed in $\mathcal{U}$. Note that Equation 3.30 is the same Monte Carlo estimator introduced in Section 3.1, only this time derived explicitly in terms of a mapping from random numbers to paths.

The idea of PSSMLT is then to define a Markov chain operating in primary sample space rather than path space. First, we define the relevant quantities in primary sample

space in terms of their path space counterparts:

$$\hat{p}(\mathbf{u}) := p(S(\mathbf{u})), \tag{3.31}$$

$$\hat{f}(\mathbf{u}) := f(S(\mathbf{u})), \tag{3.32}$$

$$\hat{f}^*(\mathbf{u}) := f^*(S(\mathbf{u})), \tag{3.33}$$

$$\hat{C}(\mathbf{u}) := \frac{\hat{f}(\mathbf{u})}{\hat{p}(\mathbf{u})}, \tag{3.34}$$

$$\hat{C}^*(\mathbf{u}) := \frac{\hat{f}^*(\mathbf{u})}{\hat{p}(\mathbf{u})}. \tag{3.35}$$

PSSMLT intends to sample the integrand of Equation 3.29, labelled $\hat{C}(\mathbf{u})$, using the scalar importance function $\hat{C}^*(\mathbf{u})$ as target distribution.

This has two key advantages: Firstly, the state space $\mathcal{U}$ has significantly simpler structure than path space, allowing the use of a very simple proposal distribution (e.g. a Gaussian distribution). The proposal distributions in this space are usually symmetric, i.e. $T(\mathbf{u} \to \mathbf{u}') = T(\mathbf{u}' \to \mathbf{u})$, and cancel out when computing the acceptance probability. A second advantage is that the integrand $\hat{C}$ is significantly flatter than the path contribution function. This is because the path sample function is designed to importance sample parts of the path contribution function, and many of the terms appearing in Equation 3.34 cancel out. We can also think of S as "warping" the path contribution function via importance sampling, stretching out peaks in path space over a larger region in primary sample space.

Kelemen et al. propose two different ways of obtaining proposals in primary sample space: Using a *small step* perturbation that samples a proposal within a small region around the current state; and a *large step* mutation that completely replaces the current state with numbers sampled uniformly random in $[0, 1]$. The latter mutation ensures that the Markov chain is able to reach any part of primary sample space with positive probability.

The resulting algorithm is remarkably simple and, assuming a symmetric proposal distribution, proceeds as follows:

1. Estimate $c^{-1} = \int_{\mathcal{U}} \hat{C}(\mathbf{u}) \mathrm{d}\mathbf{u}$ using a secondary Monte Carlo estimator
2. Choose $\mathbf{u}_1$ arbitrarily
3. For $i = 1, \ldots, N$:
   a) $\mathbf{v} = \begin{cases} \textsc{UniformRandom}(), & \text{with probability } p_{\text{large}} \\ \textsc{Perturb}(\mathbf{u}_i), & \text{otherwise} \end{cases}$

b) $\mathbf{u}_{i+1} = \begin{cases} \mathbf{v}, & \text{with probability } \frac{\hat{C}(\mathbf{v})}{\hat{C}(\mathbf{u}_i)} \\ \mathbf{u}_i, & \text{otherwise} \end{cases}$

c) Splat c^{-1}/N to the image,

where $p_{\text{large}} \in [0, 1]$ is the large step probability, a parameter that determines the ratio of large steps to small steps.

PSSMLT as a rendering algorithm makes very few assumptions about the underlying path sample function S and views it as a black blox that turns a vector of random numbers into a light path. This allows it to be employed on top of a variety of existing rendering methods, including path tracing, bidirectional path tracing and progressive photon mapping [HJ11].

Some special care is required when dealing with bidirectional path tracing, however, since it generates potentially many paths from a single sample and combines them with MIS. Using the MIS weighted sum of contributions from all transport paths leads to the undesirable situation where the MCMC algorithm will prefer longer paths over shorter paths, since they lead to more connections and therefore a higher contribution. However, longer paths are also much more expensive to evaluate, increasing the average cost per sample when an MCMC sampler is used.

Kelemen et al. propose to use the *maximum heuristic* when combining the different path contributions, which will simply select the transport path with the largest probability density and discard all other paths. The contribution of two BDPT subpaths is then simply the contribution of the connection with the largest probability density. This has the advantage that longer paths do not intrinsically contribute more to the image than shorter paths.

3.7.1 Rippling Effects

The "black box" view of the path sample function makes PSSMLT a versatile algorithm that can be employed in conjunction with many different sampling methods. However, having limited knowledge about how dimensions of $\mathbf{u}$ map to paths can interfere with the behaviour of the MCMC sampler. In particular, this will manifest as small steps in primary sample space leading to large changes to the sampled path, which can hinder local exploration of primary sample space. This can happen due to a variety of reasons, which we will discuss in Chapter 4 as well as Section 5.2 and Section 5.3 in more detail.

One of the simpler causes of unintended large changes to the path are due to "ripple"

effects. All sampling techniques discussed so far utilize random walks to generate part of the path. Performing a random walk implicitly associates dimensions of $\mathbf{u}$ with vertices on the path, and small steps perform well as long as this association remains consistent over a large number of transitions. However, what can happen is that a small change to elements of $\mathbf{u}$ cause the random walk to use a different amount of random numbers to sample a particular vertex (e.g. due to a change in BSDF). This leads to a ripple effect across all subsequent vertices on the random walk as the association of random numbers to vertices is shifted, and causes a large change to the path.

If there is an upper bound to how many random numbers the random walk will consume to produce a single vertex, then a simple remedy is to make the association of random numbers to vertices explicit and allocate a fixed number of dimensions in $\mathbf{u}$ for each vertex. We will denote the random numbers reserved for the i-th vertex as $\mathbf{u}_i$. Such an explicit association has a number of benefits, including notational simplicity, and will be assumed for the rest of this thesis.

Avoiding ripple effects is even more important in bidirectional methods. A naive implementation of BDPT first uses random numbers $\mathbf{u}_0, \ldots, \mathbf{u}_{s-1}$ to sample an emitter subpath with s vertices and $\mathbf{u}_s, \ldots, \mathbf{u}_{s+t-1}$ to sample a camera subpath with t vertices. If s changes even by just one vertex, the association of random numbers to vertices is changed for the entire camera subpath, equivalent to replacing it by a completely different path.

Kelemen et al. propose to use the elements at odd- and even numbered indices in the random number vector for different subpaths, avoiding this problem. For the rest of this thesis, we assume that the subpaths are sampled using the naive approach for notational simplicity, but opt to use two different random number vectors entirely in our implementation, one for each subpath. This approach is also followed by other rendering systems, such as Mitsuba [Jak10].

3.8 Multiplexed Metropolis Light Transport

Multiplexed Metropolis Light Transport (MMLT), introduced by Hachisuka et al. [HKD14], combines the concepts of MCMC sampling with multiple importance sampling.

We reviewed multiple importance sampling in Section 3.3.2 with the purpose of optimally combining multiple sampling techniques. It's not clear how to combine MIS with PSSMLT, since the latter intrinsically only supports a single sampling technique,

which is S. Even though algorithms such as BDPT offer up to $k+2$ sampling techniques for paths of length k, this issue was sidestepped in PSSMLT with the maximum heuristic, which simply selects the path with the highest probability density, effectively hiding the existence of multiple sampling techniques from the Markov chain.

MMLT introduces a way of exposing the different sampling techniques available in BDPT as well as their MIS weights, so that the sampling technique can be selected by the Markov chain. Bidirectional path tracing offers $k+2$ sampling techniques for a path of length k, which we will denote as $S_i(\mathbf{u}), i = 0, \ldots, k+1$. MMLT uses the *extended state space* $(t, \mathbf{u})$, which specifies both the position in primary sample space as well as the sampling technique $S_t(\mathbf{u})$ that should be used to transform it into a path. The parameter t is then changed alongside $\mathbf{u}$ as the Markov chain proceeds from one state to the next.

In order to incorporate MIS, MMLT uses the target distribution $\hat{w}_t(\mathbf{u})\hat{C}_t^*(\mathbf{u})$, which is the product of the MIS weight and the path contribution of the t-th technique,

$$\hat{C}_t^*(\mathbf{u}) = f^*(S_t(\mathbf{u}))\,. \tag{3.36}$$

Because the MIS weight is part of the target distribution, the contribution of sampling techniques inappropriate for the current path is heavily downweighted, and the Markov chain will automatically try to move away from such sampling techniques.

The main advantage of MMLT over PSSMLT is computational efficiency. Recall that PSSMLT needs to consider all possible connections that could be formed between the emitter- and camera subpath. Even when the maximum heuristic is used, the probability densities of all generated paths need to be computed before one subpath can be selected. In contrast, MMLT makes the choice of sampling technique part of the state of the Markov chain. To evaluate a sample, MMLT only ever needs to consider a single connection between the two subpaths.

For efficiency reasons, MMLT splits up the integrand and runs a separate Markov chain for each path length. This involves estimating a separate normalization factor and initial state for each Markov chain, which is done in a similar manner to the unbiased initialization introduced by Veach et al. [VG97]. Each Markov chain then operates in a state space $(t, \mathbf{u}_k) \in \{0, \ldots, k+1\} \times [0,1]^{O(k)}$, where k is the length of paths handled by the Markov chain. Rather than introducing a separate discrete perturbation for the technique index t, Hachisuka et al. propose to simply use one additional random number $\mathbf{u}_{k,t}$ to determine the technique as $t = \lfloor (k+2)\mathbf{u}_{k,t} \rfloor$. This allows the technique index to be perturbed and mutated using the original PSSMLT proposal distributions.

Hachisuka et al. note that a customized proposal sampling strategy for the technique index might prove beneficial. In this thesis, we show how to improve the performance of MMLT by taking the technique index into special consideration during large steps (Section 4.3) and small steps (Section 5.2).

CHAPTER 4

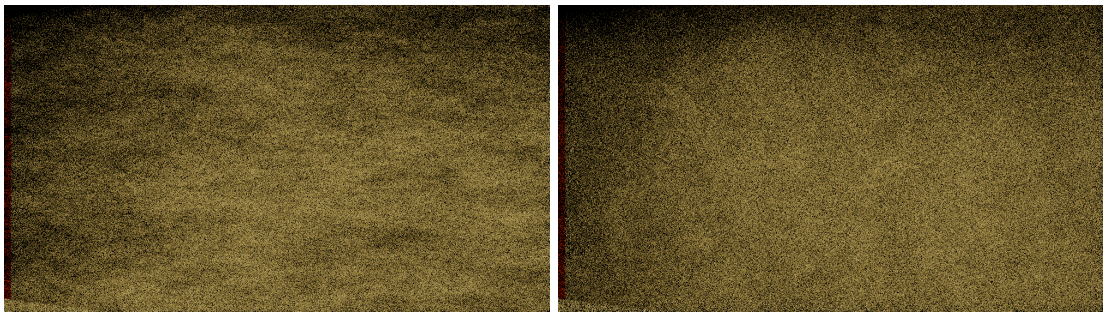
Annotated Primary Sample Space

In Section 3.7, we reviewed PSSMLT as an MCMC method in primary sample space that views the underlying path sampling method as a black box that turns a vector of random numbers into a transport path. This allows it to be employed on top of a variety of existing rendering methods, including path tracing, bidirectional path tracing and progressive photon mapping. Unfortunately, these rendering methods are usually not implemented with PSSMLT in mind, and their use of random numbers can heavily interfere with the behaviour of the Markov chain.

In this chapter, we relax the black box perspective a little and explore two applications where exposing slightly more information about the underlying path sampling algorithm and its use of the random number vector can be leveraged to generate proposals more robustly. This requires some cooperation between the path sampling function and the Markov chain, but our proposed solutions are conceptually simple and easy to implement. We conclude this chapter with a discussion of large steps in MMLT, and propose an alternative large step based on bidirectional connections.

4.1 Resolution-Aware Proposals

One of the drawbacks of MCMC methods for light transport is insufficient control over the distribution of samples on the image plane. This is in contrast to methods such as path tracing or bidirectional path tracing, which can trivially stratify samples over the image. Veach [VG97] proposes a special lens subpath mutation whose main purpose is to stratify samples over the image plane. Energy-Redistribution Path Tracing [CTE05] is an MCMC method that explores path space starting with a set of initial paths, which can be carefully chosen to be stratified over the image plane. Although effective, these techniques are generally difficult to reconcile with the primary sample space perspec-



(a) Default PSSMLT small steps **(b)** Resolution-aware proposals

Figure 4.1: *A diffuse wall rendered with PSSMLT. Using an isotropic proposal distribution in primary sample space can lead to a stretched proposal distribution on the image plane when the image is not square. This is made apparent by the resulting noise pattern on the image (a). Scaling the proposal distribution of small steps to account for the aspect ratio of the image achieves a more uniform and visually pleasing distribution of the noise (b).*

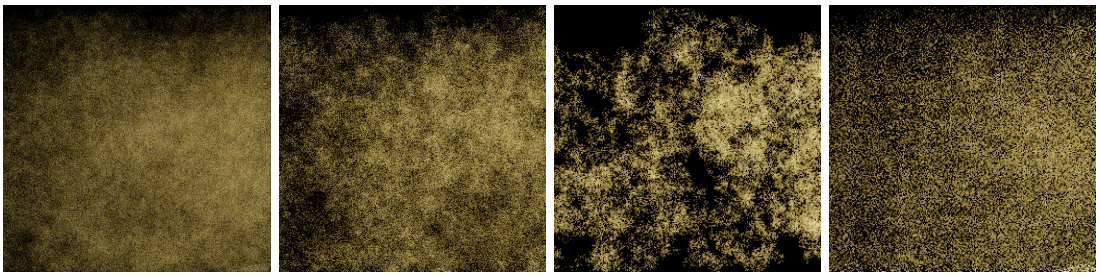
tive, and as such stratification can prove difficult in this context.

However, we can still improve the sampling distribution on the image plane using knowledge about the resolution of the image. In both PSSMLT and MMLT, launching a camera subpath involves selecting a position on the image plane, either explicitly by selecting a pixel measurement or implicitly by sampling the sensor response. In the majority of cases, the random walk will use two elements of the random number vector and multiply them with the resolution of the image in order to sample a point on the image plane.

Because PSSMLT and MMLT perform perturbations in primary sample space rather than directly on the image plane, the distribution of perturbations on the image plane is highly resolution dependent. For example, if an aspect ratio of 16:9 is used, isotropic perturbations in primary sample space will on average move approximately twice as far on the horizontal axis than on the vertical axis on the image plane, which leads to an anisotropic, "stretched" distribution of the noise (Figure 4.1 (a)).

To remedy this issue, we propose to explicitly reserve two dimensions of the random number vector for sampling a position on the image plane. The proposal distribution of small steps for those two dimensions is then adjusted to rescale the horizontal axis by the aspect ratio. The resulting noise distribution in the image is much more uniform and visually pleasing (Figure 4.1 (b)).

There are other interesting ways this information could be used. For example, because



(a) 1024x1024, PSSMLT | **(b)** 512x512, PSSMLT | **(c)** 256x256, PSSMLT | **(d)** 256x256 Resolution-aware proposals

Figure 4.2: *A diffuse wall rendered with PSSMLT at different image resolutions. The average step size of small steps in PSSMLT decreases as the resolution is decreased (a-c), leading to increased correlation across pixels. Scaling the proposal distribution in primary sample space to account for the image resolution greatly improves the noise distribution at smaller resolutions (d)*

of the way camera paths are sampled, decreasing the resolution of the image will at the same time decrease the average step size, in pixels, of small step perturbations on the image plane. This leads to the odd situation where smaller resolution images produced with PSSMLT or MMLT tend to look worse than larger-resolution images of the same scene, simply because the Markov chain performs smaller steps on the image plane on average and tends to get stuck on a single pixel much longer than on the equivalent larger-resolution image. Figure 4.2 demonstrates the resulting noise difference.

If the proposal distribution is already adjusted to account for the aspect ratio, it can additionally be scaled in order to achieve a fixed average step size in pixels, regardless of the image resolution. In some scenes, such a technique can significantly improve noise in smaller-resolution images, which we demonstrate in Figure 4.2 (c). However, such an adjustment will at the same time increase the average step size in *path space* as the resolution is decreased. In scenes with difficult glossy-to-glossy transport, we found that the average acceptance ratio drops as the resolution decreases, leading to slower convergence rates when such a resolution adjustment is used. To remain robust across a variety of scenes, we therefore only correct for aspect ratio and not resolution in our implementation, but note that for some scenes, the latter adjustment can result in significant noise improvements.

Another application of such an adjustment could be in the sampling of vertices on light sources. For example, a rectangular ceiling light could show significant stretching along one dimension, leading to a similar anisotropic distribution of small steps.

However, we expect a step size adjustment in these domains to have a less prominent impact than on the image plane, and could show robustness issues when dealing with sliver-like light sources, for example.

4.2 Constrained Discrete Choices

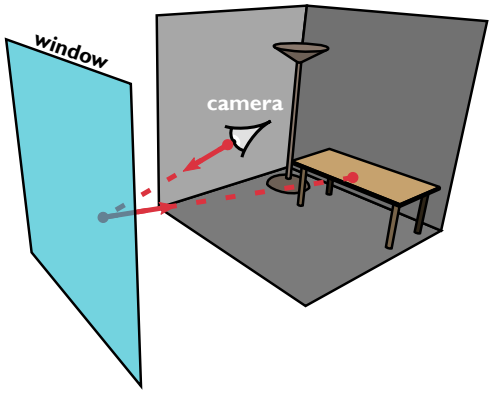
Discrete choices, i.e. mapping a random number to one of a discrete set of outcomes, is not an unusual occurrence during random walks. For example, deciding whether to reflect or refract on a dielectric surface, or whether to terminate a subpath using Russian roulette, both transform an input random number into a discrete decision. We observe that changing the outcome of a discrete choice usually leads to a large change to the path: For example, changing the decision of whether to reflect or refract on a dielectric surface will completely change the location of all subsequent vertices on the path.

Small step perturbations in PSSMLT or MMLT are built on the assumption that a small change in primary sample space will lead to a small change in path space. However, we find that this stands at odds with discrete decisions: Changing an element of the random number vector associated with a discrete decision will either lead to no change at all, or a large change in the path and its contribution which is likely to be rejected.

We therefore propose to never perturb random numbers associated with discrete decisions during small steps, since we do not expect a change in a discrete decision to lead to a small change in the path contribution function. In order to do this, we annotate the elements of the random number vector with information about whether it is used to drive a discrete decision or a continuous mapping. When the path sampling algorithm draws a random number, we instruct it to also pass a binary flag describing whether the number is used to make a discrete decision or not. This flag is stored alongside the random number vector. Elements which are flagged as discrete are left unchanged during small steps.

The effects of constrained discrete choices are usually subtle, but in some scenes with prominent light paths on dielectric surfaces it improves rendering performance significantly (Figure 4.3).

(a) BDPT **(b)** PSSMLT **(c)** PSSMLT with constrained discrete choices



Scene setup

Figure 4.3: *The reflection of a room seen in a window looking out over darkness, rendered with BDPT, PSSMLT and PSSMLT using our proposed modification at equal sample count (64 paths/pixel). Rays reflecting off of the window have a very low probability of being sampled, and BDPT (a) has difficulty rendering this scene. PSSMLT (b) performs better, but because it changes the random number associated with the reflection decision, many proposals generated with small steps change the discrete decision on the window and transmit instead, leading to a low average acceptance ratio. Fixing discrete decisions during small steps (c) preserves low probability paths that reflect off of the window.*

4.3 An Alternative Large Step Mutation

The purpose of large step mutations in PSSMLT is to ensure that all points in primary sample space can be reached with positive probability, a property of the Markov chain called *ergodicity*. They are required to avoid getting stuck in small regions of primary sample space; however, because they generate a completely new random number vec-

tor while ignoring the current sample, they suffer from low acceptance rates in scenes with difficult light transport. This is especially the case in Multiplexed MLT, which we will examine in this section.

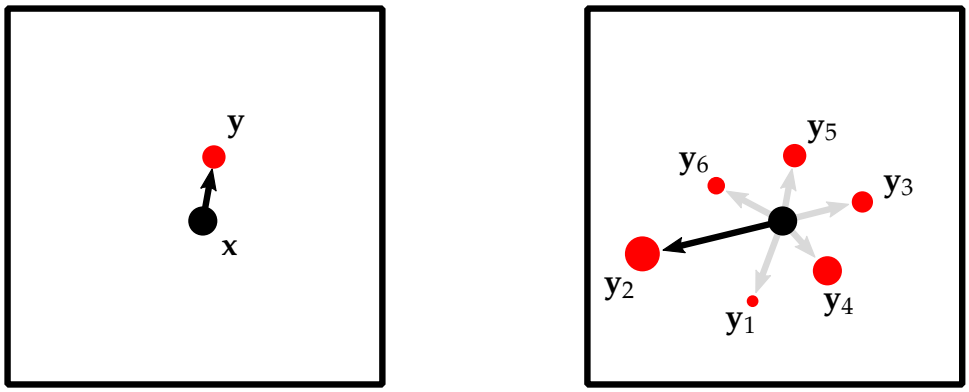
In Multiplexed MLT, one element of the random number vector is used to determine the sampling technique that should be used to generate the path. Since individual Markov chains in MMLT sample paths of a fixed length, choosing the sampling technique at the same time determines the lengths of the camera- and emitter subpaths.

This can lead to undesirable behaviour during large steps. A large step amounts to replacing the random number vector, including the technique index, with uniformly distributed random numbers; this means that without even looking at the subpaths we are about to sample, we already dictate their exact lengths and the sampling technique that should be used to construct the full path. For longer path lengths and scenes with complex visibility, it is generally difficult to trace subpaths of an exact length with mutually visible end vertices. The resulting full path can also be heavily downweighted by MIS because an inadequate sampling technique was chosen to generate it. As such, large steps in MMLT tend to generate proposals with low contribution and suffer from low acceptance rates.

In this section, we propose an alternative large step based on bidirectional connections that can greatly improve the average acceptance rate in complex scenes. The idea is that instead of choosing the sampling technique before tracing the subpaths, we first trace the subpaths, look at all $k+2$ possible connections of length k between them and then choose the sampling technique by randomly selecting one of the connections with probability proportional to its contribution.

To properly reason about such an approach, we briefly review Multiple-Try Metropolis (MTM) [LLW00]. The idea of the MTM method is to generate several proposals, rather than one, and importance sample one of the proposals based on its contribution. Figure 4.4 illustrates this idea. In its simplest form, MTM proceeds as follows:

1. Draw N independent trial proposals, $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ from $T(\mathbf{x} \to \cdot\,)$.
2. Compute $w(\mathbf{y}_j \to \mathbf{x}) := \pi(\mathbf{y}_j)T(\mathbf{y}_j \to \mathbf{x}), j = 1, \ldots, N$
3. Choose $\mathbf{y}$ among the N trial proposals with probability proportional to $w(\mathbf{y}_j \to \mathbf{x})$
4. Generate $N-1$ variates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1}$ from $T(\mathbf{y} \to \cdot)$. Set $\mathbf{x}_N = \mathbf{x}$.
5. Compute $w(\mathbf{x}_j \to \mathbf{y}), j = 1, \ldots, N$
6. Compute $r_g(\mathbf{x} \to \mathbf{y}) := \min\left\{1, \frac{w(\mathbf{y}_1 \to \mathbf{x}) + w(\mathbf{y}_2 \to \mathbf{x}) + \ldots + w(\mathbf{y}_N \to \mathbf{x})}{w(\mathbf{x}_1 \to \mathbf{y}) + w(\mathbf{x}_2 \to \mathbf{y}) + \ldots + w(\mathbf{x}_N \to \mathbf{y})}\right\}$



(a) Metropolis **(b)** Multiple-Try Metropolis

Figure 4.4: *The Metropolis-Hastings algorithm (a) generates a single proposal and either rejects or accepts it. Multiple-Try Metropolis (b) samples multiple trial proposals and selects a single proposal from them. Here, the sizes of the red circles represent the contribution of each proposal. The black arrow points to the selected proposal.*

7. Accept $\mathbf{y}$ with probability $r_g(\mathbf{x} \to \mathbf{y})$

$r_g(\mathbf{x} \to \mathbf{y})$ is called the *generalized M-H ratio*. For $N = 1$ it reduces to the standard Metropolis-Hastings acceptance probability.

The main benefit of the MTM method is that it allows considering multiple proposals in one step, increasing the chances of producing a high-contribution sample that is likely to be accepted. However, if generating the proposals or evaluating $w(\mathbf{x} \to \mathbf{y})$ is the bottleneck of the algorithm, then MTM is generally not beneficial: Although we only produce N proposals, we have to generate and evaluate $2N - 1$ samples to compute the generalized M-H ratio. In an MMLT context, evaluating $w(\mathbf{x} \to \mathbf{y})$ is unfortunately the most expensive part of the algorithm, since it involves tracing and connecting two subpaths, so we do not expect standard MTM to provide much benefit here.

Segovia et al. [SIP07] previously applied MTM in the context of MLT; however, their method relies on a packet tracing approach to amortize the evaluation of trial proposals and variates. This requires restructuring the entire rendering algorithm to support tracing multiple rays in a packet, increasing the implementation complexity.

We observe that if the trial proposals are allowed to be correlated, then we can cheaply generate up to $k + 2$ trial proposals from one camera- and emitter subpath. To do this, we first trace two subpaths up to length k, and in a manner very similar to bidirectional path tracing, we connect all possible prefixes of the two subpaths that produce a full

transport path of length k.

Although standard MTM requires the trial set to be independent, a modified algorithm named Multiple Correlated-Try Metropolis (MCTM) [CL07] allows the trial set to be correlated while still computing the correct acceptance ratio. MCTM modifies the MTM algorithm in the following ways: The N trials are drawn jointly from the proposal distribution $\tilde{T}(\mathbf{x} \to \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N)$ instead of independently from $T(\mathbf{x} \to \mathbf{y}_i)$; and the $N-1$ variates are drawn jointly from the conditional distribution $\tilde{T}(\mathbf{y} \to \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1} \mid \mathbf{x}_N = \mathbf{x})$. The rest of the algorithm stays identical.

Jointly sampling all trials from a joint distribution allows us to introduce correlation between the sampled trials. In MCTM, it is assumed that the marginal proposal distribution is equal to the original proposal distribution used in MTM, i.e.

$$\int \int \ldots \int \tilde{T}(\mathbf{x} \to \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) \mathrm{d}\mathbf{y}_1 \ldots \mathrm{d}\mathbf{y}_{i-1} \mathrm{d}\mathbf{y}_{i+1} \ldots \mathrm{d}\mathbf{y}_N = T(\mathbf{x} \to \mathbf{y}_i)\,. \tag{4.1}$$

In our case, $T(\mathbf{x} \to \mathbf{y}_i)$ is simply 1 everywhere, since we are concerned with large steps. We are free to choose any joint proposal distribution, as long as we can show that it reduces to 1 if all but one trial are integrated out. Not all choices of joint proposal distribution are useful; in general, the structure we introduce to the distribution of the trials should be beneficial somehow, for example by stratifying the trials or increasing computational efficiency. In our case, we want all trials to share the same subpaths and only wish to allow the connecting vertices to be different, since this way we only need to trace one pair of subpaths, rather than N of them.

We now show how generating multiple connections from two subpaths can be formulated as a joint proposal distribution. First, we write the state of the Markov chain $\mathbf{x}$ in its extended state space form, $\mathbf{x} = (\mathbf{x}^t, \mathbf{x}^{\mathbf{u}})$, where $\mathbf{x}^t$ is the technique index and $\mathbf{x}^{\mathbf{u}}$ is the random number vector. We then use the joint distribution

$$\tilde{T}(\mathbf{x} \to \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N) = \prod_{i=1}^{N} \frac{1}{k+2}\, \delta(\mathbf{y}_i^{\mathbf{u}} - \mathbf{y}_N^{\mathbf{u}})\,. \tag{4.2}$$

Here, δ is the Dirac delta function. Such a proposal distribution assigns zero probability to trial sets in which not all trial proposals share the same random number vector: In those cases, $\mathbf{y}_i^{\mathbf{u}} - \mathbf{y}_N^{\mathbf{u}} \neq 0$ for some i, and $\delta(\mathbf{y}_i^{\mathbf{u}} - \mathbf{y}_N^{\mathbf{u}}) = 0$. This compresses the state space of the trial set into a plane in which all trial proposals share the same random number vector, generating the same subpaths. Only the choice of sampling technique – that is, the choice of connecting vertices – is allowed to be different. Note that the technique index does not appear in this proposal distribution; in fact, this distribution is constant with respect to the sampling techniques chosen for the trial proposals. This

means that the technique index is distributed uniformly in $0, \ldots, k+1$, which also explains the normalization factor $1/(k+2)$. In a similar vein, this proposal distribution takes on the same value for all trial sets which share the same random number vector, independent of the actual random number vector chosen. This means that the random number vector shared across the trial set is distributed uniformly in $\mathcal{U}$.

We can easily show that such a proposal distribution fulfills the property assumed in MCTM: In this case, every integral in Equation 4.1 integrates over the extended state space $\{0, 1, \ldots, k+1\} \times \mathcal{U}$. Each integral collapses at least one of the Dirac deltas and sums over the technique index, resulting in a constant factor of $k+2$, which cancels with one of the $1/(k+2)$ factors in Equation 4.2. The resulting marginal distribution is 1 everywhere, which is equal to the original large step proposal distribution.

The conditional distribution is defined similarly, with all variates required to use the same random number vector as $\mathbf{x}$:

$$\tilde{T}(\mathbf{y} \to \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1} \,|\, \mathbf{x}_N = \mathbf{x}) = \prod_{i=1}^{N-1} \frac{1}{k+2} \, \delta(\mathbf{x}_i^{\mathbf{u}} - \mathbf{x}_N^{\mathbf{u}}) \,. \tag{4.3}$$

Sampling from these distributions is remarkably simple. We know that all of the trial proposals share the same random number vector, which is distributed uniformly in $\mathcal{U}$. Therefore, to jointly draw $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N$ trial proposals from $\tilde{T}(\mathbf{x} \to \cdot\,)$, we first trace a camera- and emitter subpath using uniformly generated random numbers. For each of the $\mathbf{y}_j$, we then uniformly generate $\mathbf{y}_j^t$, i.e. we uniformly choose a sampling technique and connect the corresponding vertices on the camera- and emitter subpath. All of the $\mathbf{y}_j$ use the same subpaths, generated once for the whole set.

Drawing the variates $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{N-1}$ from $\tilde{T}(\mathbf{y} \to \cdot \,|\, \mathbf{x}_N = \mathbf{x})$ is even simpler, since the random number vector is already given by the current state $\mathbf{x}$ and the two subpaths have already been partially traced. We simply extend the subpaths until they are either of length k or escape the scene and draw the variates by uniformly picking a sampling technique for each, in the same manner as for the $\mathbf{y}_j$.

This is very close to the bidirectional mutation that we want. Unfortunately, because of the way the proposals are drawn, the sampling techniques are picked uniformly, and we are not guaranteed that each sampling technique appears exactly once (which is what we would like).

Fortunately this is only a cosmetic problem. Consider the generalized M-H ratio:

$$r_g(\mathbf{x} \to \mathbf{y}) := \min \left\{ 1, \frac{w(\mathbf{y}_1 \to \mathbf{x}) + w(\mathbf{y}_2 \to \mathbf{x}) + \ldots + w(\mathbf{y}_N \to \mathbf{x})}{w(\mathbf{x}_1 \to \mathbf{y}) + w(\mathbf{x}_2 \to \mathbf{y}) + \ldots + w(\mathbf{x}_N \to \mathbf{y})} \right\} . \tag{4.4}$$

Because the random number vectors are shared among trials and proposals, the weights w can only take on one of $k+2$ different values – one for each sampling technique. That is,

$$r_g(\mathbf{x} \to \mathbf{y}) := \min\left\{1, \frac{s_0\, w((0, \mathbf{y^u}) \to \mathbf{x}) + \ldots + s_{k+1}\, w((k+1, \mathbf{y^u}) \to \mathbf{x})}{t_0\, w((0, \mathbf{x^u}) \to \mathbf{y}) + \ldots + t_{k+1}\, w((k+1, \mathbf{x^u}) \to \mathbf{y})}\right\}, \tag{4.5}$$

where the s_i and t_i count how many times the i-th sampling technique is selected for the proposals and the variates, respectively. It holds that $s_i \geq 0, t_i \geq 0$ and $\sum s_i = N$, $\sum t_i = N$.

Now we divide both numerator and denominator by N, replacing the s_i with s_i/N and similarly t_i with t_i/N:

$$r_g(\mathbf{x} \to \mathbf{y}) := \min\left\{1, \frac{(s_0/N)\, w((0, \mathbf{y^u}) \to \mathbf{x}) + \ldots + (s_{k+1}/N)\, w((k+1, \mathbf{y^u}) \to \mathbf{x})}{(t_0/N)\, w((0, \mathbf{x^u}) \to \mathbf{y}) + \ldots + (t_{k+1}/N)\, w((k+1, \mathbf{x^u}) \to \mathbf{y})}\right\}. \tag{4.6}$$

Because the sampling techniques are chosen uniformly, we know that the probability of the i-th technique being chosen for a sample is $1/(k+2)$. We can view the ratio s_i/N as a Monte Carlo estimator of $1/(k+2)$, and similarly for t_i/N. Since we know the exact value that this estimator will converge to as we increase the number of proposals, this allows us to directly draw from the limiting distribution as if we had generated an infinite number of proposals and variates. Each s_i/N and t_i/N is replaced with $1/(k+2)$; however, all of these terms cancel. We are left with

$$r_g(\mathbf{x} \to \mathbf{y}) := \min\left\{1, \frac{w((0, \mathbf{y^u}) \to \mathbf{x}) + \ldots + w((k+1, \mathbf{y^u}) \to \mathbf{x})}{w((0, \mathbf{x^u}) \to \mathbf{y}) + \ldots + w((k+1, \mathbf{x^u}) \to \mathbf{y})}\right\}. \tag{4.7}$$

That is, each sampling technique appears precisely once, which is what we wanted. We will refer to this mutation as a Multiple Correlated-Try Large Step (MCTLS).

4.3.1 Analysis

We will now have a closer look at the computational overhead of MCTLS compared to MMLT large steps in order to evaluate whether such an approach brings any benefit. To perform one MMLT large step, we need to trace two subpaths of combined length k, requiring k ray tracing operations. In contrast, MCTLS has to trace two subpaths of up to length k each and evaluate all $k+2$ combinations between them, requiring $3k+2$ ray tracing operations. Additionally, evaluating the variates requires extending the subpaths of the current state to length k and evaluating all $k+2$ combinations between

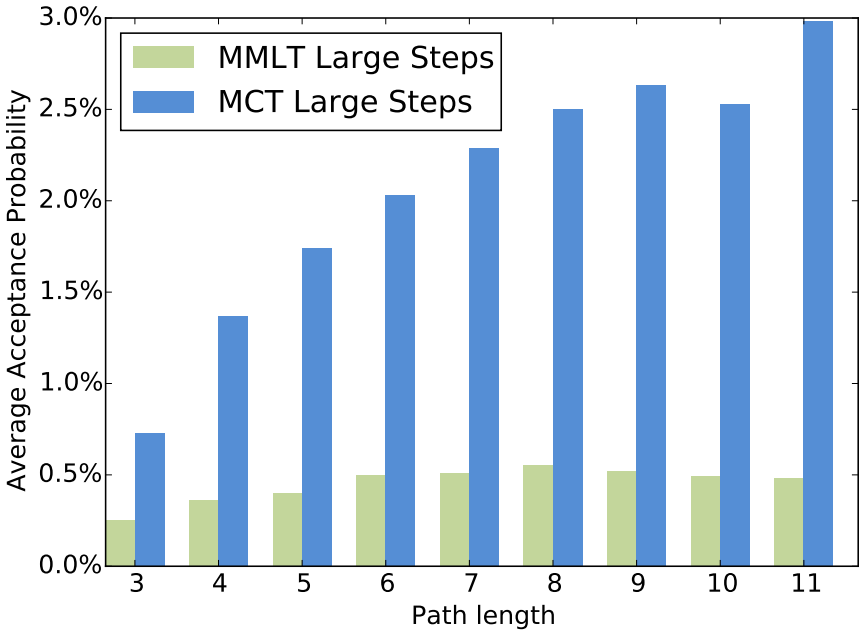
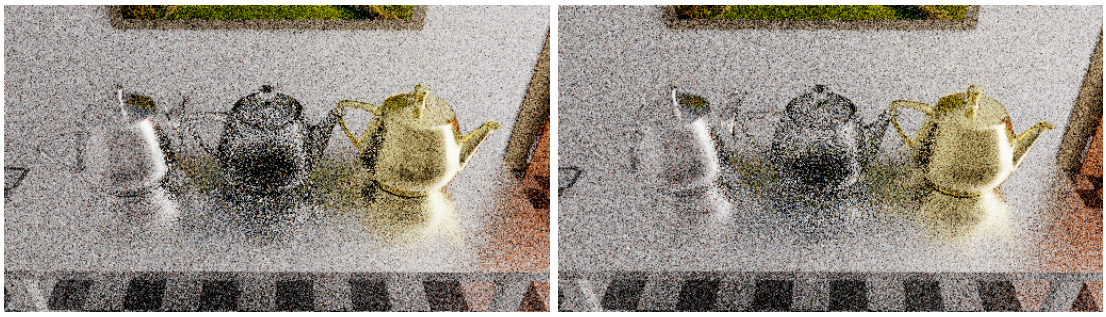


Figure 4.5: *Average acceptance probability of MMLT large steps and our proposed Multiple Correlated-Try (MCT) large steps for the scene in Figure 4.6, plotted over different path lengths. MMLT large steps have consistently low acceptance rates over all path lengths, whereas the acceptance rates of MCT large steps increase as the path length increases. This is because MCT large steps can cheaply generate more proposals for longer paths.*

them, requiring another $2k + 2$ ray tracing operations. Assuming that ray tracing operations dominate the computational cost, we therefore expect MCTLS mutations to be roughly five times more expensive than MMLT large steps on average. On the other hand, one MCTLS mutation yields up to $k + 2$ proposals, whereas an MMLT large step only results in one.

To evaluate the potential benefit of MCTLS over MMLT large steps, we measured the average acceptance probability of MMLT large steps and MCTLS mutations in a scene with complex visibility (Figure 4.6). We employed MMLT with either large steps or MCTLS to draw proposals; no small steps were used. We plot the measured average acceptance probability over different path lengths in Figure 4.5 for both mutations.

As expected, MMLT large steps are unlikely to result in proposals that are accepted, and their average acceptance probability is consistently low over all path lengths. Proposals generated with MCTLS are rejected similarly often for short paths, but their average acceptance probability increases with the path length. At path length 11, proposals generated with MCTLS are up to five times more likely to be accepted than



(a) MMLT large steps **(b)** MCTLS mutations

Figure 4.6: *An interior scene with complex visibility, rendered with purely MMLT large steps (a) and our proposed MCTLS mutation (b), at equal render time. No small steps were used. Once the increased render time is factored in, MCTLS no longer hold an advantage over MMLT large steps in this scene. Modeled after a scene by Eric Veach and Toshiya Hachisuka.*

MMLT large step proposals.

Unfortunately, the improved acceptance probability of MCTLS is rougly matched by its increased computational cost. For the same scene used before, we now show equal-time renderings produced by either technique in Figure 4.6. Both images are similarly noisy, which suggests that once the increased render time is taken into account, the advantage of MCTLS is no longer clear.

Because the benefit of MCTLS in terms of acceptance probability increases with the path length, whereas its computational overhead is a constant factor, it is conceivable that a combination of MMLT large steps for short paths and MCTLS for long paths could perform better than either technique in isolation. Such an approach requires further evaluation.

CHAPTER 5

Inverse Path Mappings

In Section 3.7, we described PSSMLT in terms of an opaque path sample function $S(\mathbf{u})$ that transforms points in primary sample space to paths. In this chapter, we are interested in the reverse process; that is, mapping a path back into the random numbers that produced it. Such an inverse mapping has interesting applications in the context of PSSMLT and MMLT.

In the following sections, we first briefly review random walks and give an explicit description in terms of pseudo-code. We then discuss how to construct the inverse of a random walk based on its explicit description, and how this can be used to invert the path sample function for path tracing and bidirectional path tracing. Finally, we conclude the section with two new perturbation strategies for PSSMLT and MMLT that make use of the inverse path sampling function.

5.1 Inverse Random Walks

Random walks and inverse random walks are tightly related, and it's not possible to describe one without the other. We looked at random walks in detail in Section 3.2; we give an explicit pseudo-code implementation based on that description in Algorithm 1. It takes three arguments: The random number vector $\mathbf{u}$, the maximum path length k, and an *adjoint* flag that denotes whether the path starts at the camera or on a light source. The first vertex is denoted as the *root vertex* and is sampled from either the camera or a light source, depending on the direction of path construction. The function will then repeatedly cast a ray and sample an outgoing direction at the next vertex until the maximum path length is reached or the path leaves the scene.

To perform an *inverse random walk* – that is, to convert a path constructed via random walk back to random numbers – we have to perform the inverse of every step of Al-

Algorithm 1: RANDOMWALK($\mathbf{u}$, k, adjoint)

if adjoint **then** $\mathbf{x}_0$, $\omega_0 \leftarrow$ SAMPLE $W_e(\mathbf{u}_0)$;
;
else $\mathbf{x}_0$, $\omega_0 \leftarrow$ SAMPLE $L_e(\mathbf{u}_0)$;
;
$i \leftarrow 0$;
while ISFINITE$(\mathbf{x}_i) \wedge i < k$ **do**
  $i \leftarrow i + 1$;
  $\mathbf{x}_i \leftarrow \mathbf{x}_{\mathcal{M}}(\mathbf{x}_{i-1}, \omega_{i-1})$;
  $\omega_i \leftarrow$ SAMPLE $f_S(\mathbf{x}_i, -\omega_{i-1}, \mathbf{u}_i)$;
return $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_i$;

Algorithm 2: INVERSERANDOMWALK($\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$, adjoint)

$\omega_i \leftarrow \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{||\mathbf{x}_{i+1} - \mathbf{x}_i||}$, $i = 0, \dots, k-1$;
if adjoint **then** $\mathbf{u}_0 \leftarrow$ INVERSESAMPLE $W_e(\mathbf{x}_0, \omega_0)$;
;
else $\mathbf{u}_0 \leftarrow$ INVERSESAMPLE $L_e(\mathbf{x}_0, \omega_0)$;
;
$i \leftarrow 0$;
while $i < k - 1$ **do**
  $i \leftarrow i + 1$;
  $\mathbf{u}_i \leftarrow$ INVERSESAMPLE $f_S(\mathbf{x}_i, -\omega_{i-1}, \omega_i)$;
return $\mathbf{u}_0\mathbf{u}_1 \dots \mathbf{u}_{k-1}$;

gorithm 1 in the same order. Because of the use of local importance sampling, we only have to invert the individual SAMPLE functions and can build the inverse random walk from these building blocks. We give a description of the resulting method in Algorithm 2.

The sample functions appearing in Algorithm 1 are usually constructed from a combination of two sampling primitives. We will review these sample methods in detail along with their inverses in the following sections.

5.1.1 The Inversion Method

The majority of sampling methods encountered in graphics are derived using the *inversion method*, which is a simple recipe for deriving an importance sampling scheme

for analytic functions. Given a 1D importance function f that we wish to sample, the inversion method proceeds as follows:

1. Compute the desired probability density p by normalizing f over the input domain

$$p(x) = \frac{f(x)}{\int f(x')\mathrm{d}x'}$$

2. Compute the CDF P as a definite integral of p

$$P(x) = \int_{-\infty}^{x} p(x')\mathrm{d}x'$$

3. Compute the inverse P^{-1} of the CDF

If P^{-1} exists, then for a random variable ξ uniformly distributed in $[0,1)$, $P^{-1}(\xi)$ is distributed with probability density proportional to f. As long as the antiderivative of f is integrable and invertible, we will always be able to derive an analytic sampling scheme, making the inversion method a powerful tool.

A very similar scheme can be applied to importance sample multidimensional importance functions: First, the importance function is normalized to arrive at a joint PDF. This PDF is then marginalized to produce one-dimensional marginal and one or more conditional densities. The inversion method can then be applied to each of those 1D densities individually to arrive at a joint importance sampling method.

We can see that in order to arrive at the inverse CDF P^{-1} to be used for importance sampling, we first have to compute the CDF P. This is precisely the inverse sampling method that we are looking for! In other words, for many importance sampling schemes used in graphics, the inverse sampling function that we are after exists and has already been derived. If an importance sampling scheme is of the form $x = P^{-1}(\xi)$, then we can easily compute the random number that produced a sample by computing $\xi = P(x)$. Even if $P(x)$ is not given, it can be readily derived by inverting $P^{-1}(u)$, which is an invertible function by construction.

5.1.2 Discrete Sampling

Discrete sampling methods, which convert a continuous random number into a discrete decision, are not unusual in graphics. For example, consider the following func-

tion that importance samples the BSDF of a smooth dielectric surface:

$$\textsc{Sample}(\omega_i, u) = \begin{cases} \text{reflect}(\omega_i), & \text{if } u < F_r(\omega_i) \\ \text{refract}(\omega_i), & \text{otherwise,} \end{cases} \tag{5.1}$$

where F_r is the Fresnel reflection coefficient.

Here, we transformed the input random number into a binary decision on whether to reflect or to refract. Unfortunately, such a function is not bijective – given only the sampled direction, it is impossible to retrieve the exact random number that produced it, since there is an entire interval of numbers mapping to the same decision.

For our purposes of constructing perturbation strategies for MLT, we fortunately do not need the exact inverse (which may not be well-defined), and allow the inverse random walk some freedom in the numbers it returns. Most importantly, we require that the result of an inverse random walk for a path $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$ results in the same path when used in a forward random walk, i.e.

$$\bar{\mathbf{x}} = \textsc{RandomWalk}(\textsc{InverseRandomWalk}(\bar{\mathbf{x}}),\ k)\,. \tag{5.2}$$

Here and in the following, the same discussion applies for both direct and adjoint random walks, and we omit the adjoint parameter for brevity.

Unfortunately, we cannot resolve ambiguities due to non-bijectivity in a deterministic manner. For symmetry reasons, we require that the values returned by the inverse random walk for a path $\bar{\mathbf{x}}$ are distributed with the same density as the numbers with which $\bar{\mathbf{x}}$ was sampled. Let $\mathbf{U}$ be a random variable in $\mathcal{U}$, and assume $\bar{\mathbf{x}} = \mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$ was sampled with $\bar{\mathbf{x}} = \textsc{RandomWalk}(\mathbf{U}, k)$. Then the result of $\textsc{InverseRandomWalk}(\bar{\mathbf{x}})$ should be a random variable distributed with

$$Pr\{\textsc{InverseRandomWalk}(\bar{\mathbf{x}}){=}\mathbf{u}\} = Pr\{\mathbf{U}{=}\mathbf{u} \mid \textsc{RandomWalk}(\mathbf{u}, k){=}\bar{\mathbf{x}}\}\,. \tag{5.3}$$

For our purposes, it is enough to assume that $\mathbf{U}$ is distributed uniformly in $\mathcal{U}$.

It's useful to look at a few examples to illustrate the intuition behind this condition. If there is only one possible state $\mathbf{u}$ that could have produced a path $\bar{\mathbf{x}}$, then $Pr\{\textsc{InverseRandomWalk}(\bar{\mathbf{x}}){=}\mathbf{u}\}$ reduces to a delta function, and $\textsc{InverseRandomWalk}(\bar{\mathbf{x}})$ simply always returns $\mathbf{u}$. If there is a contiguous region $A \subset \mathcal{U}$ of random numbers that could produce the same path, then the return value of $\textsc{InverseRandomWalk}(\bar{\mathbf{x}})$ is uniformly distributed within that region. If there are multiple contiguous regions $A_1, \dots, A_N \subset \mathcal{U}$ that map to the same path, then the return value of $\textsc{InverseRandomWalk}(\bar{\mathbf{x}})$ is uniformly distributed within each region, and $Pr\{\textsc{InverseRandomWalk}(\bar{\mathbf{x}}) \in A_i\}$ is proportional to the size of the region A_i.

This leads to a straightforward inversion scheme for discrete decisions. For the smooth dielectric BSDF from the earlier example, its inverse sample function is simply

$$\textsc{InverseSample}(\omega_i, \omega_o) = \begin{cases} \xi F_r(\omega_i), & \text{if } \omega_o = \text{reflect}(\omega_i) \\ F_r(\omega_i) + \xi(1 - F_r(\omega_i)), & \text{otherwise,} \end{cases} \tag{5.4}$$

where ξ is a random variable uniformly distributed in $[0, 1)$. That is, we simply uniformly distribute the returned result within the interval of random numbers that produce the same decision. More generally, for a discrete decision with N different outcomes, each chosen with weight w_i, a valid sampling scheme is

$$\textsc{SampleDiscrete}(u, w_1, w_2, \ldots, w_N) = \arg\min_i \{uW < w_1 + w_2 + \ldots + w_i\} \ , \tag{5.5}$$

with $W = \sum_{i=1}^{N} w_i$. Given that the sampled decision was i, a valid inversion scheme fulfilling our requirements is

$$\textsc{InvertDiscrete}(i, w_1, w_2, \ldots, w_N) = w_1 + w_2 + \ldots + w_{i-1} + \xi \, w_i \, , \tag{5.6}$$

where ξ is the same as before. Similarly to before, we simply uniformly distribute the returned result within the interval of random numbers that map to the same decision.

Finally, we might run into the case where multiple outcomes $i_1, i_2, \ldots, i_L$ of the discrete decision can lead to the same path (e.g. layered materials). This is precisely the case discussed previously, in which multiple regions A_i map to the same decision. This means we must first select a region with weight proportional to its size, i.e.

$$j = \textsc{SampleDiscrete}(\xi_1, w_{i_1}, w_{i_2}, \ldots, w_{i_L}) \, . \tag{5.7}$$

Then we invert as before

$$\textsc{InvertDiscrete}(i_1, i_2, \ldots, i_L, w_1, w_2, \ldots, w_N) = w_1 + w_2 + \ldots + w_{j-1} + \xi_2 \, w_j \, . \tag{5.8}$$

Here, ξ_1 and ξ_2 are two random variables uniformly distributed in $[0, 1)$.

5.1.3 Discussion

Maybe surprisingly, the combination of discrete mappings and the inversion method makes up nearly all importance sampling methods encountered in computer graphics. For example, importance sampling a position on an emitter involves discretely picking an emitter surface and then uniformly sampling its area; importance sampling a

layered material such as the Phong shading model [Pho75] involves sampling a discrete BSDF index and then sampling the selected BSDF; importance sampling a texture involves discretely selecting a texel and uniformly sampling its area, and so forth. Being able to robustly invert these two sampling methods allows us to invert nearly any sampled path back to random numbers.

However, there are a few exceptions to this rule. For some functions, such as the Normal distribution, specialized sampling schemes (e.g. Box-Muller [BM58]) are commonly employed that have not been derived using the inversion method. If no information is lost during the sampling process, these sampling schemes can usually be inverted, but it depends on the method.

The rejection method is an alternative sampling recipe for continuous functions that does not require the function to have an invertible antiderivative and is sometimes used for difficult sampling distributions, e.g. in the form of Woodcock tracking [WMHTC65] for heterogeneous participating media. Inverting samples produced by the rejection method is a practically impossible task, and path inversions will not be possible in these cases.

It is also important to realize that the space of paths $\bar{\mathbf{x}}$ has more degrees of freedom than the space of random numbers $\mathbf{u}$, and that there are classes of paths that cannot be produced by any random walk. Therefore, it is important to engineer inverse random walks in such a way that they are allowed to fail, and that the algorithms built around such functions can handle a path inversion failure, even if just due to limited numerical precision.

5.1.4 The Inverse Path Sample Function

In this chapter, we are primarily interested in paths generated by bidirectional path tracing, and we now have all the tools available to write down the corresponding path sample function and its inverse. The bidirectional path sample function will be denoted as $S_{k,i}(\mathbf{u})$, where k is the path length and i specifies which sampling technique was used to generate the path. If the path length k is clear from the context, we will drop it as a subscript for brevity and only write down $S_i(\mathbf{u})$.

Let $s = i$ and $t = k + 1 - i$. The path sample function is then

$$S_{k,i}(\mathbf{u}_0\mathbf{u}_1 \dots \mathbf{u}_k) = \mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_{s-1}\mathbf{z}_{t-1} \dots \mathbf{z}_1\mathbf{z}_0\,, \tag{5.9}$$

where

$$\mathbf{y}_0\mathbf{y}_1\dots\mathbf{y}_{s-1} = \textsc{RandomWalk}(\mathbf{u}_0\mathbf{u}_1\dots\mathbf{u}_{s-1}, s-1, \text{false}) \tag{5.10}$$

$$\mathbf{z}_0\mathbf{z}_1\dots\mathbf{z}_{t-1} = \textsc{RandomWalk}(\mathbf{u}_s\mathbf{u}_{s+1}\dots\mathbf{u}_k, t-1, \text{true})\,. \tag{5.11}$$

Its inverse is given by

$$S^{-1}_{k,i}(\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_k) = \mathbf{u}_0\mathbf{u}_1\dots\mathbf{u}_k\,, \tag{5.12}$$

where

$$\mathbf{u}_0\mathbf{u}_1\dots\mathbf{u}_{s-1} = \textsc{InverseRandomWalk}(\mathbf{x}_0\mathbf{x}_1\dots\mathbf{x}_{s-1}) \tag{5.13}$$

$$\mathbf{u}_s\mathbf{u}_{s+1}\dots\mathbf{u}_k = \textsc{InverseRandomWalk}(\mathbf{x}_k\mathbf{x}_{k-1}\dots\mathbf{x}_s)\,. \tag{5.14}$$

5.2 Robust Transitions between Sampling Techniques

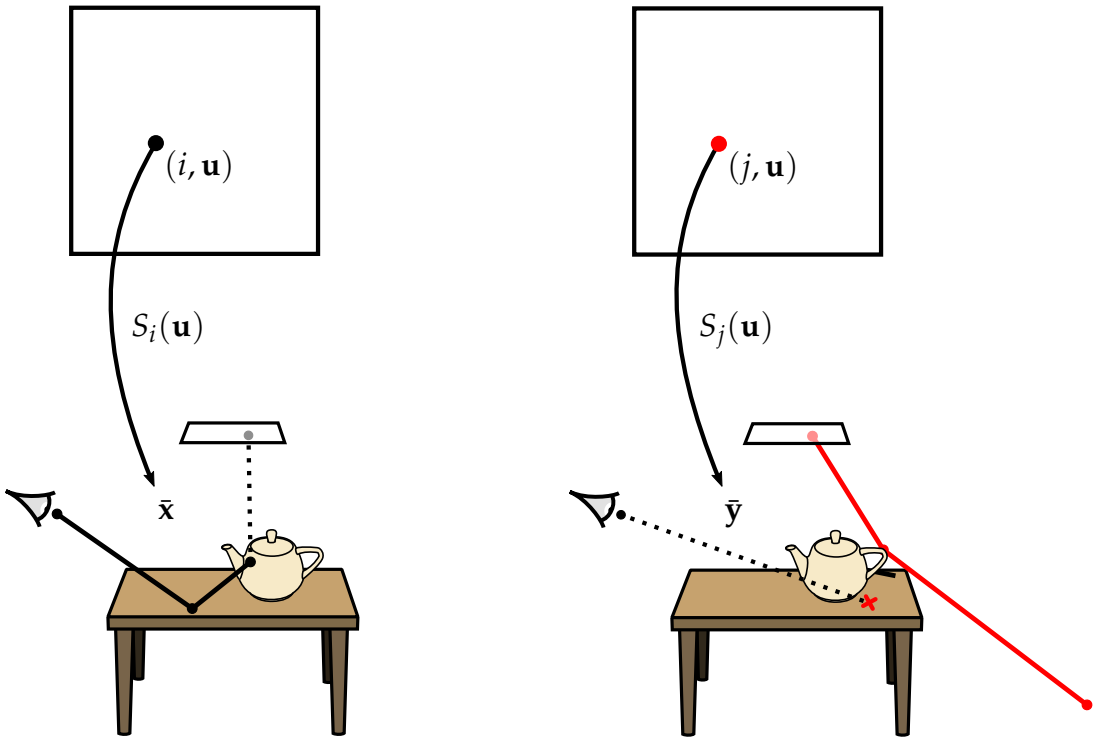
Multiplexed MLT is a light transport algorithm that exposes the use of different sampling techniques to the Markov chain. It does so by running a Markov chain in the extended state space $(t, \mathbf{u})$ which specifies both the position in primary sample space and the mapping that should be used to turn it into a path, and samples the target distribution $\hat{w}_t(\mathbf{u})\hat{C}^*_t(\mathbf{u})$. Because the MIS weight $\hat{w}_t(\mathbf{u})$ is included in the target distribution, inappropriate sampling techniques for a given path receive a lower contribution and the MCMC sampler will automatically try to move away from such states.

However, the use of different mappings from primary sample space to paths can make it difficult for the Markov chain to move from one sampling technique to another. We can illustrate this problem with a hypothetical TechniquePerturbation that keeps the random number vector fixed and only attempts to transition to a different sampling technique. Suppose that the Markov chain for paths of length k is in the state $(i, \mathbf{u})$, and the perturbation proposes a new state $(j, \mathbf{u})$, where j was chosen uniformly random from $0, \dots, k+1$. The corresponding acceptance probability is

$$r((i,\mathbf{u}) \to (j,\mathbf{u})) = \frac{\hat{w}_j(\mathbf{u})\hat{C}^*_j(\mathbf{u})T(j \to i)}{\hat{w}_i(\mathbf{u})\hat{C}^*_i(\mathbf{u})T(i \to j)}\,. \tag{5.15}$$

Because the proposal distributions are symmetric in this case, this simplifies to

$$r((i,\mathbf{u}) \to (j,\mathbf{u})) = \frac{\hat{w}_j(\mathbf{u})\hat{C}^*_j(\mathbf{u})}{\hat{w}_i(\mathbf{u})\hat{C}^*_i(\mathbf{u})}\,. \tag{5.16}$$



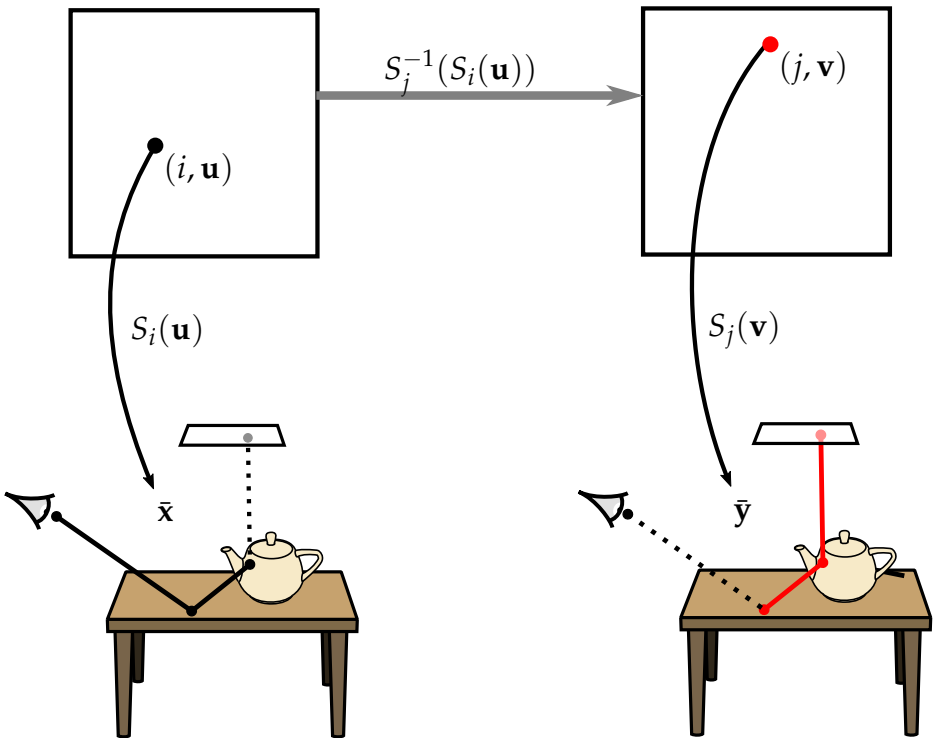
(a) Current State **(b)** Proposed State (MMLT)

Figure 5.1: *Illustration of the current state of a Markov chain (a) and a proposal path (red) generated using* TECHNIQUEPERTURBATION *(b). In MMLT style perturbations, the technique index is changed without much consideration for the position in primary sample space, leading to a completely different path.*

Since we only changed the sampling technique and not $\mathbf{u}$, we would hope the acceptance probability to only depend on the ratio of MIS weights, i.e. on how well the proposed technique samples the current path compared to the current sampling technique. However, this is not what is actually happening. This becomes clearer when we replace the target distribution in primary sample space by its counterpart in path space:

$$r((i, \mathbf{u}) \to (j, \mathbf{u})) = \frac{w_j(S_j(\mathbf{u}))C^*(S_j(\mathbf{u}))}{w_i(S_i(\mathbf{u}))C^*(S_i(\mathbf{u}))}\,. \tag{5.17}$$

Even though the position in primary sample space is not changed, the proposed state uses a *different mapping* than the current state to transform that position into a light path. In general, $S_i(\mathbf{u}) \neq S_j(\mathbf{u})$, and it is likely that the proposed path (and therefore its contribution) is very different from the current path (Figure 5.1). Such large changes are unlikely to be accepted, and the ability of the Markov chain to transition between different sampling techniques is greatly impeded.



(a) Current State **(a)** Proposed State (Ours)

Figure 5.2: *Illustration of the current state of a Markov chain (a) and a proposal path (red) generated using our path-invariant technique perturbation (b). When changing sampling techniques, our proposed perturbation simultaneously jumps to a different point in primary sample space, such that the sampled transport path remains the same.*

5.2.1 A Path-Invariant Technique Perturbation

What we would actually like to do during such a perturbation is to keep the current path fixed while we change the sampling strategy. In other words, we need to find a new point $\mathbf{v}$ in primary sample space such that $S_j(\mathbf{v}) = S_i(\mathbf{u})$. Fortunately, this is easily accomplished if we have access to the inverse sampling technique S_j^{-1}: In that case, we can simply write the new point in primary sample space as $\mathbf{v} = S_j^{-1}(S_i(\mathbf{u}))$.

In Section 5.1.4, we described how such an inverse can be constructed, and having access to S_j^{-1} allows us to build a viable perturbation for changing sampling techniques that resolves some of the shortcomings of MMLT (Figure 5.2). It proceeds as follows:

1. Pick a new sampling strategy j to transition to
2. Compute the new position in primary sample space, $\mathbf{v} = S_j^{-1}(S_i(\mathbf{u}))$

3. Accept the proposal state $(j, \mathbf{v})$ with probability $r((i, \mathbf{u}) \to (j, \mathbf{v}))$

The acceptance ratio is somewhat unusual compared to other perturbations in primary sample space. Because we fix the position of the chain in path space and no new path is sampled, the PDF of the path sampling strategy vanishes in the acceptance ratio. The remaining terms are

$$r((i, \mathbf{u}) \to (j, \mathbf{v})) = \frac{w_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) T(j \to i)}{w_i(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) T(i \to j)} . \tag{5.18}$$

In most cases, the path contribution f in path space will be the same for both sampling techniques and cancels out[1]. The remaining acceptance ratio only depends on the MIS weights and the proposal distribution, which was our original intent.

Another interesting aspect of this perturbation is that the state space that is actually changed is discrete, and the acceptance ratio can only take on one of a finite number of states. What's more is that all of the terms in the acceptance ratio are known *before generating the proposal*. Therefore, we can carefully construct the proposal distribution so that it cancels with the remaining terms in the acceptance ratio. Consider

$$T(i \to j) = w_j(\bar{\mathbf{x}}) . \tag{5.19}$$

Notably, this proposal distribution does not depend on the current state i and simply importance samples the MIS weights. Inserting this into the acceptance ratio yields

$$r((i, \mathbf{u}) \to (j, \mathbf{v})) = \frac{w_j(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) T(j \to i)}{w_i(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) T(i \to j)} = \frac{w_j(\bar{\mathbf{x}}) w_i(\bar{\mathbf{x}})}{w_i(\bar{\mathbf{x}}) w_j(\bar{\mathbf{x}})} = 1 . \tag{5.20}$$

Assembling all of these parts yields our proposed path-invariant technique perturbation, which proceeds as follows:

1. Choose a proposal technique j with probability $w_j(S_i(\mathbf{u}))$
2. Compute the proposal state $\mathbf{y} = (j, S_j^{-1}(S_i(\mathbf{u})))$
3. Accept $\mathbf{y}$

An interesting aspect of this perturbation is that it will never change the current light path; in other words, whenever this perturbation is employed, the Markov Chain stays at the same pixel. This is unfortunate, since this forces us to balance two conflicting goals: On the one hand, we want the Markov Chain to switch sampling techniques as often as possible in order to explore the state space, and we should use the technique

[1] This is not always the case due to non-symmetric scattering. We refer the reader to Veach [Vea96] for details.

perturbation frequently; on the other hand, we do not want the Markov Chain to get stuck on the same light path for many states, so we should not use this perturbation very often.

To obtain a workable algorithm, we therefore combine the technique perturbation with small steps. Whenever MMLT would do a small step, we first employ a technique perturbation as described above (which is always accepted), and then perturb the newly obtained random number vector to generate a proposal state (which is accepted with the standard MMLT acceptance probability for small steps). This ensures that the Markov Chain perturbs the sampling technique as often as possible while making sure the light path is also perturbed often.

Note that the technique perturbation has non-zero probability of choosing the current sampling technique again. Because it simply importance samples the MIS weights, we are in fact very likely to retain the current sampling technique if it is among the best techniques of the current path. This means that there is no harm in performing technique perturbations often, since they don't necessarily force us to move to a different sampling technique if the current technique is already performing well.

We evaluate the performance of this perturbation in Chapter 6 in more detail.

5.2.2 Implementation Details

Efficient Computation of all MIS Weights In order to importance sample the proposal technique j, we are required to compute the MIS weights of *all* techniques for the current path, compared to the single MIS weight required for computing the path contribution. Following Veach [Vea98], we use the following transform to compute the MIS weight of a single technique:

$$w_i(\bar{\mathbf{x}}) = \frac{p_i(\bar{\mathbf{x}})}{\Sigma_j p_j(\bar{\mathbf{x}})} = \frac{1}{\Sigma_j (p_j(\bar{\mathbf{x}})/p_i(\bar{\mathbf{x}}))}\,. \tag{5.21}$$

The ratio of probabilities p_j/p_i can be computed incrementally and is much more numerically well-behaved than the individual probabilities p_j, and this is a commonly used technique in implementations of bidirectional path tracing. It so happens that these ratios are exactly the weights we need in order to sample j. Consider the unnormalized discrete distribution

$$\hat{b}_s = p_s(\bar{\mathbf{x}})/p_i(\bar{\mathbf{x}})\,. \tag{5.22}$$

The corresponding probability mass function (PMF) is

$$b_s = \frac{p_s(\bar{\mathbf{x}}) / p_i(\bar{\mathbf{x}}))}{\Sigma_j (p_j(\bar{\mathbf{x}}) / p_i(\bar{\mathbf{x}})} \tag{5.23}$$

$$= \frac{p_s(\bar{\mathbf{x}})}{\Sigma_j p_j(\bar{\mathbf{x}})} \tag{5.24}$$

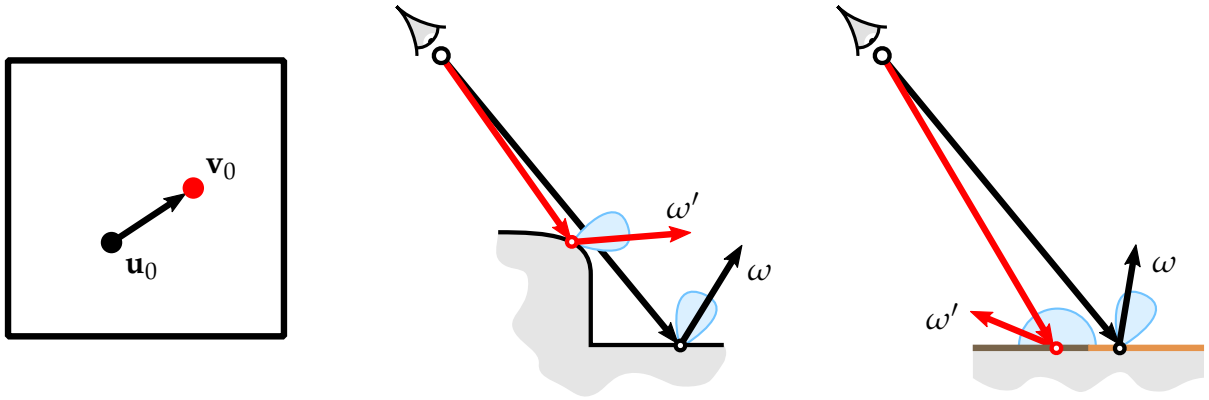
$$= w_s(\bar{\mathbf{x}}) \,. \tag{5.25}$$

In other words, we can avoid an expensive computation of all MIS weights and simply sample j with weight $p_j(\bar{\mathbf{x}}) / p_i(\bar{\mathbf{x}})$, which is equivalent to sampling j with probability $w_j(\bar{\mathbf{x}})$ regardless of i. These weights are already being computed by most implementations of bidirectional path tracing and make sampling j no more expensive than a single MIS weight computation (as opposed to $k + 2$ MIS weight computations).

Optimized Path Inversion At the core of the technique perturbation is the inverse bidirectional sample function, $S_j^{-1}(\bar{\mathbf{x}})$, which is written in terms of two invocations of an inverse random walk. In this form, a single technique change requires inverting the entirety of the current path back to primary sample space, which is an expensive operation for long paths. We can optimize this operation by reusing some of the current state, $\mathbf{u}$. We observe that during a technique change, one of the subpaths of the proposal state is truncated compared to the current state, while the other subpath is extended. During this process, $|j - i|$ vertices are removed from one subpath and added to the other, which on average is much smaller than $k + 1$, the total number of vertices on the path.

Since we know which elements of the current state were used to sample which subpath, we can simply copy the elements of $\mathbf{u}$ corresponding to the vertices of the truncated subpath, since these are left unchanged. This saves one invocation of INVERSERANDOMWALK.

Usually, we are also able to implement inverse random walks in a way that allows us to invert only part of a path. In that case, we can also reuse all of the elements of $\mathbf{u}$ that correspond to the extended subpath, and merely have to append the random numbers corresponding to the vertices that were added to the extended subpath. During a technique change, this allows us to perform a full path inversion by processing only $|j - i|$ vertices rather than $k + 1$.



(a) Geometric discontinuities **(b)** Material changes

Figure 5.3: *Small steps in primary sample space (left) can lead to large changes to the path due to geometric discontinuities (a) or material changes (b)*

5.3 Controlled Small Steps

The primary tool for path exploration in PSSMLT and MMLT is the small step perturbation, which performs a small change of the current position in primary sample space to obtain a proposal state. The intuition behind this strategy is that light transport is spatially coherent, and that nearby paths tend to have similar contribution to the image. In the primary sample space view, the Markov chain does not have direct access to the light path; instead, it is assumed that spatial coherence translates to primary sample space as well, and that similar points in primary sample space have similar contributions to the image.

We've already seen two situations where this was not the case: Section 4.2 explored the problem of discrete choices, where small changes in primary sample space can lead to a large change in the light path, and Section 5.2 showed that similar caveats apply when changing sampling strategies in MMLT. In this section, we show how geometric discontinuities and material changes can translate small steps in primary sample space to large steps in path space and propose a new perturbation based on inverse path sampling that can remedy this problem.

5.3.1 Discontinuities in Primary Sample Space

To illustrate the problems caused by discontinuities, we consider a hypothetical proposal generator, RootPerturbation. This method only perturbs the random num-

bers associated with the *root vertex*, i.e. the first vertex on a subpath. We expect such a small change in primary sample space to translate to a very small change to the light path – the proposed light path may be slightly offset compared to the current light path, but overall we expect its structure to be largely the same.

Geometric Discontinuities Figure 5.3 (a) illustrates that this is not the case when the proposed light path steps over a geometric edge, causing a vertex to land on a surface with completely different surface normal. Even though the random numbers associated with this vertex were not changed, they are now *interpreted differently*. The BSDF associated with the surface will sample an outgoing direction in a local coordinate frame aligned with the surface normal, and a large change in the surface normal leads to a large change in the sampled outgoing direction even when the random numbers are left unchanged.

Material Changes Figure 5.3 (b) illustrates how similar effects can occur if a vertex on the proposed path lands on a surface with a different material. Even though the random numbers associated with this vertex were not changed, the change in material causes them to be interpreted differently, leading to a large change in the sampled outgoing direction. This is because different BSDFs generally use different importance sampling strategies, i.e. different mappings from primary sample space to directions.

Small step perturbations have trouble stepping over these discontinuities and get "stuck" in a local region of primary sample space. PSSMLT and MMLT rely on large step mutations to jump out of these regions, but these have a much lower average acceptance ratio than small steps, causing the chain to spend more time than necessary in a small part of primary sample space.

5.3.2 Crossing Discontinuities with Path Inversions

Perturbing only the random numbers associated with the root vertex is evidently not sufficient to avoid large changes to the path. In the ideal case, the edges on the proposal path $\bar{\mathbf{y}}$ proposed by RootPerturbation are parallel to the current path $\bar{\mathbf{x}}$; that is

$$\frac{\mathbf{y}_{i+1} - \mathbf{y}_i}{||\mathbf{y}_{i+1} - \mathbf{y}_i||} = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{||\mathbf{x}_{i+1} - \mathbf{x}_i||}, j = 1, \ldots, k-1\,. \tag{5.26}$$

Maybe surprisingly, a proposal path defined this way is fully determined by the current path and the location of the proposed root vertex – no sampling of directions other

Algorithm 3: RootPerturbation($\mathbf{u}$, $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$, adjoint)

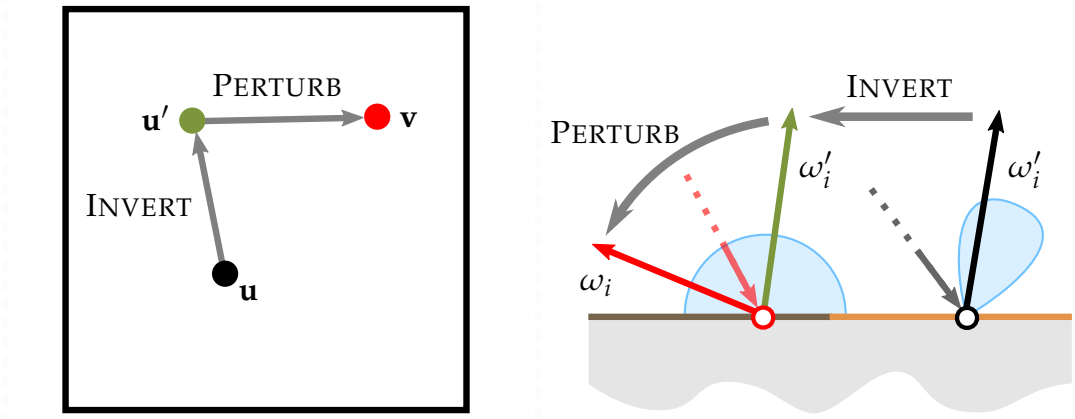
$\mathbf{v}_0 \leftarrow \text{Perturb}(\mathbf{u}_0)$;
if adjoint **then** $\mathbf{y}_0, \omega_0 \leftarrow \text{Sample}\, W_e(\mathbf{v}_0)$;
;
else $\mathbf{y}_0, \omega_0 \leftarrow \text{Sample}\, L_e(\mathbf{v}_0)$;
;
$i \leftarrow 0$;
while IsFinite($\mathbf{y}_i$) $\wedge$ $i < k$ **do**
  $i \leftarrow i + 1$;
  $\mathbf{y}_i \leftarrow \mathbf{x}_{\mathcal{M}}(\mathbf{y}_{i-1}, \omega_{i-1})$;
  $\omega_i \leftarrow \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{||\mathbf{x}_{i+1} - \mathbf{x}_i||}$;
return InverseRandomWalk($\mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_i$, adjoint);

than at the root vertex is required. If the Markov chain operated purely in path space, then the implementation of such a perturbation is straightforward.

Unfortunately, in the primary sample space perspective, such direct operations on paths are normally not available to us. The state of the Markov chain resides in a different space, and any change to the path would have to be transformed into an equivalent change to the random number vector.

However, the inverse path sample function promises to do just that! Since it is able to transform light paths into points in primary sample space, any change we make to the path can be turned into an equivalent change to the random number vector. This allows us to create an implementation of RootPerturbation, given in Algorithm 3. It takes the current state $\mathbf{u}$ as well as the corresponding light path $\bar{\mathbf{x}}$ and returns a proposal state in primary sample space. It does so by first perturbing the random numbers associated with the first vertex (e.g. using a PSSMLT small step) and retrieves the position of the perturbed root vertex. After that, it traces a proposal path $\bar{\mathbf{y}}$, copying the outgoing directions from $\bar{\mathbf{x}}$ at each step. Finally, the proposal state is computed from the proposal path using an inverse random walk.

Such a perturbation allows us to trace an "offset path" in a way that is robust against discontinuities caused by material changes or geometric edges. In general, we do not expect RootPerturbation to be a very useful strategy when employed in a PSSMLT context, since the changes it proposes are much too small to properly explore the state space. However, we can reuse some of its ideas to construct an alternative small step perturbation that is more robust to discontinuities.



(a) Primary Sample Space **(b)** Path Space

Figure 5.4: *An illustration of our proposed small steps in primary sample space (a) and path space (b). The proposal path (dotted red) lands on a surface with a different material than the current path (dotted black). We first jump to an intermediate point* $\mathbf{u}'$ *(green) in primary sample space that reproduces the same outgoing direction at the vertex on the proposed path (green arrow) as on the current path (black arrow). We then perturb this point to obtain the proposal state* $\mathbf{v}$ *(red). For illustration purposes we use a relatively large perturbation here; in practice, the perturbation would be much smaller.*

5.3.3 An Alternative Small Step Perturbation

When we perform a small step in PSSMLT, some of the change in the outgoing direction at each proposal vertex will be due to the random numbers being slightly perturbed, and some of the change will be due to the vertices landing on surfaces with different orientation or material. We wish to remove the change introduced by landing on a different surface, since the magnitude of this change is largely uncontrollable and might make the mutation strategy less robust to complex scenes.

Given a vertex on the proposal path $\mathbf{y}_i$, we know what the outgoing direction at this vertex should be if its random numbers were not changed: It is $(\mathbf{x}_{i+1} - \mathbf{x}_i)/||\mathbf{x}_{i+1} - \mathbf{x}_i||$, which is what we used in ROOTPERTURBATION. However, during a small step, we also want to perturb this direction slightly, using the same tools as PSSMLT. In order to do this, we first need to compute what the position in primary sample space would need

to be to produce the unperturbed direction:

$$\omega_i' = \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{||\mathbf{x}_{i+1} - \mathbf{x}_i||} \tag{5.27}$$

$$\mathbf{u}_i' = \textsc{InverseSample}\, f_S(\mathbf{y}_i,\, -\omega_{i-1},\, \omega_i')\,. \tag{5.28}$$

Computing $\mathbf{u}_i'$ allows us to reassume the primary sample space perspective and perform the actual perturbation of the direction using the random numbers only:

$$\mathbf{v}_i = \textsc{Perturb}(\mathbf{u}_i') \tag{5.29}$$

$$\omega_i = \textsc{Sample}\, f_S(\mathbf{y}_i, -\omega_{i-1}, \mathbf{v}_i)\,. \tag{5.30}$$

Repeating this invert-perturb-sample pattern along the entire path allows us to implement an improved version of RootPerturbation that can perturb all vertices while retaining the robustness of RootPerturbation. We illustrate this process in Figure 5.4.

Computing the correct acceptance probability for such a perturbation is slightly more difficult than for other proposals in PSSMLT, since the proposal distributions are no longer symmetric. Fortunately, the actual perturbation is still performed in primary sample space, and inherits much of the simplicity of PSSMLT style perturbations.

The proposal distribution of such a perturbation can be written as the product of proposal distributions at each vertex:

$$T(\mathbf{u} \to \mathbf{v}) = \prod_i T_i(\mathbf{u}_i \to \mathbf{v}_i)\,. \tag{5.31}$$

We denote the PSSMLT small step proposal distribution as T_{PSSMLT}. This directly allows us to write down the proposal distribution of the first vertex

$$T_0(\mathbf{u}_0 \to \mathbf{v}_0) = T_{\text{PSSMLT}}(\mathbf{u}_0 \to \mathbf{v}_0)\,. \tag{5.32}$$

At the remaining vertices, we first perform a jump from $\mathbf{u}_i$ to $\mathbf{u}_i'$ via a call to the inverse sample function. We then perturb $\mathbf{u}_i'$ to obtain the final proposal state, $\mathbf{v}_i$. The jump to $\mathbf{u}_i'$ is fully deterministic, and the proposal distribution is simply

$$T_i(\mathbf{u}_i \to \mathbf{v}_i) = T_{\text{PSSMLT}}(\mathbf{u}_i' \to \mathbf{v}_i) \tag{5.33}$$

$$= T_{\text{PSSMLT}}(S_{\mathbf{y}_i}^{-1}(S_{\mathbf{x}_i}(\mathbf{u}_i)) \to \mathbf{v}_i)\,. \tag{5.34}$$

Here, $S_{\mathbf{x}_i}$ and $S_{\mathbf{y}_i}^{-1}$ are shorthand for forward and inverse BSDF sampling at the vertices $\mathbf{x}_i$ and $\mathbf{y}_i$, respectively.

Inserting this proposal density into the acceptance probability yields

$$r(\mathbf{u} \to \mathbf{v}) = \frac{\hat{f}^*(\mathbf{v}) T(\mathbf{v} \to \mathbf{u})}{\hat{f}^*(\mathbf{u}) T(\mathbf{u} \to \mathbf{v})} \tag{5.35}$$

$$= \frac{\hat{f}^*(\mathbf{v}) T_{\text{PSSMLT}}(\mathbf{v}_0 \to \mathbf{u}_0) \prod_{i=1} T_{\text{PSSMLT}}(\mathbf{v}'_i \to \mathbf{u}_i)}{\hat{f}^*(\mathbf{u}) T_{\text{PSSMLT}}(\mathbf{u}_0 \to \mathbf{v}_0) \prod_{i=1} T_{\text{PSSMLT}}(\mathbf{u}'_i \to \mathbf{v}_i)} \tag{5.36}$$

$$= \frac{\hat{f}^*(\mathbf{v}) T_{\text{PSSMLT}}(\mathbf{v}_0 \to \mathbf{u}_0) \prod_{i=1} T_{\text{PSSMLT}}(S^{-1}_{\mathbf{x}_i}(S_{\mathbf{y}_i}(\mathbf{v}_i)) \to \mathbf{u}_i)}{\hat{f}^*(\mathbf{u}) T_{\text{PSSMLT}}(\mathbf{u}_0 \to \mathbf{v}_0) \prod_{i=1} T_{\text{PSSMLT}}(S^{-1}_{\mathbf{y}_i}(S_{\mathbf{x}_i}(\mathbf{u}_i)) \to \mathbf{v}_i)}\,. \tag{5.37}$$

Note that even if the PSSMLT proposal distribution T_{PSSMLT} is symmetric, the distribution of the proposed perturbation is not. This is because the perturbation does not transition from $\mathbf{u}_i$ to $\mathbf{v}_i$ in primary sample space like small steps do, but from an intermediate point $\mathbf{u}'_i$ to $\mathbf{v}_i$. Generally, $\mathbf{v}_i - \mathbf{u}'_i$ is not equal to $\mathbf{v}'_i - \mathbf{u}_i$, so even a radially symmetric proposal distribution such as a Gaussian would not cancel when inserted into the acceptance probability above.

In order to compute the correct acceptance probability, we need to keep track of four points in primary sample space: $\mathbf{u}_i$, the current state; $\mathbf{u}'_i$, the state corresponding to the current direction at the proposed vertex; $\mathbf{v}_i$, the state corresponding to the proposed direction at the proposed vertex; and $\mathbf{v}'_i$, the state corresponding to the proposed direction at the current vertex. All of these points can be computed as the path is sampled.

Algorithm 4 gives a pseudo-code implementation of the proposed perturbation. It takes the current state, i.e. the position in primary sample space and the resulting path, and returns the proposed position in primary sample space and the evaluated proposal distributions required to compute the acceptance probability.

The algorithm first computes a perturbed root vertex, sampled from either camera or emitter, depending on the transport direction. It then proceeds to apply the invert-perturb-sample pattern along the entire path, tracking the four points in primary sample space required to compute the acceptance probability along the way. Finally, the proposal distributions are evaluated and the proposed primary sample space position is returned.

Note that Algorithm 4 can only handle unidirectionally sampled paths. For bidirectionally sampled paths, we simply need to run SAMPLEPERTURBEDPATH twice, once for each subpath. The evaluated proposal distributions for both subpaths are then simply multiplied to arrive at the factors required for computing the acceptance probability.

Algorithm 4: SAMPLEPERTURBEDPATH(adjoint, $\mathbf{u}_0\mathbf{u}_1 \dots \mathbf{u}_k$, $\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_k$)

$\mathbf{v}_0 \leftarrow \text{PERTURB}(\mathbf{u}_0)$;
if adjoint **then** $\mathbf{y}_0, \omega_0 \leftarrow \text{SAMPLE}\, W_e(\mathbf{v}_0)$;
;
else $\mathbf{y}_0, \omega_0 \leftarrow \text{SAMPLE}\, L_e(\mathbf{v}_0)$;
;
$i \leftarrow 0$;
while $\text{ISFINITE}(\mathbf{y}_i) \wedge i < k$ **do**
  $i \leftarrow i + 1$;
  $\mathbf{y}_i \leftarrow \mathbf{x}_{\mathcal{M}}(\mathbf{y}_{i-1}, \omega_{i-1})$;
  $\omega_i' \leftarrow \frac{\mathbf{x}_{i+1} - \mathbf{x}_i}{||\mathbf{x}_{i+1} - \mathbf{x}_i||}$;
  $\mathbf{u}_i' \leftarrow \text{INVERSESAMPLE}\, f_S(\mathbf{y}_i, -\omega_{i-1}, \omega_i')$;
  $\mathbf{v}_i \leftarrow \text{PERTURB}(\mathbf{u}_i')$;
  $\omega_i \leftarrow \text{SAMPLE}\, f_S(\mathbf{y}_i, -\omega_i, \mathbf{v}_i)$;
  $\mathbf{v}_i' \leftarrow \text{INVERSESAMPLE}\, f_S(\mathbf{x}_i, -\omega_{i-1}', \omega_i)$;
$T(\mathbf{u} \to \mathbf{v}) \leftarrow T_{\text{PSSMLT}}(\mathbf{u}_0 \to \mathbf{v}_0) \prod_{l=1} T_{\text{PSSMLT}}(\mathbf{u}_l' \to \mathbf{v}_l)$;
$T(\mathbf{v} \to \mathbf{u}) \leftarrow T_{\text{PSSMLT}}(\mathbf{v}_0 \to \mathbf{u}_0) \prod_{l=1} T_{\text{PSSMLT}}(\mathbf{v}_l' \to \mathbf{u}_l)$;
return $\{\mathbf{v}_0\mathbf{v}_1 \dots, \mathbf{v}_i,\ T(\mathbf{u} \to \mathbf{v}),\ T(\mathbf{v} \to \mathbf{u})\}$;

Analysis

In order to evaluate whether such a perturbation is beneficial, we introduce the notion of the average angle change between two transport paths. Consider a subpath $\bar{\mathbf{y}} = \mathbf{y}_0\mathbf{y}_1 \dots \mathbf{y}_{s-1}$ with s vertices, and a perturbed version of that subpath $\bar{\mathbf{y}}' = \mathbf{y}_0'\mathbf{y}_1' \dots \mathbf{y}_{s-1}'$. We measure the angle change between these subpaths as

$$\Delta\alpha(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = \sum_{i=0}^{s-2} \cos^{-1}\left(\frac{\mathbf{y}_{i+1} - \mathbf{y}_i}{||\mathbf{y}_{i+1} - \mathbf{y}_i||} \bullet \frac{\mathbf{y}_{i+1}' - \mathbf{y}_i'}{||\mathbf{y}_{i+1}' - \mathbf{y}_i'||}\right). \tag{5.38}$$

This is simply the sum of the angle differences between corresponding edges on the two subpaths. If $s < 2$, we set $\Delta\alpha(\bar{\mathbf{y}}, \bar{\mathbf{y}}') = 0$.

We wish to compute a similar metric for full transport paths $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ of equal length. If both paths were generated by connecting two subpaths $\bar{\mathbf{y}}$, $\bar{\mathbf{z}}$ and $\bar{\mathbf{y}}'$, $\bar{\mathbf{z}}'$ respectively, then we measure the average angle change between $\bar{\mathbf{x}}$ and $\bar{\mathbf{x}}'$ as

$$\Delta\psi(\bar{\mathbf{x}}, \bar{\mathbf{x}}') = \frac{\Delta\alpha(\bar{\mathbf{y}}, \bar{\mathbf{y}}') + \Delta\alpha(\bar{\mathbf{z}}, \bar{\mathbf{z}}')}{\max(0, s-2) + \max(0, t-2)}. \tag{5.39}$$

This function is simply the sum of angle changes on camera- and emitter subpath, divided by the total number of edges considered. Notably, the connecting edge is not

(a) MMLT small steps **(b)** Our small steps

Figure 5.5: *These images show the average angle change between the current path and the proposal path, generated either by MMLT small steps (a) or our alternative small steps (b) in a jewelry scene with glossy surfaces and geometric discontinuities. Note how MMLT small steps introduce large changes to the path near geometric discontinuities, such as the edges of the jewelry or between faces of the curved backdrop in the top right. This manifests as white halos around edges. Our alternative small steps are much less sensitive to such discontinuities and the image recording the average angle change is considerably more uniform.*

taken into account, since it is not importance sampled by either subpath. It should be mentioned that this function may be ill-defined for paths of length 1, and we disregard such paths for our analysis.

We now utilize this function in order to show how MMLT small steps and our alternative small steps differ in the proposals they generate. Whenever a small step is performed, we compute the average angle change between the current and the proposal path, and splat the computed value to the image plane. If we run the Markov chain for a long time, this will converge to an image where each pixel shows the average angle change introduced by proposals emanating from that pixel.

We show the resulting images for a jewelry scene with glossy surfaces and moderately complex geometry in Figure 5.5, rendered with both MMLT small steps and our alternative small steps. Both images are at equal exposure, where a black pixel corresponds to no change and a completely white pixel corresponds to an average angle change of half a radian or more.

As expected, MMLT small steps introduce larger changes to the path near geometric discontinuities, which manifests as bright halos around the edges of the jewelry. This effect is also visible in the upper right corner, where it is caused by changes in the surface normal between faces of the curved backdrop. On the other hand, our alternative

(a) MMLT small steps **(b)** Our small steps

Figure 5.6: *The same jewelry scene as depicted in Figure 5.5, this time showing the rendered images produced by MMLT small steps (a) and our proposed small steps (b). The noise distribution is worse and less uniform when our small steps are used, visible e.g. on the top of the left ring or in front of the gold ring in the back.*

small steps are relatively robust against these discontinuities, and there are no bright halos around the edges of the jewelry or between faces of the backdrop.

Unfortunately, the actual rendered images tell a different story. To demonstrate the noise behaviour of the two different small steps, we lowered the probability of using a large step to 1%, which forces the Markov Chain to explore path space using small steps most of the time. Figure 5.6 shows the resulting images, rendered with MMLT small steps and our small steps. It appears that by enforcing more controlled changes to the path, our alternative small steps also change the way the Markov Chain explores path space. For example, the noise on top of the left ring in Figure 5.6 is distributed in streaks when our method is used, as the Markov Chain prefers to slide along the curve of the ring and penalizes orthogonal movement. Similarly, glossy caustics, such as on the inside of the left ring or in front of the gold ring in the back, are explored much less uniformly when our small steps are used, manifesting as splotches of noise. These artifacts disappear over time, but much more slowly than comparable artifacts of MMLT small steps.

Therefore, we do not expect our alternative small steps to be useful as a replacement for MMLT small steps. However, they serve as an example of how perturbations operating directly on paths can be reconciled with the primary sample space perspective, and there may be other such perturbations that could benefit primary sample space methods.

CHAPTER 6

Results

In this section, we will evaluate our proposed Path-Invariant Technique Perturbation (PITP), introduced in Section 5.2, in more detail. We implemented our proposed technique and all relevant prior work as additional integrators in the Tungsten renderer [Bit14]. In the following, we compare Multiplexed Metropolis Light Transport augmented with PITP to standard Multiplexed Metropolis Light Transport (MMLT) and Primary Sample Space Metropolis Light Transport (PSSMLT), employed on top of a bidirectional path tracer using the balance heuristic.

We evaluate the benefits of our approach compared to the baseline methods in three interior scenes with difficult lighting: LIVINGROOM, a living room lit by a high-frequency environment map visible through a window outside the camera viewport; STAIRCASE, a staircase lit through a skylight by an environment map; and AJAR, a room lit by a light source in an adjacent room, reachable only through the gap of slightly ajar door. All scenes feature glossy surfaces and complex visibility.

Both LIVINGROOM and STAIRCASE were generously made available by Wayne (Wig42); they were originally titled *The Modern Living Room* and *The Wooden Staircase* and are both available on `http://blendswap.com`. AJAR is modeled after a scene by Eric Veach and Toshiya Hachisuka.

We show comparisons of the resulting images in Figure 6.2, 6.3 and 6.4. We show a reference image, obtained with bidirectional path tracing, and two insets taken from images rendered with PSSMLT, MMLT and our proposed method at equal render time. We used one million samples, generated with a bidirectional path tracer, to estimate the normalization factor.

To allow for an easier comparison, resolution-aware proposals, multiple correlated-try large steps and constrained discrete choices were disabled for our method. For all methods, we set the probability of using a large step to $p_{\text{large}} = 0.1$; a small step is

performed otherwise. When PITP is used, the algorithm will additionally select a new sampling technique before performing a small step.

All of the results were rendered in the cloud using four Google Compute Engine `n1-highcpu-16` instances, which at that time utilized Intel Xeon E5 processors and offered a total of 16 virtual cores each. The reference images were rendered with bidirectional path tracing at 40'000 paths per pixel. In order to exploit parallelism, our implementation launches 16 threads, each running a separate Markov chain. All Markov chains contribute to the same image using lock-free splatting operations. Since MCMC methods perform better on indirect lighting, we exclude direct lighting from the images to allow for a better comparison.

For completeness, we also show full-resolution images of all three scenes rendered with each method in Figure 6.5, 6.6 and 6.7.

In all three scenes, PSSMLT performs the worst out of the methods shown. This is because it considers all possible connections between emitter- and camera subpath, which is more computationally expensive than the single connection considered by MMLT. At equal render time, PSSMLT therefore produces fewer samples than MMLT, leading to a noisier result. MMLT performs much better than PSSMLT, but it has trouble transitioning between different sampling techniques, causing it to become stuck in local regions of path space. This manifests as "streaky" or "splotchy" artifacts, such as on the couch in Figure 6.2 (red inset), the floor in Figure 6.3 (red inset), or the golden teapot in Figure 6.4 (yellow inset). In contrast, PITP can rapidly switch between sampling techniques without suffering high rejection rates, and explores path space more uniformly than MMLT, resulting in more uniform noise.

To further illustrate the advantage of PITP, we record the average acceptance probability of small steps in MMLT and PITP for the AJAR scene and plot it in Figure 6.1. We separately record the acceptance probabilities of small steps that change the sampling technique used to generate the path, and those that leave it unchanged. As expected, small steps in MMLT that change the sampling technique generally cause large changes to the path, and the resulting proposals are very likely to be rejected. In contrast, PITP can change sampling techniques without such a penalty, and the acceptance probabilities in PITP are largely unaffected by technique changes.

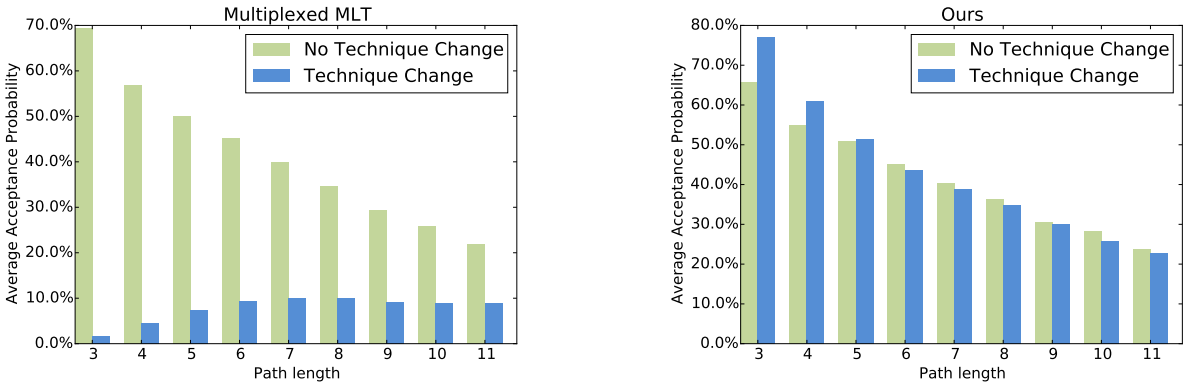


Figure 6.1: *Average acceptance probability of small steps for MMLT (left) and PITP (right) in the* AJAR *scene, plotted over different path lengths. We differentiate between small steps that change the sampling technique and those that leave it unchanged. Note how technique changes in MMLT are very unlikely to be accepted, whereas technique changes in PITP have little effect on the acceptance probability.*

Figure 6.2: *Equal-time renders of a living room scene lit by an environment map, visible through a small window. The combination of dielectric and layered materials and complex visibility make this scene challenging to render.*

Figure 6.3: *Equal-time renders of a staircase, lit by a skylight above the chandelier. Much of the lighting in this scene is indirect and reflects off of glossy surfaces such as the wooden floor and staircase.*

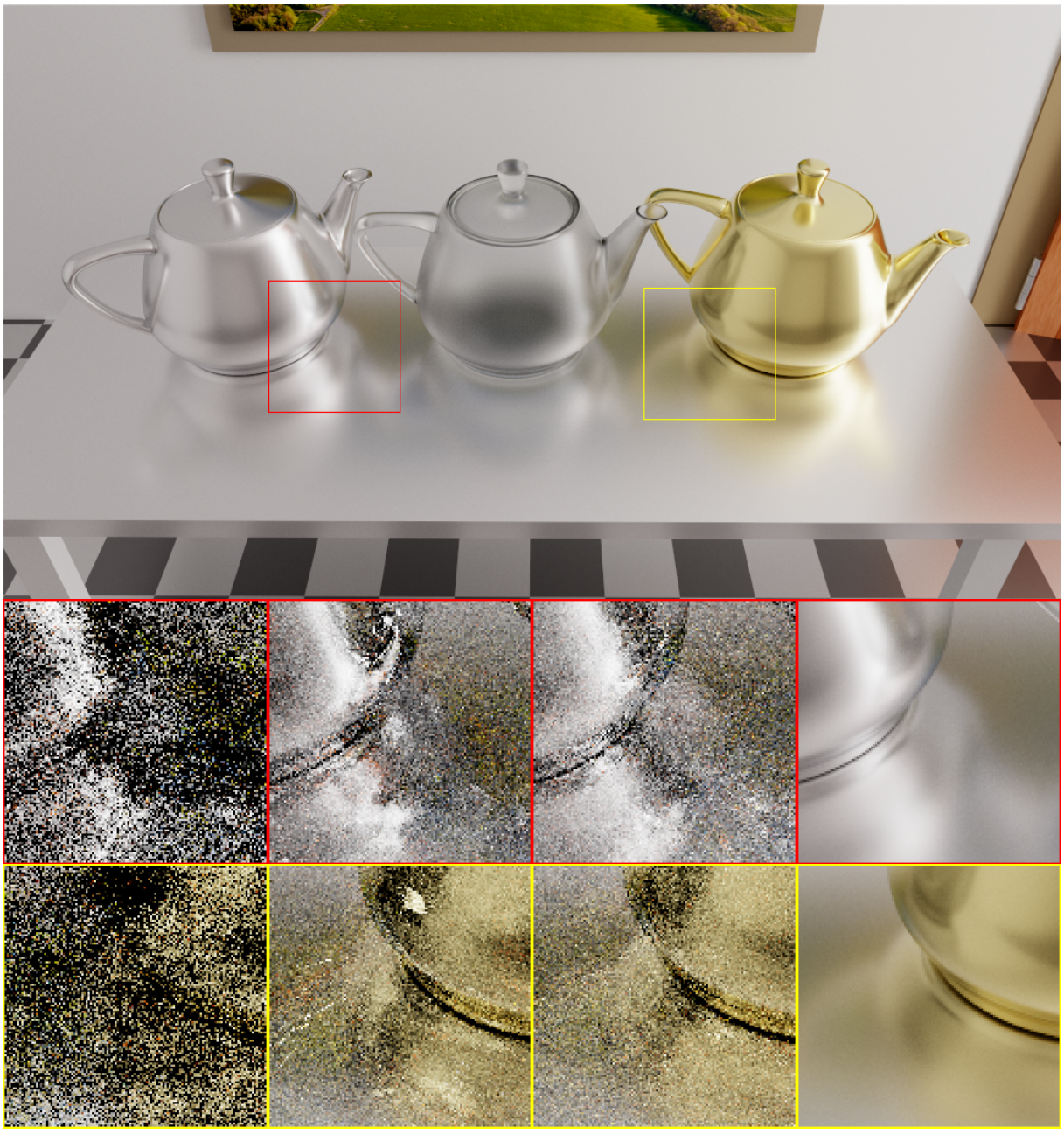


Figure 6.4: *Equal-time renders of an interior room, light by a light source in an adjacent room. The light can only enter through a slightly ajar door, and light-carrying paths are difficult to find. Much of the lighting comes from glossy interreflections. Modeled after a scene by Eric Veach and Toshiya Hachisuka.*

Figure 6.5: *Full resolution images of the insets from Figure 6.2*

Figure 6.6: *Full resolution images of the insets from Figure 6.3*

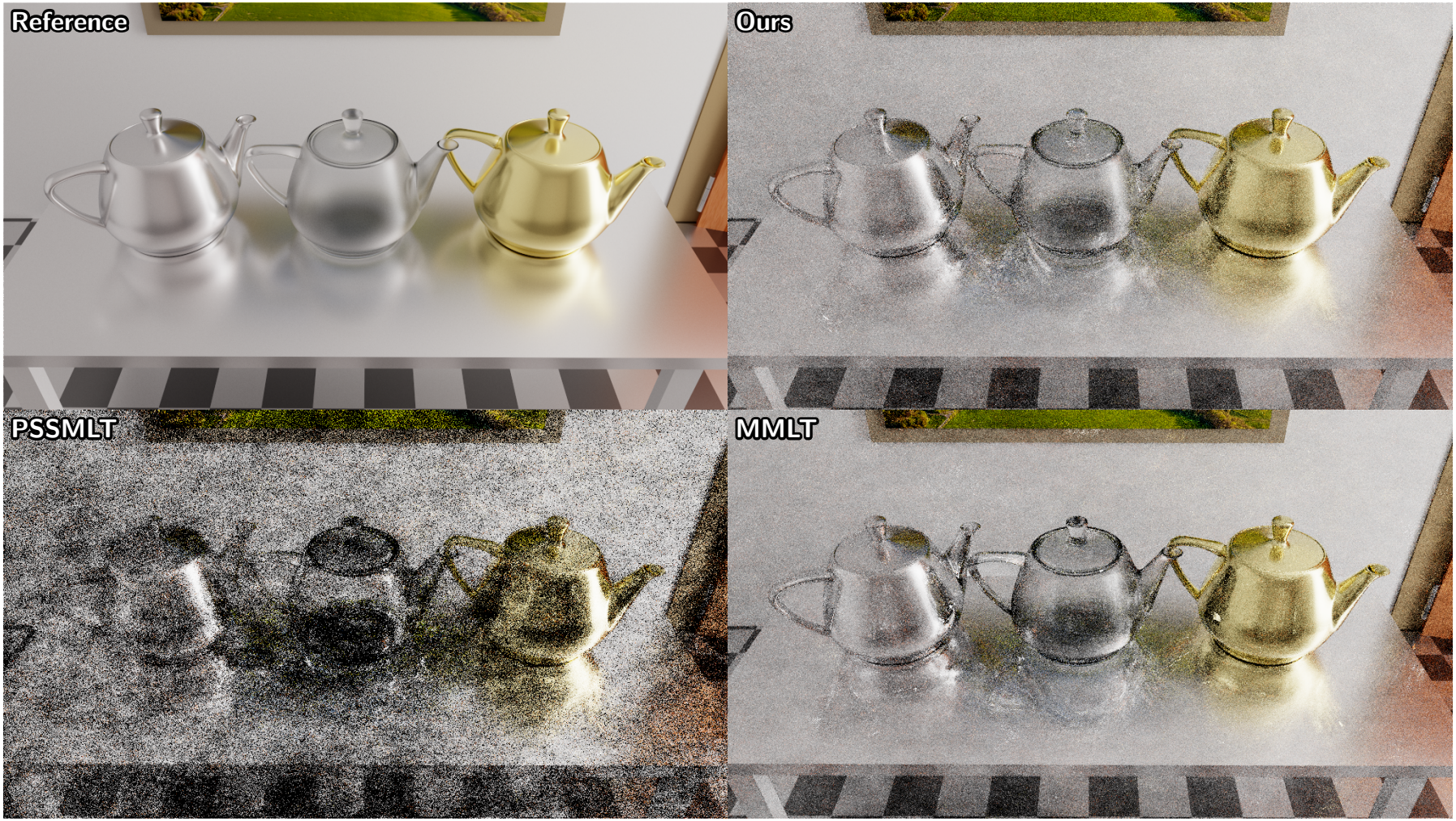


Figure 6.7: *Full resolution images of the insets from Figure 6.4*

CHAPTER 7

Conclusion

In this thesis, we took a closer look at mappings from the space of random numbers to path space and how they relate to MCMC methods operating in primary sample space.

In the first half of the thesis, we began by investigating how the image resolution and discrete choices on random walks interact suboptimally with the sampling process of MCMC methods in primary sample space. We proposed simple remedies to these shortcomings that can be easily integrated with previous work.

We also showed how large steps in Multiplexed Metropolis Light Transport can perform suboptimally and proposed an alternative large step mutation that does not suffer from these issues. Our proposed large step mutation is built on the framework of Multiple Correlated-Try Metropolis and can leverage multiple proposals in one step, generated cheaply from correlated bidirectional connections. We showed how our proposed large step mutation is no longer beneficial when render time is taken into account, but motivated a hybrid approach that could perform better.

In the second half of the thesis, we introduced the notion of the inverse path sample function, which can reconstruct the random numbers that generated a light path. We gave simple recipes for inverting sampling methods commonly used in graphics, and gave examples of sampling processes that cannot be inverted. We then showed two applications of such an inverse in the context of Markov Chain Monte Carlo integration in primary sample space, both targeting shortcomings of previous work.

To motivate our first approach, we showed how Multiplexed Metropolis Light Transport cannot easily transition between sampling techniques, since changes to the technique index lead to large changes to the light path. We then showed how this problem can be solved using the inverse path sample function, which allowed us to introduce a path-invariant technique perturbation that is able keep the light path fixed while switching sampling techniques. We analysed the performance of this method in detail

in Chapter 6 and showed how it improves upon previous work in a variety of scenes.

Finally, we also investigated how geometric and material discontinuities interact with MCMC methods operating in primary sample space. We gave an intuition as to why such discontinuities can prevent the Markov chain from properly exploring the state space with small steps, and proposed an alternative small step perturbation that uses the path sample inverse in order to step over such discontinuities. Our analysis showed that such an approach succeeds in preventing uncontrolled changes to the light path, but fails to improve rendering performance.

7.1 Future Work

Chapter 4 demonstrated several uses of random numbers in the path sampling process that can interact suboptimally with primary sample space MCMC methods. It would be interesting to investigate whether more such cases exist and whether their effect is large enough to warrant a solution.

In Section 4.3, we also showed how our proposed multiple correlated-try large steps are no longer beneficial when render time is taken into account. We suspect that they still hold an advantage over MMLT large steps for longer paths, and it would be interesting to investigate whether a combination of MMLT large steps for short paths and our proposed large steps for long paths improves overall rendering performance.

In Chapter 5, we only showed two possible uses of the inverse path sample function, but we strongly believe that it has other applications as well. These could be in the form of additional perturbations or mutations for MCMC methods in primary sample space, in a similar vein to the ones proposed. We showed how our alternative small step, although more robust to discontinuities, does not improve rendering performance. However, it demonstrates how direct operations on paths can be reconciled with the primary sample space perspective, and there could be other perturbations that can make use of such an approach. Finally, the path sample inverse is general enough that there might be applications in other areas of rendering, and it would be interesting to explore the implications of having such an inverse mapping available.

Bibliography

[AK90] James Arvo and David Kirk. Particle transport and image synthesis. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 63–66, New York, NY, USA, 1990. ACM.

[Bit14] Benedikt Bitterli. Tungsten renderer, 2014. https://github.com/tunabrain/tungsten.

[BM58] G. E.P̃. Box and Mervin E. Muller. A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, 29(2):610–611, 06 1958.

[CL07] Radu V. Craiu and Christiane Lemieux. Acceleration of the multiple-try metropolis algorithm using antithetic and stratified sampling. *Statistics and Computing*, 17(2):109–120, 2007.

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.

[CTE05] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Trans. Graph.*, 24(3):1186–1195, July 2005.

[Deb98] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Annual Conference Series (Proc. SIGGRAPH)*, pages 189–198, 1998.

[DHM+01] Philip Dutré, Paul Heckbert, Vincent Ma, Fabio Pellacini, Robert Porschka, Mahesh Ramasubramanian, Cyril Soler, and Greg Ward. Global illumination compendium, 2001.

[ENSB13] Christian Eisenacher, Gregory Nichols, Andrew Selle, and Brent Burley. Sorted deferred shading for production path tracing. In *Proceedings of the*

Eurographics Symposium on Rendering, EGSR '13, pages 125–132, Aire-la-Ville, Switzerland, Switzerland, 2013. Eurographics Association.

[ESG06] Manfred Ernst, Marc Stamminger, and Günther Greiner. Filter importance sampling. In *IEEE Symposium on Interactive Ray Tracing 2006*, pages 125–132, Sept 2006.

[GKH+13] Iliyan Georgiev, Jaroslav Křivánek, Toshiya Hachisuka, Derek Nowrouzezahrai, and Wojciech Jarosz. Joint importance sampling of low-order volumetric scattering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6), November 2013.

[GTGB84] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. *SIGGRAPH Comput. Graph.*, 18(3):213–222, January 1984.

[Has70] W.K̃. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[HH10] Jared Hoberock and John C. Hart. Arbitrary importance functions for metropolis light transport. In *Computer Graphics Forum*, pages 1993–2003, 2010.

[HJ11] Toshiya Hachisuka and Henrik Wann Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Trans. Graph.*, 30(5):114:1–114:11, October 2011.

[HKD14] Toshiya Hachisuka, Anton S. Kaplanyan, and Carsten Dachsbacher. Multiplexed metropolis light transport. *ACM Trans. Graph.*, pages 100–100, 2014.

[Ill91] Illumination Engineering Society of North America. *IES standard file format for electronic transfer of photometric data and related information*, 1991.

[Jak10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[Jak13] Wenzel Jakob. *Light Transport on Path-Space Manifolds*. PhD thesis, Cornell University, August 2013.

[Jar08] Wojciech Jarosz. *Efficient Monte Carlo Methods for Light Transport in Scattering Media*. PhD thesis, UC San Diego, September 2008.

[Kaj86] James T. Kajiya. The rendering equation. In *Proceedings of the 13th Annual*

Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '86, pages 143–150, New York, NY, USA, 1986. ACM.

[KSKAC02] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, pages 531–540, 2002.

[LKA13] Samuli Laine, Tero Karras, and Timo Aila. Megakernels considered harmful: Wavefront path tracing on gpus. In *Proceedings of the 5th High-Performance Graphics Conference*, HPG '13, pages 137–143, New York, NY, USA, 2013. ACM.

[LLW00] Jun S. Liu, Faming Liang, and Wing H. Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121+, March 2000.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional path tracing. In *Proceedings of 3rd International Conference on Computational Graphics and Visualization Techniques (COMPUGRAPHICS '93*, pages 145–153, 1993.

[LW96] Eric P. Lafortune and Yves D. Willems. Rendering participating media with bidirectional path tracing. In *Proceedings of the Eurographics Workshop on Rendering Techniques '96*, pages 91–100, London, UK, UK, 1996. Springer-Verlag.

[MRR+53] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[NRH+77] F. Nicodemus, J. Richmond, J. Hsia, I. Ginsberg, and T. Limperis. Geometric considerations and nomenclature for reflectance. Monograph 160, National Bureau of Standards (US), October 1977.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010.

[Pho75] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, June 1975.

[SIP07] Benjamin Segovia, Jean-Claude Iehl, and Bernard Proche. Coherent metropolis light transport with multiple-try mutations. Technical report,

LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/cole Centrale de Lyon, April 2007.

[Vea96] 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.

[Vea98] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, Stanford, CA, USA, 1998. AAI9837162.

[VG94] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Proceedings of Eurographics Rendering Workshop*, pages 147–162, 1994.

[VG95] 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, New York, NY, USA, 1995. ACM.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 65–76, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[Whi80] Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, June 1980.

[WMHTC65] E. Woodcock, T. Murphy, P. Hemmings, and L. T.C̃. Techniques used in the GEM code for monte carlo neutronics calculations in reactors and other systems of complex geometry. In *Proceedings of the Conference on Applications of Computing Methods to Reactor Problems*, 1965.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, EGSR '07, pages 195–206, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.