A Doctor Thesis

博士論文

Bridging Different Spaces in Light Transport Simulations

(異なる空間を繋ぐ光輸送シミュレーション)

by

Hisanari Otsu

大津 久平

Submitted to

the Graduate School of The University of Tokyo

on December 8, 2017

in Partial Fulfillment of the Requirements

for the Degree of Doctor of Information Science and Technology

in Creative Informatics

Thesis Supervisor: Toshiya Hachisuka　蜂須賀 恵也

Professor of Creative Informatics

ABSTRACT

The light transport simulation has been one of the major topics in computer graphics from the very beginning of the field, supported by insatiable demand of photorealistic image synthesis in various industries including movie industries, game industries, etc. The light transport simulation is formulated as solutions of the equation called rendering equation. The problem of numerically solving the rendering equation is called the *light transport problem*. We focus on the two problem domains in light transport: the *simulation*, and the *modeling* of the input of the light transport simulation. The simulation part of light transport simulation determines how we actually solve the rendering equation numerically, where the *efficiency* of the simulation is the major challenge. On the other hand, the modeling part of light transport simulation determines how the components of the input scene are mathematically defined, where the *accuracy* of resemblance to the actual physical objects is challenging.

Among the challenges of the light transport simulation, we focus on two problems. First, in the domain of simulation, we focus on improving efficiency of the light transport simulation using Monte Carlo methods, which is de facto standard approach for photorealistic rendering. Second, in the domain of modeling, we focus on improving accuracy of the spectral reflectance reconstruction from a tristimulus values, which is one of the important building blocks in spectral rendering because the commonly available data is defined in tristimulus values (e.g., RGB colors).

We tackled these problems by *bridging different spaces* in light transport simulation. In this thesis, we focus on three different spaces and proposed the methods to bridge the spaces using different approaches to achieve the goals.

1. The first method bridges the *strategy spaces* of light transport simulation which determine the solution space of the Monte Carlo rendering algorithms. The efficiency of the light transport simulation is known to be influenced by the selection of the rendering algorithms according to the input scene. However, the selection of the algorithms can be cumbersome because the users need to know the detail of each algorithm. The proposed method solves this issue by blending the solutions of the two different rendering algorithms using a machine learning approach.

2. The second method bridges the *state spaces* of Markov chain Monte Carlo (MCMC) rendering. Current MCMC rendering is built upon the two different state spaces with different mathematical background. While the two approaches are related by the sampling of transport paths, all existing MCMC rendering algorithms are designed to work within only one of the state spaces. We propose the first framework to bridge two state spaces. Using this framework, we can use mutation strategies designed for one space in the other space.

3. The third method bridges the *chromatic spaces* which determines the representation of spectra to improve the accuracy in the spectral reflectance reconstruction. The existing approaches in computer graphics are only based on the simple heuristics and ignores the actual shape of the measured spectra. Based on the observation from the

color science field, we propose a spectral reconstruction method that can faithfully reproduce the shape of the spectra as well as the converted tristimulus values. The method facilitates the bridge between the original and lower dimensional representation of the spectra obtained with the knowledge of the measured spectra.

From the results of three methods, we observed each of the methods faithfully achieve the goals of improving the efficiency or the accuracy in respective problem domains. We hope the insights from our study can be a step toward the future evolution of the light transport simulation.

論文要旨

光輸送シミュレーションは映画産業, ゲーム産業等, 様々な産業における写実的画像生成の飽くなき需要に支えられ, コンピュータグラフィックスの黎明から主な研究分野の一つである. 光輸送シミュレーションはレンダリング方程式と呼ばれる方程式の解として定式化され, レンダリング方程式を数値的に解くことは光輸送問題と呼ばれる. 光輸送シミュレーションに関連して, 本稿ではふたつの問題のクラスに着目した. ひとつはシミュレーションそのものの設計, もうひとつはシミュレーションの入力のモデル設計である. シミュレーション設計はレンダリング方程式を実際にどのように解くかに着目し, シミュレーションの**効率**を高めることが挑戦となる. 一方でモデル設計は入力となるシーンの構成要素がどのように数学的に定義されているかに着目し, いかにモデルの現実の物体に類似しているかを意味する物理的**正確さ**を達成するかが挑戦となる.

これらの光輸送シミュレーションにおける挑戦において, 本稿ではふたつの問題に着目した. ひとつはシミュレーション設計における問題で, 写実的レンダリングにおいて事実上の標準であるモンテカルロ法を用いた光輸送シミュレーションの効率向上を行うことである. もうひとつがモデル設計における問題で, RGB 等の三原色から分光反射率の再構築に着目する. これはスペクトルレンダリングにおいて重要な構成要素のひとつであり, 多くの利用可能なデータが三原色によって定義されるため必要となる.

本稿ではこれらの問題に対し, 光輸送シミュレーションにおける**異なる空間を繋ぐ**ことで解決を図った. 特に以下の 3 つの空間に着目し, ぞれぞれ異なる手法を提案した.

1. 一つ目の手法は, モンテカルロ法を用いたレンダリング手法の解空間によって定められる, 光輸送シミュレーションの**戦略空間** (strategy space) 同士を繋ぐ. 光輸送シミュレーションの効率は入力となるシーンに応じてレンダリング手法に依存することが知られているが, このような手法の選択は使用者が手法の詳細について知る必要があるため困難な場合がある. 提案法ではこの問題を機械学習を用いて 2 つの異なる手法の解を混ぜ合わせることで解決を図った.
2. 二つ目の手法は, マルコフ連鎖モンテカルロ法 (MCMC 法) を用いたレンダリング手法における**状態空間** (state space) 同士を繋ぐ. 現在の MCMC 法を用いたレンダリング手法は主に二つの異なる数学的背景を持つ状態空間上で定義される. これら二つのアプローチは光の経路のサンプリングで関連しているが, ふたつを組み合わせることはできない. 提案法ではこの 2 つの状態空間を繋ぐ手法をはじめて提案する. 提案するフレームワークを用いることにより, ひとつの状態空間で定義された変異手法をもう一つの空間で使用できるようになる.
3. 三つ目の手法は, 分光反射率の再構築の精度を向上させるために, スペクトルを表現する**色彩空間** (chromatic space) 同士を繋ぐ. コンピュータグラフィックスで用いられている既存手法は簡易なヒューリスティックを用いて再構築を行うが, 実際のスペクトルの形を考慮しない手法であった. 提案法では色彩工学の知識を用いる

ことにより, 再構築されたスペクトルから得られる三原色とともにスペクトルの形もうまく再現できる手法を提案する. 本手法ではオリジナルのスペクトル表現と測定されたスペクトルの情報に基づいた低次元のスペクトル表現を繋ぐことを活用した.

これら 3 つの手法の結果から, 各々の問題のクラスにおいて効率, 正確の目標をそれぞれ達成していることが観察された. 本研究から得られた知見が将来の光輸送シミュレーションの発展に寄与することを期待する.

Acknowledgements

First of all, I want to express my deepest gratitude to my advisor Toshiya Hachisuka who inspires my own interest in the compute graphics and guides me throughout my PhD study. He always helps me with stimulating discussions and suggesting interesting research directions through a number of research projects in my PhD period. From him, I could learn a lot of knowledge on the research on computer graphics as well as how to organize myself as a researcher. I am sure my current sense of research is owed to his great attitude to the research.

I would like to thank the thesis committee members: Masayuki Inaba, Shigeru Chiba, Takeo Igarashi, Takayasu Matsuo, Hideki Nakayama for their insightful comments on the thesis from the point of view in the various research fields.

I would like to thank my colleagues in Computer Graphics Group in The University of Tokyo, including Sadashige Ishida, Issei Takarada, Sabyasachi Mukherjee, Jamorn Sriwasansak, Igor Zavialov, Rex West, and many others. I really like the group and I pretty much enjoyed daily conversations among the members. Especially, I greatly thank Adrien Gruson for the discussion on the research, Masafumi Yamamoto for collaborating one of the research projects contained in the thesis.

During my PhD study, I was very fortunate to have opportunities to visit two research groups: Compute Graphics Group at Karlsruhe Institute of Technology and Charles University in Prague. I am grateful to Carsten Dachsbacher and Jaroslav Křivánek for giving me a chance to visit their groups. I also would like to thank the group members for the daily conversation or the discussion on the research during my visit. I also have great opportunities to collaborate with researchers on several research projects. I am grateful to Martin Šik, Shinichi Kinuwaki, Anton Kaplanyan, Johannes Hanika for the fruitful discussion and the guidance to the higher quality in the collaborated projects.

Last but by no means least, I would like to thank my parents for their support and encouragement over years, as well as during my PhD period. The random talks with my parents is pretty relaxing to me. My thesis would not be possible without their sincere support and encouragement.

Contents

Chapter 1

Introduction

1.1 Motivation

1.1.1 Background

Rendering is a long-standing concept in the field of computer graphics from the very beginning of the field. In an abstract view, the rendering, or more specifically 3D rendering, can be considered as a process of computation which inputs a description of the virtual world called the *scene*, and outputs a 2D image which provides a depiction of the virtual world. Rendering is imperative in computer graphics because every computer-generated model or animation eventually needs to be converted to the visible and human-perceptible formats. The description of the virtual world contains all the information necessary to generate an image, such as the shapes representing the surface geometries, the material properties of the scene surfaces, the emission properties of the light sources, or the sensor information describing how we capture the virtual world through a virtual camera.

Persistent sophistication in research and development in rendering makes it possible to arise an paradigm called the photo-realistic rendering. Photo-realistic rendering is a sub-field in rendering that aims to achieve the images as similar as the real world like a real photograph. A major approach for photo-realistic rendering is to simulate real-world phenomena as accurate as possible based on the laws of physics. As a natural consequence, this resemblance is only truly possible by a deep understanding on the underlying physical effects how the real-world phenomena happens and perceived by human. On the other hand, from the practical perspective, it is also important to properly model the physical effects into the feasible manner suited to the computation. Apparently, it is not an easy task and this is why the photo-realistic rendering attracts major attention and has been a strong motivator of the research field over the last decades.

Achieving photo-realistic rendering via physically based approaches by simulating the propagation of lights in the virtual world is called light transport simulation.

In the context of rendering research, light transport simulation is initially formalized as rendering equation by Kajiya [58] and further sophisticated by Veach [112]. The equation is written as form of an integral equation, describing the change of incoming light energy by the scattering on the surfaces. Common solution for the problem is to use numerical integration techniques using random sampling such as Monte Carlo method or Markov chain Monte Carlo method. Utilizing random sampling based numerical integration techniques is beneficial for the light transport simulation, because it can achieve the predictable output, the stable and discretization-free algorithms agnostic to surface topologies, or the better convergence properties especially in the high dimensional integration, which is often the case in light transport simulation.

1.1.2 Challenges

In this thesis, we focus on light transport simulation based on the numerical integration with random sampling. The complexity of light transport simulation, however, creates the remaining issues that has not been solved despite of the rigorous endeavor. A notable example includes the rendering of the scene containing complex scene geometry or the materials that can introduce tricky light transport such as the intereflection between glossy materials.

We focus on the challenges of light transport simulation in the two research directions: (1) improving *efficiency in simulation*, (2) improving *accuracy in modeling*. The simulation in light transport simulation concerns how we efficiently solve light transport simulation as an integral equation. This direction of the research includes for instance developing new statistical methods, acceleration structures, or rendering algorithms. On the other hand, the modeling in light transport simulation focuses on how we accurately develop an physical models of materials, light sources, sensors, or even the mechanism of light transport itself. This direction includes for instance the development of the new materials, the theory of light transport mainly by means of the knowledge of physics.

1.2 Contribution

1.2.1 Bridging Representative Spaces

In the highest abstraction, the fundamental idea of the approaches introduced in this thesis is to resolve the problems related to light transport simulation by means of *bridging representative spaces* inherent to the problem formulation.

In this thesis, we define the representative spaces as a set of spaces describing the fundamental objects representing a concept used in the problem definitions, which is

	Bridging strategy spaces (Chapter 5)	*Bridging state spaces* (Chapter 6)	*Bridging chromatic spaces* (Chapter 7)
Challenge	Efficiency in simulation	Efficiency in simulation	Accuracy in modeling
Problem	MC rendering	MCMC rendering	Spectral reflectance reconstruction
Bridge	Combining rendering techniques	Combining mutation strategies	Connecting representations of measured spectra
Approach	Machine learning	Inverse mapping	Data compression

Table 1.1: Overview of the methods introduced in the thesis. The three methods are summarized by four elements, the target challenge in simulation or modeling (*Challenge*), the target problem domain (*Problem*), the concrete description of the bridging spaces (*Bridge*), and the approach to achieve the method (*Approach*).

often naturally arises from the formulation of the problem definitions. The representative spaces is typically described as a form of the formulation using the language of mathematics, or as a form of the technical concepts. For instance, the tristimulus color spaces, a set of spaces representing a color with three components, such as RGB or XYZ color spaces, is a representative spaces because the color space as its component is used to represent the same concept — the color perceptible to human.

Given the definition, *bridging* representative spaces means to solve an interest problem facilitating the relationship among the spaces as a component of the representative spaces. Recalling the color space example, we can convert tristimulus colors between two color spaces by a matrix multiplication. If a problem of interest facilitates an another color space different from the space of the input, we can say the solution is obtained by bridging the representative spaces.

1.2.2 Problem Definition And Representative Spaces

Among the challenges categorized in Sec. 1.1.2, we focus on the two concrete problems corresponding to each of the categories: (1) improving the efficiency in (Markov chain) Monte Carlo rendering, and (2) improving the accuracy in the spectral reflectance reconstruction. Given the problem definition, we consider the three representative spaces inherent to the specific problems: strategy spaces, state spaces, and chromatic spaces. The summary of the spaces and the corresponding methods are summarized in Tab. 1.1.

1.2.3 Original Contribution

Bridging Strategy Spaces of Light Transport Simulations The first method bridges *strategy spaces* of light transport simulations which determine the solution space of the Monte Carlo rendering algorithms. In light transport simulation, because we solve the same governing equations for light transport, different algorithms all converge to the same result. However, since different algorithms have different efficiencies depending on input scene configurations, a user would try to find the most efficient algorithm based on trials and errors. This selection of an algorithm can be cumbersome because a user needs to know technical details of each algorithm. We propose a framework which blends the results of two different rendering algorithms, such that a blending weight per pixel becomes automatically larger for a more efficient algorithm. Our framework utilizes a popular machine learning technique, regression forests, for analyzing statistics of outputs of rendering algorithms and then generating an appropriate blending weight for each pixel. The key idea is to determine blending weights based on classification of path types. This idea is inspired by the same common practice in movie industries; an artist composites multiple rendered images where each image contains only a part of light transport paths (e.g., caustics) rendered by a specific algorithm. Since our framework treats each algorithm as a black-box, we can easily combine very different rendering algorithms as long as they eventually generate the same results based on light transport simulation. The blended results with our algorithm are almost always more accurate than taking the average, and no worse than the results with an inefficient algorithm alone.

Bridging State Spaces of Markov Chain Monte Carlo Rendering The second method bridges *state spaces* of Markov chain Monte Carlo (MCMC) rendering. Rendering algorithms using Markov chain Monte Carlo (MCMC) currently build upon two different state spaces. One of them is the path space, where the algorithms operate on the vertices of actual transport paths. The other state space is the primary sample space, where the algorithms operate on sequences of numbers used for generating transport paths. While the two state spaces are related by the sampling procedure of transport paths, all existing MCMC rendering algorithms are designed to work within only one of the state spaces. We propose a first framework which provides a comprehensive connection between the path space and the primary sample space. Using this framework, we can use mutation strategies designed for one space with mutation strategies in the respective other space. As a practical example, we take a combination of manifold exploration and multiplexed Metropolis light transport using our framework. Our results show that the simultaneous use of the two state spaces improves the robustness of MCMC rendering. By combining efficient local exploration in the path space with global jumps in primary sample space, our method achieves more uniform convergence as compared to using only one space.

Bridging Chromatic Spaces For Spectral Reflectance Reconstruction The third method bridges *chromatic spaces* which determines the representation of spectra to improve the accuracy in the spectral reflectance reconstruction. Physically based rendering systems often support spectral rendering to simulate light transport in the real world. Material representations in such simulations need to be defined as spectral distributions. Since commonly available material data are in tristimulus colors, we ideally would like to obtain spectral distributions from tristimulus colors as an input to spectral rendering systems. Reproduction of spectral distributions given tristimulus colors, however, has been considered an ill-posed problem since single tristimulus color corresponds to a set of different spectra due to metamerism. We show how to resolve this problem using a data-driven approach based on measured spectra and propose a practical algorithm that can faithfully reproduce a corresponding spectrum only from the given tristimulus color. The key observation in color science is that a natural measured spectrum is usually well approximated by a weighted sum of a few basis functions. We show how to reformulate conversion of tristimulus colors to spectra via principal component analysis. To improve accuracy of conversion, we propose a greedy clustering algorithm which minimizes reconstruction error. Using precomputation, the runtime computation is just a single matrix multiplication with an input tristimulus color. Numerical experiments show that our method well reproduces the reference measured spectra using only the tristimulus colors as input.

1.3 Thesis Organization

The thesis is divided into seven chapters. The rest of six chapters are organized as follows. In Chapter 2, we present the basic concept of the light transport, introducing the mathematical model for light transport simulation. Chapter 3 introduces the general overview of Monte Carlo integration. Chapter 4 presents the numerical solution of light transport simulation, especially the overview of the light transport algorithms based on path sampling. In Chapter 5, 6, and 7, we present the proposed methods as described in Sec. 1.2.3. Finally, Chapter 8 concludes the thesis with some possible idea for the future work.

Chapter 2

Theory of Light Transport

2.1 Introduction

In the computer graphics field, *rendering* or *image synthesis* is a process of computation generating an image, a two-dimensional data structure typically composed of a set of pixels, given a scene as an input. The scene is a data structure describing the artificial environment of the three-dimensional world, which is typically composed of an representation of the geometry, the description of the appearance of the objects, and the information of light or camera. Specifically, the photo-realistic rendering focus on generating the images as similar to the depiction of the real world as possible. In order to achieve the photo-realism, the apparent approach is the physically-based approach, where we focus on the raw of physics to simulate the reality. The current development of the photo-realistic rendering is based on the physically-based approach. The phenomena in the world can be described by the raws of physics, including the phenomena on the light, which is undoubtedly the most important physical entity for the physically-based rendering. Based on the context, the term of light transport coincides with the physical aspect of the transportation of light, which is emitted from the light source, propagated on the three-dimensional space by interacting the object surfaces, and measured by the human eyes or the sensor of the camera. The algorithms for generating photo-realistic images by simulating the transportation of light is called the light transport simulation. The real-world applications of the light transport simulation is vast, which includes movie production, game production, or the visualization related to the architecture or the engineering. In this chapter, we will introduce the basic concepts of the light transport especially the knowledge on how we physically model the light transport in mathematics.

2.2 Domains And Measures

Theory of light transport is written as a form of integral with respect to various domains and the corresponding measures. Also, the physical quantities used in the

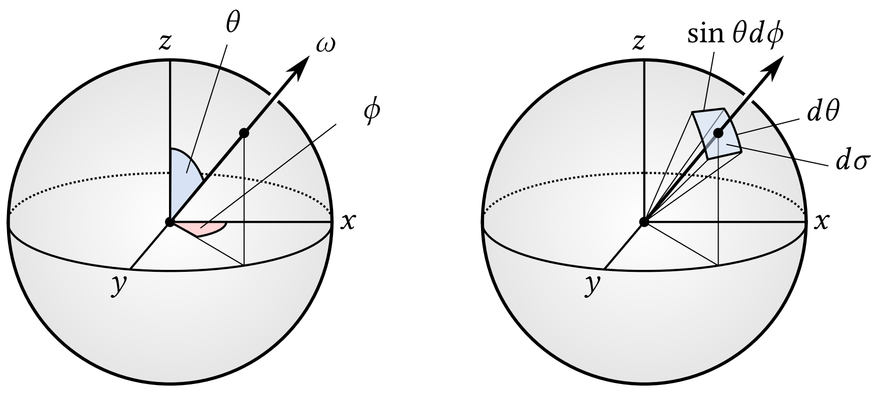


Figure 2.1: Spherical coordinates (left) and the relationship between the differential solid angle $d\sigma$ and the differential area on the spherical coordinates $d\theta d\phi$ (right). Geometrically, the differential solid angle can be represented by a product of the differential lengths of the two edges defined by the arc lengths on the spherical coordinates fixing one coordinate: $\sin\theta\, d\phi$ and $d\theta$.

light transport simulation are defined with various measures. In this section, before diving into the actual explanation of the light transport theory, we will introduce common domains and measures which are often used in the context of light transport simulation, as well as the relationship between these domains and measures.

We first define the *scene surface* $\mathcal{M} \subset \mathbb{R}^3$ as an union of all the surfaces of the objects. We represent a *point* on the surface as bold letters like $\mathbf{x} \in \mathcal{M}$. We can define the measure with respect to the area of the scene surface $\mathcal{M}$, called the *area measure dA*. Using the measure, we can define the integral over the scene surface $\mathcal{M}$ defined for instance as $\int_{\mathcal{M}} f(\mathbf{x}) dA(\mathbf{x})$, which is often seen in the integral used in the light transport simulation. We note that, in the context of the light transport simulation, the scene surface $\mathcal{M}$ is not necessarily a differentiable manifold. For instance, we can easily assume the scene containing two spheres contacting one point, which is known to be non-manifold. Practically, we can ignore these pathetic parts of the geometries because the integral over the null measure (in this case, the parts like area measure is zero, like contacting points or lines) becomes zero. Still the contacting planes are a problem, but at least we can select one of the object surface contacting each other to make the entire scene surface measurable for almost every set except for the null measures [112]. In the context of differential geometry, the surface integral on manifold can be defined as an integral of the differential 2-form. Using this kind of definitions, more rigorous handling of the integral is possible [68], however, we will stay in the definition of the surface integral with a naive measure theoretic definition of the area measure.

In addition to the integral with respect to the area, we often use the integral with respect to the *directions*. The directions are represented by a normalized vector ω on the unit sphere $\mathcal{S}^2$. As a measure on the domain, we define the *solid angle measure*

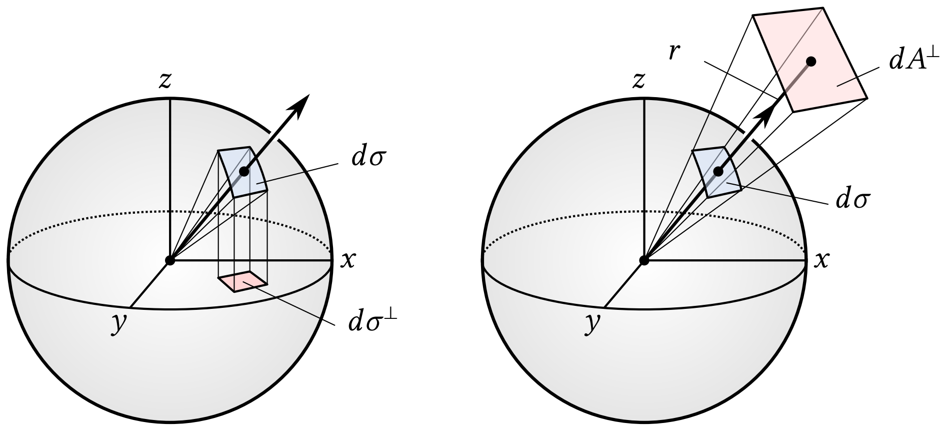


Figure 2.2: Projected solid angle (left) and the relationship between the differential area $dA^{\perp}$ and the differential solid angle $d\sigma$ (right). The relationship can be described by projecting dA onto the unit sphere. Two differential elements are related by multiplying the inverse square distance from the center of the unit sphere $1/r^2$, which intuitively means the change in the proportion in the distance $1/r$ is squared when we consider the proportion in the area.

$d\sigma$ which measures the area on the sphere $\mathcal{S}^2$. This implies we can consider the solid angle measure as an area measure defined on the unit sphere. The directions can be also represented by the spherical coordinates (θ, ϕ) where θ is the azimuthal angle and ϕ is the zenith angle (Fig. 2.1, left). The relationship between the solid angle measure $d\sigma$ and the differential area on the spherical coordinate $d\theta d\phi$ can be written by multiplying the differential arc lengths around the direction (Fig. 2.1, right). Therefore, we obtain

$$d\sigma = \sin\theta \, d\theta d\phi. \tag{2.1}$$

Moreover, we can consider the area or solid angle measure projected on the different surface. This kind of measures are useful in the light transport simulation because the light illuminates from the perpendicular direction to the surface. The *projected solid angle measure* $d\sigma^{\perp}$ is defined as the measure on the projected area of the solid angle onto the tangent plane (Fig. 2.2, left). The relationship between the solid angle measure $d\sigma$ and the projected solid angle measure $d\sigma^{\perp}$ can be written as

$$d\sigma^{\perp} = |\cos\theta| \, d\sigma, \tag{2.2}$$

where θ is the angle between the direction ω and the surface normal $\mathbf{n}$. Alternatively, we can project the differential area dA to the plane perpendicular to the direction ω. The *projected area measure* $dA^{\perp}$ is defined in this way as

$$dA^{\perp} = |\cos\theta| \, dA, \tag{2.3}$$

with the same angle θ as in Eq. 2.2. From these equations, therefore we can obtain the identity on the two product measures:

$$dA^{\perp} d\sigma = |\cos\theta|\, dA d\sigma = dA d\sigma^{\perp}. \tag{2.4}$$

The area measure and the solid angle measure is related by considering the projection of the differential area perpendicular to direction to the surface on the unit sphere (Fig. 2.2, right). Therefore, the solid angle measure $d\sigma$ can be written as

$$d\sigma = \frac{dA^{\perp}}{r^2} = \frac{|\cos\theta'|}{r^2} dA \tag{2.5}$$

where r is the distance between the center of the sphere to the surface and θ' is the angle between the surface normal on the point at the area measure of an interest. We note that the angle θ' is different from the angle θ because θ' is the angle defined for the surface normal around the other differential area projected to the differential solid angle. From this equation, we can also obtain the relationship between the area measure and the projected solid angle measure:

$$d\sigma^{\perp} = |\cos\theta|\, d\sigma = \frac{|\cos\theta||\cos\theta'|}{r^2} dA, \tag{2.6}$$

which we will later used as a definition of the geometry factor.

2.3 Radiometry

Radiometry is a study field of the measurement of the electromagnetic radiation. In the light transport simulation, we are interested in the behavior of the visible light, which is also a kind of the electromagnetic radiation. Radiometry is based on the geometric optics, where we consider the light as a movement of the rigid particles. This means some of the physical phenomena related to the wave property of the light, such as diffraction or interferences, where the field of the wave optics handles, cannot be captured in the formulation based on the radiometry. Recent development of the light transport theory [68] shows rigid relationship between the formulation of the light transport and the electromagnetic theory and derived the complete formulation starting from the Maxwell equation. In this thesis, however, we will merely introduce the classical handling of the light transport based on the radiometry.

Specifically, the field of studies on the measurement of visible electromagnetic radiation in terms of the human perception. The radiometric quantities have corresponding equivalent in the photometry, like the luminance to the radiance. Although the most of interests on light transport simulation is in the visible wavelength, we

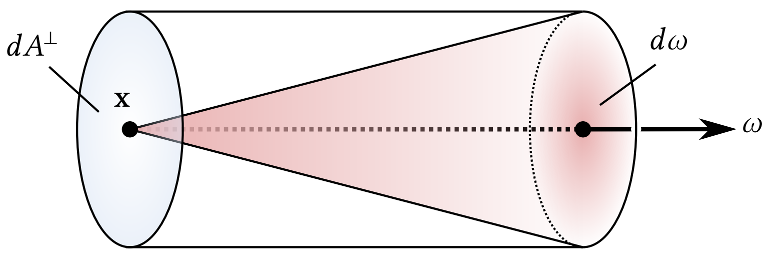


Figure 2.3: Radiance along with the direction ω from the position $\mathbf{x}$ is defined as an amount of light energy traveling through within the tube defined by the projected differential area $dA^{\perp}$ perpendicular to ω and the differential solid angle $d\omega$.

will stick to the use of radiometric terms in the following discussions.

Radiance Now we will introduce the basic quantities utilized to describe the light transport theory. Especially we will focus on the *radiance*, a basic quantity used to describe the light transport. The radiance is a fundamental quantity in light transport simulation because the sensors such as human eyes or a camera are sensitive to the radiance. The radiometric quantities are defined as derivatives of the electromagnetic energy Q, including the radiance. Given a point $\mathbf{x}$ and a direction ω, the radiance $L(\mathbf{x}, \omega)$ is a five-dimensional function defined as the amount of energy passed through a projected differential area $dA^{\perp}$ on a hypothetical plane (cross section) perpendicular to the differential solid angle $d\omega$ in the unit time (Fig. 2.3). Using derivatives, the radiance $L(\mathbf{x}, \omega)$ is written as

$$L(\mathbf{x}, \omega) = \frac{dQ}{dt dA^{\perp} d\omega} = \frac{d\Phi}{dA^{\perp} d\omega}, \tag{2.7}$$

where $\Phi = dQ/dt$ is radiance power defined as electromagnetic energy per unit time. If the radiance is measured on the surface point $\mathbf{x}$ with a surface normal $\mathbf{n}$, the projected differential area $dA^{\perp}$ can be written as $dA^{\perp} = dA \cos\theta$, where θ is the angle between ω and $\mathbf{n}$. Intuitively, the cosine term of the equation can be described as a term to capture the increase in amount of energy on the unit surface according to the increase of the grazing angle.

Alternatively, in order to achieve more rigid handling of the radiance, we can define $dA^{\perp}$ as a differential 2-form [68]. In this formulation, the radiance becomes a three component vectors whose element represents the flow of the flux through the surface element perpendicular to each x, y, z axis. In this case, the scalar variant of the radiance is obtained as a pullback of the 2-form onto the chart on the surface. The operation naturally introduces the cosine term in $dA^{\perp} = dA \cos\theta$.

The radiance is known to be invariant along straight line, as long as the light travels through a vacuum, the space without a participating media. This characteristic is important for rendering because we can express the radiance by a sequence of the surface points.

Given the definition of the radiance, the other radiometric quantities can be written as an integral of the radiance. For instance, the total flux Φ on the specific total surface area A and the specific total solid angle Ω can be expressed with the double integral of the radiance:

$$\Phi = \int_A \int_\Omega L(\mathbf{x}, \omega) d\phi(\omega) dA(\mathbf{x}). \tag{2.8}$$

We distinguish the incident and exitant radiances. The incident radiance $L_i(\mathbf{x}, \omega)$ measures the radiance arriving at the position $\mathbf{x}$ from the direction ω. On the other hand, the exitant radiance $L_o(\mathbf{x}, \omega)$ measures the radiance leaving from the position $\mathbf{x}$ to the direction ω. In a vacuum without participating media, the two quantities are related by

$$L_i(\mathbf{x}, \omega) = L_o(\mathbf{x}, -\omega). \tag{2.9}$$

We note that this separation is essential, because the radiance on the surface can be discontinuous on the point on the surface, that is, there exists the case the for $\mathbf{x} \in \mathcal{M}$,

$$L^+(\mathbf{x}, \omega) \neq L^-(\mathbf{x}, \omega), \tag{2.10}$$

where L^+ and L^- is the radiance defined in above and below the surface. These functions are written with the one-side limits respect to the surface normal $\mathbf{n}$ as

$$L^+(\mathbf{x}, \omega) = \lim_{t \longrightarrow +0} L(\mathbf{x} + t\mathbf{n}, \omega), \tag{2.11}$$

$$L^-(\mathbf{x}, \omega) = \lim_{t \longrightarrow -0} L(\mathbf{x} + t\mathbf{n}, \omega). \tag{2.12}$$

Alternatively, these quantities can be also defined with the limits of time in photon trajectories [112]. Using these one-side limits of the radiance, we can intuitively define the incident and exitant radiances as

$$L_i(\mathbf{x}, \omega) = \begin{cases} L^+(\mathbf{x}, -\omega), & \omega \cdot \mathbf{n} > 0 \\ L^-(\mathbf{x}, -\omega), & \omega \cdot \mathbf{n} < 0 \end{cases} \tag{2.13}$$

$$L_o(\mathbf{x}, \omega) = \begin{cases} L^+(\mathbf{x}, \omega), & \omega \cdot \mathbf{n} > 0 \\ L^-(\mathbf{x}, \omega), & \omega \cdot \mathbf{n} < 0 \end{cases} \tag{2.14}$$

One can easily check Eq. 2.9 when $\mathbf{x}$ in a point in a space, because L is continuous so $L^+ = L^-$ holds. In the following discussions, for simplicity, we sometimes omits the subscripts i and o from L_i or L_o and just write L instead unless the ambiguity concerns.

Surface Scattering The energy of light emitted from the light source travels along the ray of light by reflecting or transmitting on the object surfaces, where some part of the energy of light is absorbed and dissipated as a heat. The characteristics of describing how the energy of light reflect on the surface is called the *reflectance*. The reflectance dominates the appearance of the object. For instance, let us consider an object with red color illuminated with a light source with white color. The object looks red because the surface of the object absorbs the energy corresponding to the wavelengths.

In order to introduce how we define such a reflectance property in the language of mathematics. Here we want to assume several assumptions. First, we assume the interaction of lights with the fixed wavelength, that is, the case that the wavelength does not change due to the interaction of light and ignores wavelength-dependent effects such as fluorescence. Second, we assume the the interaction happens only on the single point on the surface, that is, the interaction of light is only happens in terms of the single point on the surface. This means we ignore the volumetric effect on the subsurface, where the interaction involves in multiple points on the surface and we consider the energy of light is transmitted through under the surface. This kind of surface interaction is called the subsurface scattering.

Given these assumptions, the reflectance can be describe as the bidirectional reflectance distribution function (BRDF), a function with respect to a surface point $\mathbf{x}$, the incident direction ω_i and the outgoing direction ω_o (Fig. 2.4), is defined as

$$f_r(\mathbf{x}, \omega_i, \omega_o) = \frac{dL_o(\mathbf{x}, \omega_o)}{L_i(\mathbf{x}, \omega_i) d\omega_o^{\perp}} = \frac{dL_o(\mathbf{x}, \omega_o)}{L_i(\mathbf{x}, \omega_i) \cos\theta d\omega_o}. \tag{2.15}$$

In the case that the light is transmitted into the surface, that is, when ω_i and ω_p are in the opposite side of the surface, the reflectance function is called the bidirectional transmittance distribution function (BTDF) f_t. We call the function combining BRDF and BTDF as the bidirectional scattering distribution function (BSDF) f_s. In the definition of BSDF, the cosine term in Eq. 2.15 is substituted by $|\cos\theta|$ in order to support the two cases in one equation.

Spectral Quantities More fundamentally, the aforementioned quantities can also be defined with respect to wavelengths. For instance, *spectral radiance* can be defined as

$$L_\lambda(\mathbf{x}, \omega, \lambda) = \frac{d\Phi}{dA^{\perp} d\omega d\lambda} = \frac{dL(\mathbf{x}, \omega)}{d\lambda}, \tag{2.16}$$

The other radiometric quantities can be similarly extended to the spectral quantities. Specifically, spectral radiance is the fundamental quantity in radiometry because the other quantities can be derived from the integration of spectral radiance. The spectral quantities are important for *spectral rendering*, the light transport simulation

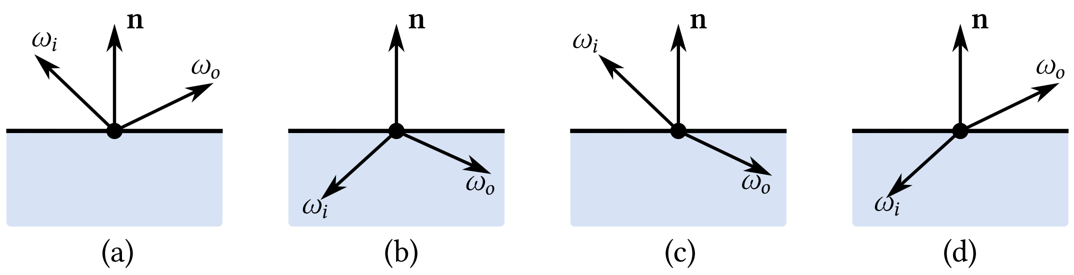


Figure 2.4: Bidirectional scattering distribution function (BSDF) is four-dimensional function defined with respect to the incoming and outgoing directions ω_i and ω_o. Specifically, the function is called BRDF if ω_i and ω_o are in the same side with respect to the surface normal $\mathbf{x}$ (a,b). On the other hand, if ω_i and ω_o are in the opposite side with respect to $\mathbf{n}$, the function is called BTDF (c,d).

modelded with the full wavelength instead of using tristimulus color spaces like RGB color space.

2.4 Colors

In this section we will briefly introduce the concept in color science related to light transport simulation. It is important to consider proper handling of colors in rendering system, because the final output of light trnsport simulation is a *color image*.

Spectral Representation Fundamentally, physical representation of colors is originated from the spectral quantities defined as a function of wavelengths. The domain of wavelength or the function mapped from that domain is called the *spectrum*, by context representing the wavelength-dependent radiometric quantities such as spectral radiance, spectral reflectance, and so on. If the wavelengths are in the visible range that humans can perceive, the spectrum is called the *visible spectrum*. The visible wavelengths are around 380 to 730 nm.

Human Perception of Color Human can perceive the incoming light as color by different types of cells in *retina*. Retina is an inner part of the eye that is composed of the light-sensitive photoreceptor cells. It is known that there are several different types of photoreceptor cells in retina such as *cone cells* or *rod cells*.

Cone cells perceive the light with relatively high brightness and are mainly responsible for the color perception of the human eye. Cone cells are furthers categorized into three types according to the spectrum that the cells can perceive: S-cones, M-cones, and L-cones and each cone is responsible for the ranges of short, midium, and long wavelengths respectively. It is also known that the cone cells have faster responce time than the other photoreceptor cells, which enables us to perceive the rapid change of the scene, especially in bright places. The existence of three different cones explains why the human vision is said to be *trichromatic*.

On the other hand, rod cells are more sensitive than the cone cells and perceive the light with relatively low intensity. Therefore rod cells are mainly responsible for night vision. Unlike cone cells, rod cells is composed of only type of photoreceptor cells so they do little contribution to the color vision. This explains why we find it hard to distinguish colors at the dark places.

Tristimulus Color Spaces As we explained, human vision system is trichromatic. Therefore we can characterize the human perception of colors with three values, in order to model the color that human can perceive. These three values are called the *tristimulus values* and the *tristimulus color space* defines the representation of color by tristimulus values. Here we will describe the relashionship between the spectrum and various representations of the colors with tristimulus values.

CIE XYZ color space is the standard tristimulus color space that links between the visible spectrum and the human-perceived colors. The color space is created and standardized by International Commission on Illumination (CIE) [102] based on the series of experiments independently conducted by Wright and Guild [123, 31]. The data obtained with the experiments by Wright and Guild are initially used to specify an another color space (CIE RGB color space) and based on that color space CIE XYZ color space is derived.

The participants of the experiments are asked to match the test colors with a mixture of three three monochromatic primary colors representing red (700 nm), green (546.1 nm), and blue (435.8 nm). By using the monochromatic colors represented by a single wavelength as test colors, we can obtain three spectra of the coefficients required to match the test colors for each wavelength. This is possible under the assumption of the principal known as *Grassman's law*, which empirically explains any colors can be composed of a linear superposition of the monochromatic colors. The three spectra obtained in this way is called the *color matching functions*. By collecting the measurements we can define the virtual observer who has an averaged color matching function, known as *standard observer*. Given the color matching functions $\bar{r}(\lambda)$, $\bar{g}(\lambda)$, and $\bar{b}(\lambda)$ corresponding to the primary colors, the RGB tristimulus values for the spectrum $S(\lambda)$ can be obtained as

$$R = \int_\Lambda S(\lambda)\bar{r}(\lambda)d\lambda,\ G = \int_\Lambda S(\lambda)\bar{g}(\lambda)d\lambda,\ B = \int_\Lambda S(\lambda)\bar{b}(\lambda)d\lambda, \tag{2.17}$$

where Λ is a range of the visible wavelengths. We note that the integral represents a general form of Grassman's law, considering a Riemann sum of the integral. We also note that the color matching functions are normalized to exclude the effect of brightness of the primaries:

$$\int_\Lambda \bar{r}(\lambda)d\lambda = \int_\Lambda \bar{g}(\lambda)d\lambda = \int_\Lambda \bar{b}(\lambda)d\lambda = \int_\Lambda V(\lambda)d\lambda, \tag{2.18}$$

where $V(\lambda)$ is the luminosity function. CIE also scandalized human perception to the brightness as a luminosity function $V(\lambda)$, which approximates the human response to the brightness. The function is used to convert the radiance energy to luminous energy: $V = \int_\lambda S(\lambda)V(\lambda)d\lambda$. The conversion from the RGB tristimulus values to the luminance is derived with a least square fit of the mixed primaries. Note that this derivation assumes the assumption that the human perception is roughly linear to the human response. The coefficients can be written as

$$(a', b', c') = \operatorname*{argmin}_{a,b,c} \int_\Lambda \left(a\bar{r}(\lambda) + b\bar{g}(\lambda) + c\bar{b}(\lambda) - V(\lambda)\right)^2 d\lambda. \tag{2.19}$$

According to the CIE standard, these values are $(a', b', c') = (0.17697, 0.81240, 0.01063)$. Using these coefficients, the luminance V can be approximated by the RGB tristimulus values as $V \approx \int_\Lambda S(\lambda)\left(a'\bar{r}(\lambda) + b'\bar{g}(\lambda) + c'\bar{b}(\lambda)\right) d\lambda = a'R + b'G + c'B$.

Based on aforementioned CIE RGB color space, CIE XYZ color space is carefully designed to have good characteristics as a standard tristimulus color space, mainly to reduce the computational burden for the users. The space is induced by a new set of color matching functions $\bar{x}(\lambda)$, $\bar{y}(\lambda)$, and $\bar{z}(\lambda)$. The characteristics of the space are as follows. (1) The color matching functions are non-negative. This property fixed the problem that the color matching functions of CIE RGB color space include negative parts and the resulting components of the tristimulus values can include the negative values. (2) The color matching function $\bar{y}(\lambda)$ is exactly equal to the luminocity function $V(\lambda)$. (3) The tristimulus values are selected to have coordinates with equal values for the constant energy spectrum. The constant energy spectrum defines a flat spectrum that has the same values at any wavelengths. This means the *white point*, the tristimulus values that defines the *white* color, for the constant energy spectrum has a coordinates (X, Y, Z) with $X = Y = Z$. (4) The all possible colors in the space (*gamut*) are included in the triangle of $(1, 0)$, $(0, 0)$, and $(0, 1)$ in the chromaticity diagram. Along with the CIE XYZ color space, CIE xyY color space is introduced to separate the chromatic and brightness components in the tristimulus values. The chromaticity diagram is defined as a projection of the space into the xy plane. For a brightness term, we can use Y terms according to the property (2). For a chromatic terms, the space introduces the normalized values of XYZ and use two of them as coodinates, because the rest can be obtained with subtracting the sum of the two from one. The normalized values are defined as

$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = 1 - x - y. \tag{2.20}$$

The detailed derivation of the CIE XYZ color space given the conditions that satisfies these properties can be found in the literature [24].

sRGB Color Space Various other tristimulus color spaces can be derived from CIE XYZ color space. For instance, we can derive notable sRGB color space [105] through a series of transformations from XYZ tristimulus values. First we consider the transformation between XYZ and linear sRGB color space. The transformation of linear sRGB shares the same specification as ITU-R BT.709 standard [111]. According to the standard, the chromaticity coordinates of the three primaries corresponding to red (R), green (G), and blue (B) are respectively defined as

$$\begin{bmatrix} x_R \\ y_R \end{bmatrix} = \begin{bmatrix} 0.640 \\ 0.330 \end{bmatrix}, \quad \begin{bmatrix} x_G \\ y_G \end{bmatrix} = \begin{bmatrix} 0.300 \\ 0.600 \end{bmatrix}, \quad \begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} 0.150 \\ 0.060 \end{bmatrix}. \tag{2.21}$$

The standard assumes the D65 illuminant to define the white color. The chromaticity coodinates that all components of the RGB values are equal to unity (while point) under the D65 illuminant is also defined as

$$\begin{bmatrix} x_W \\ y_W \end{bmatrix} = \begin{bmatrix} 0.3127 \\ 0.3290 \end{bmatrix}. \tag{2.22}$$

The luminance is defined as $V = 0.2126R + 0.7152G + 0.0722B$. Therefore the Y coordinates for each primary can be written as $Y_R = 0.2126$, $Y_G = 0.7152$, and $Y_B = 0.0722$. We assume the luminance of the while point is $Y_W = 1$. According to Eq. 2.20, the XYZ coordinates for each primary and while point can be calculate as $X = xY/y$, $Z = (1 - x - y)Y/y$ and Y remains the same between two spaces. Using these values, the conversion from RGB to XYZ coordinates can be written as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} X_R & X_G & X_B \\ Y_R & Y_G & Y_B \\ Z_R & Z_G & Z_B \end{bmatrix} \begin{bmatrix} \alpha_R & 0 & 0 \\ 0 & \alpha_G & 0 \\ 0 & 0 & \alpha_B \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \tag{2.23}$$

where the coefficients α_i control the scaling of the each basis and determines the factor contributes to brightness of each primary. Note that this scaling operation does not alter the chromacity of the colors. Substituting $[R\,G\,B]^T = [1\,1\,1]^T$ and $[X\;Y\;Z]^T = [X_W\;Y_W\;Z_W]^T$, we obtain the coefficients α_i by solving the linear equation:

$$\begin{bmatrix}\alpha_R \\ \alpha_G \\ \alpha_B\end{bmatrix} = \begin{bmatrix}X_R & X_G & X_B \\ Y_R & Y_G & Y_B \\ Z_R & Z_G & Z_B\end{bmatrix}^{-1} \begin{bmatrix}X_W \\ Y_W \\ Z_W\end{bmatrix}. \tag{2.24}$$

Substituting the coefficients α_i in Eq. 2.23 for Eq. 2.24, we finally obtain the conversion matrix from linear sRGB to CIE XYZ color space. The inverse conversion from XYZ to sRGB can be obtained by calculating inverse of the matrix.

It is known that human perceives the intensity of light by the distribution similar to power function (Stevens's power law). Facilitating this nature, the display device are designed to maximize the visual quality by reducing the required bandwidth of the input signal using the *compressed* information instead of the original information. This process of encoding or decoding the tristimulus values are called the *gamma correction*. The sRGB color space explicitly defines the average gamma value of 2.2. Therefore the aforementioned linear sRGB colors need to be converted according to the average gamma, to obtain the final colors in the sRGB color space. The function to encode the linear RGB colors are called the transfer function, which is defined in sRGB color space as

$$C_{\mathrm{sRGB}} = \begin{cases} 12.92\ C_{\mathrm{linear}} & C_{\mathrm{linear}} \le 0.0031308 \\ 1.055\ C_{\mathrm{linear}}^{1/2.4} - 0.055 & \text{otherwise,} \end{cases} \tag{2.25}$$

where C_{linear} is a component of the linear sRGB colors and C_{sRGB} is the corresponding output. We note that the transformation is designed to approximate the gamma compression with gamma 2.2 with a piecewise-defined functions. This explains why the transformation uses the exponents of 2.4 instead of 2.2. The linear parts are introduced to prevent the numerical problems that power function can have extremely large slope when C_{linear} is near zero.

2.5 Rendering Equation

The *rendering equation* describes the equilibrium distribution of the energy of light, describing how a distribution of the scattered light changes according to the surface property. The equation is originally introduced to the compute graphics field by Kajiya [58], which has now became as a fundamental equation in light transport simulation and every light transport simulation algorithm solves the equation to obtain the image. Continuing to the definition of BSDF, we assume the light travels in a space with no participating media, thus involves in no scattering in the free space. The equation for volume scattering originated from the radiative transfer equation [13] is also famous in light transport simulation, yet in this thesis we will focus on merely the surface scattering.

The rendering equation formulates the outgoing radiance $L_o(\mathbf{x}, \omega_o)$ at the surface point $\mathbf{x}$ to the direction ω_o. Considering the conservation of energy of total incident and outgoing radiance, the outgoing radiance can be written as a sum of the emitted radiance $L_e(\mathbf{x}, \omega_o)$ and the scattered radiance $L_s(\mathbf{x}, \omega_o)$:

$$L_o(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega_o) + L_s(\mathbf{x}, \omega_o). \tag{2.26}$$

The scattered radiance can be obtained from the definition of BSDF by the integration over the sphere of the incident direction around $\mathbf{x}$:

$$L_s(\mathbf{x}, \omega_o) = \int_{\mathcal{S}^2} f_s(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) d\sigma^{\perp}(\omega_i), \tag{2.27}$$

where $\mathcal{S}^2$ is a unit sphere representing a set of directions around the point $\mathbf{x}$, and $d\sigma^{\perp}$ is the projected solid angle measure. This form of the equation is also called the scattering equation. Substituting L_s in Eq. 2.28, we obtain the *light transport equation*:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) d\sigma^{\perp}(\omega_i). \tag{2.28}$$

Recursive Formulation Here we want to convert the rendering equation in the recursive form according to the outgoing radiance $L_o(\mathbf{x}, \omega_o)$. Under the assumption that the light travels in the vacuum, the incoming radiance can be represented by a function of the position $\mathbf{x}$ and the outgoing direction ω_o using the outgoing radiance:

$$L_i(\mathbf{x}, \omega) = L_o(\chi_{\mathcal{M}}(\mathbf{x}, \omega), -\omega). \tag{2.29}$$

Here $\chi_{\mathcal{M}}(\mathbf{x}, \omega)$ is the ray-casting function that returns the closed surface along the direction ω originated from the position $\mathbf{x}$, that is, along the ray $(\mathbf{x}, \omega)$. Specifically the function is defined as

$$\chi_{\mathcal{M}}(\mathbf{x}, \omega) = \mathbf{x} + \underbrace{\inf\{d > 0 \mid \mathbf{x} + d\omega \in \mathcal{M}\}}_{d_{\mathcal{M}}(\mathbf{x},\omega)} \cdot \omega, \tag{2.30}$$

where $\mathcal{M}$ is a set of scene surfaces and $d_{\mathcal{M}}(\mathbf{x}, \omega)$ means the distance to the next surface. For convenience, we append the singleton $\{\infty\}$ to the set applied by inf in $d_{\mathcal{M}}$. If there is no intersecting surfaces, the set becomes $\emptyset$ therefore $\chi_{\mathcal{M}}(\mathbf{x}, \omega)$ returns ∞ because $\inf \emptyset = \infty$. Substituting Eq. 2.29 for Eq. 2.28, we obtain the recursive form of the rendering equation:

$$L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{\mathcal{S}^2} f_s(\mathbf{x}, \omega_i, \omega_o) L_o(\chi_{\mathcal{M}}(\mathbf{x}, \omega_i), -\omega_i) d\sigma^{\perp}(\omega_i). \tag{2.31}$$

Area Formulation The rendering equation in Eq. 2.31 is formulated as an integral with respect to the project solid angle measure. By means of changing the measure, we can obtain the alternative form of the equation to consider the integral over the scene surfaces, instead of a set of directions on the unit sphere. Sometimes this kind of formulation is useful, for instance, when we want to consider the integral over the area light sources.

The projected solid angle measure $d\omega^{\perp}$ can be converted to the area measure dA by the Jacobian, specifically called the *geometric term* $G(\mathbf{x} \leftrightarrow \mathbf{y})$ in the rendering context:

$$G(\mathbf{x} \leftrightarrow \mathbf{y}) = \left|\frac{d\sigma^{\perp}}{dA}\right| = V(\mathbf{x} \leftrightarrow \mathbf{y}) \cdot \frac{|\mathbf{n}_{\mathbf{x}} \cdot \overrightarrow{\mathbf{xy}}||\mathbf{n}_{\mathbf{y}} \cdot \overrightarrow{\mathbf{yx}}|}{\|\mathbf{x} - \mathbf{y}\|^2}, \tag{2.32}$$

where $\mathbf{y}$ and $\mathbf{y}$ are the points on the scene surface with the surface normals $\mathbf{n}_{\mathbf{x}}$ and $\mathbf{n}_{\mathbf{y}}$ respectively. $\overrightarrow{\mathbf{xy}}$ is the direction from $\mathbf{x}$ to $\mathbf{y}$, and $V(\mathbf{x} \leftrightarrow \mathbf{y})$ is a visibility function defined as

$$V(\mathbf{x} \leftrightarrow \mathbf{y}) = \begin{cases} 1, & \text{if } \{t\mathbf{x} + (1-t)\mathbf{y} \mid t \in (0,1)\} \cap \mathcal{M} = \emptyset \\ 0, & \text{otherwise.} \end{cases} \tag{2.33}$$

Intuitively, the visibility function becomes unity when the points $\mathbf{x}$ and $\mathbf{y}$ is mutually visible. Also, the visibility function can be computed by the ray-casting function $\chi_{\mathcal{M}}$: $V(\mathbf{x} \leftrightarrow \mathbf{y}) = 1$ if and only if $\chi_{\mathcal{M}}(\mathbf{x}, \overrightarrow{\mathbf{xy}}) = \mathbf{y}$. Using these equations, we can rewrite Eq. 2.31 with respect to the function of the consequent intersection points $\mathbf{x}, \mathbf{y}, \mathbf{z}$ by letting $\mathbf{y} = \mathbf{x}, \omega_o = \overrightarrow{\mathbf{yz}}, \omega_i = \overrightarrow{\mathbf{yx}}$. This form of the rendering equation is called the *three point form* of the rendering equation, which reads

$$L(\mathbf{y} \longrightarrow \mathbf{z}) = L_e(\mathbf{y} \longrightarrow \mathbf{z}) + \int_{\mathcal{M}} f_s(\mathbf{x} \longrightarrow \mathbf{y} \longrightarrow \mathbf{z}) L(\mathbf{x} \longrightarrow \mathbf{y}) G(\mathbf{x} \leftrightarrow \mathbf{y}) dA(\mathbf{x}). \tag{2.34}$$

The equation is now described only with the points on the surfaces, instead of the points and the directions. We introduced some notations to specify the function depends only on the surface points like $f_s(\mathbf{x} \longrightarrow \mathbf{y} \longrightarrow \mathbf{z}) = f_s(\mathbf{y}, \overrightarrow{\mathbf{yx}}, \overrightarrow{\mathbf{yz}})$.

Measurement Equation The purpose of the light transport simulation is to compute measurements of the light energy that contribute to each pixel. Eq. 2.28,2.31,2.34 formulates the distribution of the radiance, yet we actually need to perform the *measurement* of the radiance on the sensor to obtain the pixel values from the radiance distribution. We let the measurements for j-th pixels be I_j $(j = 0, \ldots, M)$. We can formulate the measurement of the radiance by the measurement equation defined as

$$I_j = \int_{\mathcal{M} \times \mathcal{S}^2} W_e^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) dA(\mathbf{x}) d\sigma_{\mathbf{x}}^{\perp}(\omega). \tag{2.35}$$

We assume the sensor is a part of the scene surface. $W_e^{(j)}(\mathbf{x}, \omega)$ is the sensor responsibility function corresponding to the pixel j, which is also called as the importance. We often drop the superscript (j) if we don't need to point to the specific pixels. The function describes how a sensor changes its response according to the position and the direction on the sensor. We note that the importance is the dual concept as the radiance and we can even obtain the transport equation for the importance similar to Eq. 2.28,2.31,2.34. This characteristic is a theoretical basis for the adjoint methods [112], which means we can solve the light transport problem by solving the transport equation for the importance, as well as by solving the equation for radiance.

The measurement equation in Eq. 2.35 can also be expressed with respect to the integral over the scene surfaces, applying similar procedure that the area formulation of the light transport equation (Eq. 2.34) is obtained from the directional variant (Eq. 2.31):

$$I_j = \int_{\mathcal{M}\times\mathcal{M}} W_e^{(j)}(\mathbf{x} \longrightarrow \mathbf{y}) L(\mathbf{x} \longrightarrow \mathbf{y}) G(\mathbf{x} \leftrightarrow \mathbf{y}) dA(\mathbf{x}) dA(\mathbf{y}), \tag{2.36}$$

where $\mathbf{y} \in \mathcal{M}$ is a point on the scene surface and we used the notation $W_e^{(j)}(\mathbf{x} \longrightarrow \mathbf{y}) = W_e^{(j)}(\mathbf{x}, \overrightarrow{\mathbf{xy}})$ as well as the notations in Eq. 2.34.

Path Integral Formulation Facilitating the predefined formulations, we can obtain the another form of the light transport equation known as the *path integral formulation* of the light transport. The purpose of the path integral formulation is to express the light transport equations with a simple form of the integral instead of the form of the recursively defined integral equations. In this formulation, the integral is defined with respect to the set of *light transportation paths*, or simply called the *paths*, which is represented by the sequence of surface points.

The main benefit of the path integral formulation is the formulation enables us to apply the general-purpose integration methods to solve the light transport simulation. Also the formulation detaches the paths from the way of its construction. Using this formulation, the structure of the paths are now independent from the information how they are actually constructed, for instance, the paths are traced from the side of the sensor or the light. This property leads to the bidirectional approaches [112], one of the fundamental approaches in the current development of the light transport simulations.

In order to obtain the path integral formulation, starting from the measurement equation in Eq. 2.36, we recursively expand the radiance term $L(\mathbf{x} \longrightarrow \mathbf{y})$ using the three-point form of the light transport equation (Eq. 2.34):

$$
\begin{aligned}
I &= \int_{\mathcal{M}^2} L_e(\mathbf{x}_0 \to \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) W_e(\mathbf{x}_0 \to \mathbf{x}_1) dA(\mathbf{x}_0) dA(\mathbf{x}_1) \\
&+ \int_{\mathcal{M}^3} L_e(\mathbf{x}_0 \to \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) f_s(\mathbf{x}_0 \to \mathbf{x}_1 \to \mathbf{x}_2) \\
&\qquad \times G(\mathbf{x}_1 \leftrightarrow \mathbf{x}_2) W_e(\mathbf{x}_1 \to \mathbf{x}_2) dA(\mathbf{x}_0) dA(\mathbf{x}_1) dA(\mathbf{x}_2) + \dots \\
&= \sum_{n=1}^{\infty} \underbrace{\int_{\mathcal{M}^{n+1}} f(\mathbf{x}_0, \cdots, \mathbf{x}_n) dA(\mathbf{x}_0) \cdots dA(\mathbf{x}_n)}_{I_n}.
\end{aligned}
\tag{2.37}
$$

Now the pixel contribution I can be expressed by the infinite sum of the integral I_n, which is a contribution from the paths with the number of vertex $n + 1$ or the length n. Here we call the length of the path by the number of edges between the vertices, not by the actual distance that the light travels. $f(\mathbf{x}_0, \cdots, \mathbf{x}_n)$ is the function called the measurement contribution function, which is composed of the product of the geometric terms and the BSDFs evaluated at every vertex, defined as

$$
\begin{aligned}
f(\mathbf{x}_0, \cdots, \mathbf{x}_n) &= L_e(\mathbf{x}_0 \to \mathbf{x}_1) G(\mathbf{x}_0 \leftrightarrow \mathbf{x}_1) \\
&\times \left(\prod_{i=1}^{n-1} f_s(\mathbf{x}_{i-2} \to \mathbf{x}_{i-1} \to \mathbf{x}_i) G(\mathbf{x}_{i-1} \leftrightarrow \mathbf{x}_i) \right) W_e(\mathbf{x}_{n-1} \to \mathbf{x}_n)
\end{aligned}
\tag{2.38}
$$

We will rewrite the pixel contribution I with respect to the integral on the *path space*. The path space is defined as a set of all possible paths with any lengths. The path is defined as a sequence of points on the scene surface. We represent a path containing the vertices $(\mathbf{x}_0, \dots, \mathbf{x}_n)$ by the notation $\bar{x} = \mathbf{x}_0 \dots \mathbf{x}_n$. Specifically, the path space Ω is defined as

$$
\Omega = \bigcup_{n=1}^{\infty} \Omega_n, \text{ where } \Omega_n = \{\mathbf{x}_0 \dots \mathbf{x}_n \mid \mathbf{x}_0, \dots, \mathbf{x}_n \in \mathcal{M}\}. \tag{2.39}
$$

Ω_n is a set of paths with the length n. Corresponding to the path space, we also define the *path space measure* μ by the area-product measure [38]:

$$
d\mu(\bar{x}) = dA(\mathbf{x}_0) dA(\mathbf{x}_1) \cdots dA(\mathbf{x}_n). \tag{2.40}
$$

Using these equations, we can rewrite the per-length pixel contribution I_n in Eq. 2.37 to the well-known path integral form of the light transport equation as

$$
I_n = \int_{\Omega} f(\bar{x}) d\mu(\bar{x}). \tag{2.41}
$$

Now we can write the rendering equation with a form of a simple integral. According to the context, we sometimes drop the subscript n when we do not need to focus on the specific length of the paths. In the following discussion in this thesis, we rely on this formulation unless stated otherwise.

Chapter 3

Monte Carlo Integration

3.1 Introduction

In this chapter, we will introduce the general overview of Monte Carlo integration, a mathematical tool for numerical solutions the light transport simulation. As shown in Chapter 2, the light transport simulation can be formulated as a form of an integral. In principle, we can consider to apply various integration methods to solve the integral. The longstanding research and development of light transport simulation shows, however, Monte Carlo method is a current de facto standard approach and occupies a dominant position of the current research and development of the light transport simulation. This fact is mainly due to the possibility of handling of the high dimensional integral, which is often the case with light transport simulation. The deterministic integration methods are not free from the curse of dimensionality and the computational cost grows exponentially to the number of dimensions [21]. On the other hand, the computational cost in Monte Carlo methods does not depend on the number of dimensions. Given this context, in this chapter, we will introduce the general overview of the Monte Carlo method including the Markov chain Monte Carlo method. Followed by this chapter, we will introduce the solution techniques of the light transport simulation facilitating the Monte Carlo method.

3.2 Integration With Monte Carlo Method

Monte Carlo integration is one of the numerical integration techniques using random sampling. Current approaches in light transport simulation largely relies on Monte Carlo integration due to its strength to the high-dimensional integrals. We suppose the target integral that we want to estimate as

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

where f is a real-valued function defined on Ω, and μ is some measure on Ω. The basic idea of Monte Carlo integration is to estimate the integral as a problem of finding the expected values. For instance, we consider a random variable $y = f(x)/p(x)$ where a random variable x is sampled from the probability distribution with the probability density function (pdf) $p(x)$. We denote the random variable distributed over the density function $p(x)$ as the notation $x \sim p(x)$. The expected value of $y = f(x)/p(x)$ can be calculated as

$$\mathbb{E}_p[y] = \int_\Omega \underbrace{\frac{f(x)}{p(x)}}_{y} p(x) d\mu(x) = \int_\Omega f(x) d\mu(x) = I. \tag{3.2}$$

In this sense, the random variable y is an *estimate* of the integral I. However, this estimate y is practically useless because the variance can be large. The quality of an estimate can be measured by calculating the variance, the quantity that indicates how a random variable is distributed around the expected value $\mathbb{E}_p[y]$:

$$Var_p[y] = \mathbb{E}_p\left[(y - \mathbb{E}_p[y])^2\right] = \int_\Omega (y - \mathbb{E}_p[y])^2 p(x) d\mu(x). \tag{3.3}$$

Practically, instead of the one-sample estimate y, we utilize N number of independent random variables $(x_1, \ldots, x_n)$ identically distributed to the distribution $p(x)$ and considers an empirical average of $f(x_i)/f(x_i)$ as an estimate:

$$y_N = \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}. \tag{3.4}$$

We can easily verify the expected value of y_N matches I from the linearity of the expected value because the random variable x_i is mutually independent:

$$\mathbb{E}_p[y_N] = \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_p\left[\frac{f(x_i)}{p(x_i)}\right] = \frac{1}{N} \sum_{i=1}^{N} I = I. \tag{3.5}$$

We note that the estimate y_N will converge to $\mathbb{E}_p[y_N] = I$ almost surely as $N \longrightarrow \infty$ according to the strong law of large number (SLLN). Moreover, the convergence rate of the estimate y_N with respect to N can be obtained from the central limit theorem (CLT). According to the theorem, $\sqrt{N} \cdot (y_N - I)$ is converged in distribution to a normal distribution $\mathcal{N}(0, Var_p[y_N])$.

Error and Bias We will introduce two quantities to assess the quality of an estimate: the error and the bias. The error of the N-sample estimate y_N is defined as $y_N - I$. The bias of the estimate y_N is defined as the expected value of the error:

$$B[y_N] = \mathbb{E}_p[y_N - I] \tag{3.6}$$

The estimate y_N is called *unbiased* if $B[y_N] = 0$ for all N, i.e., $\mathbb{E}_p[y_N] = I$ for all N. On the other hand, the estimate y_N is called *consistent* if y_N is converged to I in probability as $N \longrightarrow \infty$, that is, for all $\varepsilon > 0$.

$$\lim_{N \longrightarrow \infty} Pr\left(|y_N - I| > \varepsilon\right) = 0. \tag{3.7}$$

Although both properties are preferable as an estimate, we note that unbiasedness and consistency is orthogonal concepts.

3.3 Variance Reduction

In order to increase the efficiency of the Monte Carlo integration, it is important to reduce variance of the estimate. Here we will briefly introduce several variance reduction techniques which is often used in the light transport simulation.

3.3.1 Sample Placement

First approach is to control the placement of the samples as uniform as possible over the integration domain. The stratified sampling is one of such a variance reduction techniques. The idea behind the stratified sampling is to subdivide the sample space into several non-overlapping regions. That is, the sample space Ω is subdivided into m disjoint sub-spaces $\Omega_i (i = 1, \dots, m)$ called stratum, and we generate the random samples $x_{(i,j)}$ corresponding to each Ω_i. Although the stratified sampling achieves higher convergence ratio than the ordinary sampling, the strategy is not effective on a high dimensional space because the number of stratum and the number of samples required for each stratum grows exponentially. The Latin hypercube sampling [75] alleviates the problem by generating samples so that only one sample is assigned for each row and each column. The quasi-Monte Carlo (QMC) [85] generalizes the Latin hypercube sampling and utilizes a low discrepancy sequence instead of a sequence of random numbers for estimating an integral. A low discrepancy sequence is deterministically chosen so that the samples are uniformly distributed over the sample space.

3.3.2 Importance Sampling

The importance sampling is a technique to reduce the variance by properly choosing the probability density function p. The selection of the distribution p has considerable influence on the variance of the estimate. In the light transport simulation, properly designing the sampling strategy is important because it can reduce the variance drastically.

The intuitive idea of the importance sampling is to design the distribution p as similar to the integrand f as possible. Specifically, it is known that the optimal choice of the distribution p^* that minimizes the variance is written as

$$p^*(x) = \frac{|f(x)|}{\left|\int_\Omega f(x) d\mu(x)\right|} = \frac{|f(x)|}{|c|}. \tag{3.8}$$

In this case, the variance of the estimate becomes always zero. This equation is, however, impractical because we must know the normalization constant c which is the same as the target integral.

3.3.3 Multiple Importance Sampling

Multiple importance sampling (MIS) proposed by Veach [116] is an importance sampling technique which combines multiple samples from the different probability densities. The technique is useful for an integrand that is difficult to sample with a single distribution, for instance the integrand containing multiple modalities. MIS is also useful for light transport simulation because we can combine the paths constructed with multiple strategies. Such a combination can alleviate a trade-off between sampling strategies, for instance by combining two complementary strategies that one strategy is effective for certain lighting effect but the other strategy is effective for a different light effect.

We assume the combined probability densities as $p_i (i = 1, \ldots, N)$ and the weighing function associated with the distribution p_i as w_i. The estimate for multiple importance sampling called the multi-sample estimate is defined as

$$\hat{I}_{\text{MIS}} = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_i} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})}, \tag{3.9}$$

where N_i is a number of samples taken from the distribution p_i and $x_{i,j} \sim p_i$. w_i is the weighting function corresponding to the i-th strategy. Although we can assign the arbitrary number of samples for each strategy, from the practical point of view in the context of the light transport simulation, we often assign the number samples per strategy $N_i = 1$.

The estimate $\hat{I}_{\text{MIS}}$ is unbiased if following conditions on weights are satisfied: (1) for all x, $f(x) \neq 0 \implies \sum_{i=1}^{N} w_i(x) = 1$ and (2) for all x, $p_i(x) = 0 \implies w_i(x) = 0$. These conditions implies at least one strategy must generate the sample x with $f(x) \neq 0$. This fact can be shown in the following argument: if we assume there is no strategy can generate the sample x, that is, $p_i(x) = 0$ for all i, by the condition (2) we get $w_i(x) = 0$ and by the contraposition of the condition (1) the value of $f(x)$ should be zero but actually $f(x) \neq 0$ from the assumption. In other words, this fact can be stated that the support of the function f must be included in the union of the support of the distribution p_i:

$$\operatorname{supp}(f) \subseteq \bigcup_{i=1}^{N} \operatorname{supp}(p_i). \tag{3.10}$$

Veach also proposed how to design the weighting function w_j. The ideal goal of designing weighting function is to select the function so that the variance of the resulting estimate is minimum. This is generally hard problem because we can combine arbitrary densities. Veach proposed the weighting functions called *balance heuristics* defined as

$$w_i(x) = \frac{N_i p_i(x)}{\sum_{k=1}^{N} N_k p_k(x)}. \tag{3.11}$$

He also proposed the general version of the balance heuristics parametrized by $\beta > 2$, called as *power heuristics*:

$$w_i(x) = \frac{N_i p_i(x)^\beta}{\sum_{k=1}^{N} N_k p_k(x)^\beta}. \tag{3.12}$$

Especially balance heuristics is known to have a moderately good variance bound even compared to the best estimate possible. This property is stated for any multi-sample estimate $\bar{I}$ as an inequality

$$Var[\hat{I}_{\mathrm{MIS}}] - Var[\bar{I}] \leq \left(\frac{1}{\min_i N_i} - \frac{1}{\sum_i N_i} \right) \mu, \tag{3.13}$$

where $\mu = \mathbb{E}[\hat{I}_{\mathrm{MIS}}] = \mathbb{E}[\bar{I}]$.

3.4 Integration With Markov Chain Monte Carlo Method

In this section we will introduce the Markov chain Monte Carlo method (MCMC), which is an another class of numerical integration method using random samples. Instead of using independent samples as in MC method, MCMC utilizes the correlated sequence of samples. Specifically, the correlated sequence is generated sequentially only based on the previous samples, which is called the Markov chain.

The use of MCMC can alleviate the challenging situation for the ordinary MC methods that the shape of the integrand is complex. In the context of the light transport simulation, MCMC helps to sample the difficult paths such as the paths generated for the scene containing difficult visibilities, or path contribution due to the material selection, which is often said to be difficult to handle with ordinary importance sampling.

The basic idea of MCMC is to use a correlated sequence of random variables called a Markov chain, which is a sequence of random variable whose samples depend only on samples one before. Specifically, a Markov chain is defined as a sequence of random variables $x_1, x_2, \cdots$ such that x_{i+1} depends only on $x_i (i = 1, 2, \ldots)$, that is, $X_{i+1} \sim K(x_{i+1}|x_i)$ where $K(y|x)$ is a conditional probability density function called the *transition kernel.* The class of algorithms which generates *stationary* Markov chain according to some distribution is called Markov chain Monte Carlo (MCMC). A Markov chain is called stationary if there exists a distribution π such that

$$\pi(x) = \int_\Omega K(y \longrightarrow x)\pi(y)d\mu(y). \tag{3.14}$$

In order to obtain a stationary Markov chain, many of MCMC algorithms utilize a stronger condition called *detailed balance condition.* A transition kernel K satisfied the detailed balance condition if there exists a distribution π for all $x, y \in \Omega$,

$$K(y|x)\pi(x) = K(x|y)\pi(y). \tag{3.15}$$

A Markov chain satisfying this condition is called a *reversible* Markov chain, and we can show that in this case the distribution π is stationary.

In order to numerically compute Eq. 3.1 using Monte Carlo integration with MCMC, we want to generate samples according to some distribution proportional to f. In order to achieve the goal, we will introduce the Metropolis-Hastings (MH) algorithm [77, 41], which is initially developed MCMC algorithm and often used in various field including the light transport simulation. The transition kernel for the algorithm, as known as Metropolis-Hastings update, is defined as follows procedure. Given the current sample x_i, we first choose a tentative sample $x'_i \sim T(x'_i|x_i)$, where $T(y|x)$ is transition kernel. Here we select the next sample according to the probability $a(x_i, x'_i)$. The next sample x_{i+1} is define as

$$x_{i+1} = \begin{cases} x_i, & \text{with probability } a(x'_i|x_i), \\ x'_i, & \text{otherwise,} \end{cases} \tag{3.16}$$

where the probability $a(x_i, x'_i)$ is the *acceptance ratio* defined as

$$a(x_i \longrightarrow x'_i) = \min\left(1, \frac{f(x')T(x|x')}{f(x)T(x'|x)}\right) \tag{3.17}$$

We note that Eq. 3.16 implies the integrand f need not to be normalized, and we can observe the transition kernel associated with the algorithm

$$K(y|x) = a(x, y)T(y|x) + \left(1 - \int_\Omega a(x, y)T(y|x)d\mu(y)\right)\delta_x(y) \tag{3.18}$$

satisfies Eq. 3.15 with f as the stationary distribution.

Convergence We can show that if a Markov chain satisfies the following two conditions, the chain converges to the target distribution: (1) *f-irreducible* which implies that any part of the state space can be reached from the selection of the initial state with finite mutations, and (2) $apediodic$ which implies the Markov chain does not contain cycles, a sequence of samples is converged to be distributed according to the stationary distribution. Moreover, similar to SLLN for i.i.d. random number case, the expected value of the N-sample estimate y_N converges to I even when a Markov chain satisfying above conditions is used as a sequence of samples, which is known as (weak) Ergodic Theorem. We also note that MH update implicitly satisfies these conditions with respect to the target distribution f and the transition function.

Chapter 4

Numerical Solutions of Light Transport

In this chapter, we will introduce the several light transport simulation techniques and related concepts fundamental in the latest approaches. Especially, we focus on the approaches using Monte Carlo method based on path sampling.

4.1 Early History

4.1.1 Deterministic to Stochastic

The *ray-tracing* is known as the origin of the light transport simulation. The algorithm generates an image by tracing a set of rays from the side of the viewer and determines the intersected surface position and the color. The basic idea of the algorithm is initially described by Appel [4] for the hidden surface removal. The idea is further extended to the recursive evaluation of the specular surfaces by Whitted [121]. The use of random sampling in the light transport simulation is initially attempted by Cook et al. [19]. They extended Whitted's ray-tracing by the random sampling in order to support the distributed effects such as the soft shadow, the depth of field, or the motion blur. Hereafter, the development of the light transport simulation began to focus on the global illumination algorithm, which is capable of handling the lighting effects involving the indirect light transport.

4.1.2 FEM Based Rendering

Attempts to the indirect light transport are initiated from the approaches employing the finite element method (FEM). FEM is an general numerical integration techniques making use of the partitioning of the integration domain. Goral et al. [27] initially introduced the concept to the graphics community based on the literature of the heat transfer, which is called the *radiosity* method. The radiosity method first subdivides the scene surfaces into a set of patches. Utilizing FEM, the method computes the indirect light transport between patches.

Assuming the scene surfaces are Lambertian diffuse surfaces, Eq. 2.34 can be rewritten to the transport equation with respect to the radiosity $B(\mathbf{x})$, one of the the radiometric quantities representing the exitant energy per unit area:

$$B(\mathbf{x}) = E(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int_{\mathcal{M}} B(\mathbf{x}')G(\mathbf{x} \leftrightarrow \mathbf{x}')dA(\mathbf{x}'), \tag{4.1}$$

where $E(\mathbf{x})$ is the radiant exitance at $\mathbf{x}$ and $\rho(\mathbf{x})$ is the diffuse reflectance at $\mathbf{x}$. Note that the BSDF for Lambertian diffuse surface is defined as $f_s(\mathbf{x}' \longrightarrow \mathbf{x} \longrightarrow \mathbf{x}'') = \rho(\mathbf{x})/\pi$, where the term $1/\pi$ ensures energy conservation. We assume the surface geometry $\mathcal{M}$ is approximated by N small patches $\{\mathcal{M}_i\}_{i=1,\dots,N}$ where $\mathcal{M} \approx \cup_{i=1}^{N} \mathcal{M}_i$. Using this partition, we can approximate the radiosity $B(\mathbf{x})$ by a linear weighted sum of the basis functions $N_i(\mathbf{x})$:

$$B(\mathbf{x}) \approx B_{\text{approx}}(\mathbf{x}) = \sum_{i=1}^{N} B_i N_i(\mathbf{x}), \tag{4.2}$$

where $N_i(\mathbf{x})$ has a compact support on $\mathcal{M}_i$, meaning the non-zero support of the function $N_i(\mathbf{x})$ is defined only on the patch $\mathcal{M}_i$. Although various selection of the basis function is possible [107], for the sake of simplicity, we will use the constant basis function defined as $N(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathcal{M}_i$ and $N(\mathbf{x}) = 0$ otherwise.

The basic idea of FEM is to obtain the weights $\{B_i\}_{i=1,\dots,N}$ to achieve the minimum error between $B(\mathbf{x})$ and $B_{\text{approx}}(\mathbf{x})$. We are tempted to solve the minimization problem on the residual $r(\mathbf{x}) = B_{\text{approx}}(\mathbf{x}) - B(\mathbf{x})$ but unfortunately it is known to be intractable. Instead of directly minimizing the residual $r(\mathbf{x})$, we will minimize the weighted residual:

$$\langle r(\mathbf{x}), W_i(\mathbf{x}) \rangle = \int_{\mathcal{M}} r(\mathbf{x}) W_i(\mathbf{x}) dA(\mathbf{x}), \tag{4.3}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product between the functions, and $W_i(\mathbf{x})$ is the basis function having an local support on $\mathcal{M}_i$. The general approach of using the weighted residual is called the weighted residual method.

Among the various selection of the weighting function $W_i(\mathbf{x})$, we use the Galerkin method. The method utilizes the basis function $N_i(\mathbf{x})$ as the weighting function. The condition for the minimization of the weight residual then becomes

$$\underbrace{\int_{\mathcal{M}} r(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x})}_{(*)} = 0, \text{ for all } i. \tag{4.4}$$

Expending LHS (*) with Eq. 4.2 and Eq. 4.3, we obtain a linear equation according the weights B_i called the radiosity equation:

$$B_i + \frac{\rho_i}{\pi} \sum_{j=1}^{N} B_j F_{i,j} = E_i, \tag{4.5}$$

where ρ_i is the diffuse reflectance and E_i is the exitant emittance both associated to the patch $\mathcal{M}_i$. The term $F_{i,j}$ is called the form factor. The term describes the fraction of the energy emitted from the patch $\mathcal{M}_i$ and received by the patch $\mathcal{M}_j$, which is defined as

$$F_{i,j} = \frac{1}{A_i} \int_{\mathcal{M}_i} \int_{\mathcal{M}_j} G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') dA(\mathbf{x}), \tag{4.6}$$

where A_i is the area of the patch $\mathcal{M}_i$. We note that the original formulation by Goral et al. [27] assumes the scene contains no occluders, that is, the case that $V(\mathbf{x} \leftrightarrow \mathbf{x}')$ is always zero. The handling of visibility in the radiosity method is initially introduced independently by two papers [86, 18]. Specifically, the expansion of (*) in Eq. 4.4 can be calculated as

$$\begin{aligned}
(*) &= \int_{\mathcal{M}} (B_{\text{approx}}(\mathbf{x}) - B(\mathbf{x})) N_i(\mathbf{x}) dA(\mathbf{x}) \\
&= \underbrace{\int_{\mathcal{M}} B_{\text{approx}}(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x})}_{(1)} - \underbrace{\int_{\mathcal{M}} B(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x}),}_{(2)} \\
(1) &= \int_{\mathcal{M}} \underbrace{\left(\sum_{j=1}^{N} B_j N_j(\mathbf{x}) \right)}_{B_{\text{approx}}(\mathbf{x})} N_i(\mathbf{x}) dA(\mathbf{x}) = \sum_{j=1}^{N} B_j \underbrace{\int_{\mathcal{M}} N_j(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x})}_{\delta_{i,j} A_i} = B_i A_i, \\
(2) &= \int_{\mathcal{M}} \left(E(\mathbf{x}) + \frac{\rho(\mathbf{x})}{\pi} \int_{\mathcal{M}} B_{\text{approx}}(\mathbf{x}') G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') \right) N_i(\mathbf{x}) dA(\mathbf{x}) \\
&= \underbrace{\int_{\mathcal{M}} E(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x})}_{(3)} - \underbrace{\int_{\mathcal{M}} N_i(\mathbf{x}) \frac{\rho(\mathbf{x})}{\pi} \int_{\mathcal{M}} B_{\text{approx}}(\mathbf{x}') G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') dA(\mathbf{x}),}_{(4)} \\
(3) &= \int_{\mathcal{M}_i} E_i dA(\mathbf{x}) = E_i A_i, \\
(4) &= \int_{\mathcal{M}} N_i(\mathbf{x}) \frac{\rho(\mathbf{x})}{\pi} \int_{\mathcal{M}} \underbrace{\left(\sum_{j=1}^{N} B_j N_j(\mathbf{x}) \right)}_{B_{\text{approx}}(\mathbf{x})} G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') dA(\mathbf{x}) \\
&= \sum_{j=1}^{N} B_j \int_{\mathcal{M}} N_i(\mathbf{x}) \frac{\rho(\mathbf{x})}{\pi} \int_{\mathcal{M}} N_j(\mathbf{x}') G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') dA(\mathbf{x}) \\
&= \frac{\rho_i}{\pi} \sum_{j=1}^{N} B_j A_i \underbrace{\frac{1}{A_i} \int_{\mathcal{M}_i} \int_{\mathcal{M}_j} G(\mathbf{x} \leftrightarrow \mathbf{x}') dA(\mathbf{x}') dA(\mathbf{x})}_{F_{i,j}} = A_i \frac{\rho_i}{\pi} \sum_{j=1}^{N} B_j F_{i,j},
\end{aligned} \tag{4.7}$$

where $\delta_{i,j}$ is the Kronecker delta defined as $\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$. Eq. 4.5 can be obtained with dividing the both hand sides with A_i. In the middle of the calculation we implicitly used some identities such as $\int_{\mathcal{M}} f(\mathbf{x}) N_i(\mathbf{x}) dA(\mathbf{x}) = \int_{\mathcal{M}'} f(\mathbf{x}) dA(\mathbf{x})$.

As a result, the radiosity method comes down to solving a linear equation Eq. 4.5. We can use the general linear equation solvers. We often employ the iteration methods such as Jacobi iteration or Gauss-Seidel iteration. Some extensions of the radiosity method facilitates the internal flow of the iteration methods into the technique. The examples include the progressive radiosity [16], the stochastic radiosity [82, 83, 81], or the random walk radiosity [89, 96, 95]. The comprehensible introduction refers to the literature [8]. The other extensions of the radiosity method include handling of the discontinuities of the meshes due to the illumination change [70, 48], the adaptive mesh generation [12], the hierarchical structures [17, 40], the clustering [104], or the wavelet methods [28].

The major limitations on the FEM based approaches are their limited handling of the surface geometry and the materials. Because every FEM based approach needs the discretization of the surface geometry, the accuracy and the efficiency of the rendering depends on the complexity of the surface topology. If the subdivision of the surface is not enough, we can notice the discretization artifacts. Even if we can use the adaptive remeshing scheme, it is still hard to handle the complex lighting effects like caustics within the allowance of the trade-off between the quality and the computation cost. Although there are some trials for supporting general materials in the FEM approaches [50, 101, 100, 119], their support is still based on some assumptions and the approximations that cannot achieve the exotic features. Due to the limitation above, unfortunately, FEM based approaches cannot meet the real-world requirements for the increasing complexity and richness of the geometry and materials and this explains why the current research and the application mainly focus on the approaches based on Monte Carlo method.

4.2 Light Transport Simulation Based on Path Sampling

In this section, we will describe the light transport simulation based on Monte Carlo method, in particular those based on path sampling approaches. The general light transport simulation with Monte Carlo method has initially formulated and established by Kajiya [58] as the method named the *path tracing*. Followed by Kajiya, Veach [112] elaborated the formulation with the introduction of the path space formulation, as described in Chapter 2. The Kajiya's original path tracing is build upon the recursive formulation of the rendering equation (Eq. 2.28). Hense, the realization of the path tracing based on the recursive formulation incurs the recursive estimation of the integral. The estimation based on the recursive formulation is intuitively understood because we can directly estimate the original rendering equation, but their generality is still lacking when we want to consider the bidirectional transport. In this section, therefore, we will focus on the path sampling techniques in the path space formulation upon the explanation.

4.2.1 Estimation With Path Sampling

We will start our discussion with the simple N-sample estimate using the path sampling. Recalling the path is defined as a sequence of surface point, we denote the path with the length k with the notation $\bar{x} = \mathbf{x}_0 \dots \mathbf{x}_k$. According to Eq. 2.41 and Eq. 3.4, N–sample estimate $\hat{I}_k$ of the measurement with path length k can be written as

$$\hat{I}_k = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i)}{p(\bar{x}_i)}, \tag{4.8}$$

where $f(\bar{x}_i)$ is the measurement contribution function defined in Eq. 2.38 and $\bar{x}_i \sim p(\bar{x})$ is an i.i.d. set of samples. $p(\bar{x})$ is the pdf for path sampling, which is named as the *path sampler*. We also call the process characterized by the density as the *path sampling technique*.

The ultimate goal of the Monte Carlo method based approach is to design this path sampling technique as efficiently as possible. According to the knowledge of the importance sampling (Sec. 3.3.2), the goal is achieved when the density $p(\bar{x})$ can perfectly generate samples according to the contribution function $f(\bar{x}_i)$. However, as discussed in Sec. 3.3.2, it is virtually not possible because the contribution function contains the scene-dependent terms like the geometry factor $G(\mathbf{x} \leftrightarrow \mathbf{y})$.

4.2.2 Designing Path Sampler

Given the difficulties of making the perfect sampler, practically we want to design the path sampling technique as similar to the contribution function as possible. Here we will discuss on the design selection of the path samplers. In general, the path sampler $p(\bar{x})$ can be written as a joint pdf:

$$p(\bar{x}) = p(\mathbf{x}_0, \mathbf{x}_2, \dots, \mathbf{x}_k). \tag{4.9}$$

This joint pdf can be decomposed into the multiplication of the densities for the several mutually dependent sequence of vertices $\bar{x}_i$, which we call the *subpath*:

$$p(\bar{x}) = p(\underbrace{\mathbf{x}_{1,1}\mathbf{x}_{1,2} \dots \mathbf{x}_{1,k_1}}_{\bar{x}_1}) \times p(\underbrace{\mathbf{x}_{2,1}\mathbf{x}_{2,2} \dots \mathbf{x}_{2,k_2}}_{\bar{x}_2}) \times \cdots, \tag{4.10}$$

where the subpath $\bar{x}_i$ is a partition of the original path vertices $\{\mathbf{x} \in \bar{x}\}$, that is, $\bar{x}_i$ satisfies $\cup_i \bar{x}_i = \{\mathbf{x} \in \bar{x}\}$ and $\{\mathbf{x} \in \bar{x}_i\} \cap \{\mathbf{x} \in \bar{x}_j\} = \emptyset$ for all $i \neq j$. However this equation is still in the general form and not yet practical. In order to derive the practical decomposition, we assume the following two assumptions that the many of the existing light transport simulation techniques assume.

First we assume the path sampler can be decomposed into the *local samplers*, where each sampler is responsible for sampling a single vertex based on the information on the previous vertices. Using the local sampling, a subpath can be constructed by generating the vertices incrementally, making the random trajectories over the entire scene. This way of sampling a path is called the incremental path construction, or simply the random walk. This assumption comes from the physical process analog that the photon is propagated and scattered incrementally by the surface to the surface. The difference to the analog is that the actual sampling *direction* is arbitrary, meaning we can sample the vertice to the direction from the side of the sensor. Under this assumption, we can simplify the decomposition in Eq. 4.10 to

$$p(\bar{x}) = p(\underbrace{\mathbf{x}_0\mathbf{x}_1 \dots \mathbf{x}_{l_1-1}}_{\bar{x}_1}) \times p(\underbrace{\mathbf{x}_{l_1}\mathbf{x}_1 \dots \mathbf{x}_{l_2-1}}_{\bar{x}_2}) \times \cdots, \tag{4.11}$$

where $\mathbf{x}_i$ is a surface point of the input path $\bar{x}$. Here the vertices in the subpath $\bar{x}_i$ becomes a consecutive surface points, instead of the random partition of the input path vertices as in Eq. 4.10, because the dependent vertices are only constructed from the adjacent vertices by the assumption. Following the assumption, furthermore, the density $p(\bar{y})$ for each subpath $\bar{y} = \mathbf{y}_1 \dots \mathbf{y}_s$ can be also decomposed using the conditional pdfs as either

$$\begin{aligned} p(\bar{y}) &= p(\mathbf{y}_1) \times p(\mathbf{y}_2|\mathbf{y}_1) \times \cdots \times p(\mathbf{y}_{s-1}|\mathbf{y}_0, \dots, \mathbf{y}_{s-2}) \times p(\mathbf{y}_s|\mathbf{y}_0, \dots, \mathbf{y}_{s-1}), \text{ or} \\ p(\bar{y}) &= p(\mathbf{y}_s) \times p(\mathbf{y}_{s-1}|\mathbf{y}_s) \times \cdots \times p(\mathbf{y}_1|\mathbf{y}_s, \dots, \mathbf{y}_2) \times p(\mathbf{y}_0|\mathbf{y}_s, \dots, \mathbf{y}_1). \end{aligned} \tag{4.12}$$

The two pdfs differ in the direction of the sampling. The initial equation starts to sample from the side of the vertex $\mathbf{y}_1$, on the other hand, the second equation starts to sample the side of the vertex $\mathbf{y}_s$. We note that the conditional pdf often depends on the just one before vertex. In this case, we simply denote the equation as

$$\begin{aligned} p(\bar{y}) &= p(\mathbf{y}_1) \times p(\mathbf{y}_2|\mathbf{y}_1) \times \cdots \times p(\mathbf{y}_{s-1}|\mathbf{y}_{s-2}) \times p(\mathbf{y}_s|\mathbf{y}_{s-1}), \text{ or} \\ p(\bar{y}) &= p(\mathbf{y}_s) \times p(\mathbf{y}_{s-1}|\mathbf{y}_s) \times \cdots \times p(\mathbf{y}_1|\mathbf{y}_2) \times p(\mathbf{y}_0|\mathbf{y}_1). \end{aligned} \tag{4.13}$$

Even with this assumption we still have a freedom to design the path sampler. The second assumption limits the surface area sampling to the path edges, that is, the vertex corresponding to the sensor and the light source. In other words, the assumption means the sampling vertices must always start from the side of the sensor or the light source. The rationale behind the assumption is the observation that the sampling surface area in the middle of the paths is inefficient. We will explain the observation in an empirical example where every vertex in the subpaths in Eq. 4.11 is a singleton that contain the single vertex and $k \geq 2$:

$$p(\bar{x}) = p(\bar{x}_1) \times \cdots \times p(\bar{x}_{k+1}) = p(\mathbf{x}_0) \times \cdots \times p(\mathbf{x}_k). \tag{4.14}$$

In this case, the path sampling becomes a collection of vertex sampling of the surface area. For instance, if we want to sample the scene surfaces in an uniform probability, the densities becomes $p(\mathbf{x}_0) = \cdots = p(\mathbf{x}_k) = \frac{1}{A_{\text{total}}}$ where A_{total} is a total area of the scene surface. Although this sampling strategy is simple enough, unfortunately, this ways of sampling leads to the high variance. The inefficiency comes from the connection between the two adjacent vertices is likely to be have no contribution due to the visibility. To make matters worse, this issue becomes significant as the complexity of the scene surface increases. Furthermore, this sampling strategy cannot make use of the importance sampling according to the BSDF associated with the surface, because the distribution of the outgoing directions in BSDF needs the incoming direction, which is only obtained with the information of the previous vertex. For some materials containing delta functions like specular materials, this inability is critical and there is eventually no way to sample these paths because any directions except for a single outgoing direction which is deterministically calculated from an incoming direction can have a contribution.

The second assumption avoids this kind of situation by limiting the surface sampling to the edge vertices. Compared to the general surface sampling, the sampling of the surface on the sensor or the light sources is simpler because generally the total area (if they have) of the sensor or the light sources is much smaller than the total area of the entire scene surface. This means we can construct the specialized surface sampling targeted only to a part of scene surface where the sensor or the light sources are defined. With this assumption, the number of subpaths in Eq. 4.11 becomes one or two because otherwise at least one intermediate vertex is sampled from the surface. Furthermore, the decomposed conditional pdfs for each subpath in Eq. 4.13 can be simplified to one candidate, because each subpath contains exactly one edge vertex. Finally, we have the path pdf defined as

$$p(\bar{x}) = p(\underbrace{\mathbf{y}_0\mathbf{y}_1 \ldots \mathbf{y}_{s-1}}_{\bar{y}}) \times p(\underbrace{\mathbf{z}_0\mathbf{z}_1 \ldots \mathbf{z}_{t-1}}_{\bar{z}}), \tag{4.15}$$

in the case that the number of subpaths are two. Here we introduced the another notations for the subpaths from the each edges as a beginning defined as $\bar{y} = \mathbf{y}_0 \ldots \mathbf{y}_{s-1} = \mathbf{x}_0 \ldots \mathbf{x}_{s-1}$ and $\bar{z} = \mathbf{z}_0 \ldots \mathbf{z}_{t-1} = \mathbf{x}_k \ldots \mathbf{x}_s$, where s or t denote the number of vertices respectively in the subpaths $\bar{y}$ or $\bar{z}$. Likewise, the path pdf in the case of the single subpath can be written as either

$$p(\bar{x}) = p(\underbrace{\mathbf{y}_0\mathbf{y}_1 \ldots \mathbf{y}_k}_{\bar{y}}), \text{ or } p(\bar{x}) = p(\underbrace{\mathbf{z}_0\mathbf{z}_1 \ldots \mathbf{z}_k}_{\bar{z}}), \tag{4.16}$$

depending the direction of the sampling either from the side of the sensor or the light source. For convenience, we allow the *empty* subpath $\emptyset$ with the number of the vertices is zero and define the pdf for the empty subpath as $p(\emptyset) = 1$. Then we can merge the case in Eq. 4.16 into the case with Eq. 4.15. In this case, the number of vertices in each subpath is either $(s, t) = (k - 1, 0)$ or $(s, t) = (0, k - 1)$. Categorized by the type of the edge vertex, we shall call the subpath $\bar{y}$ as the *light subpath* and the subpath $\bar{z}$ as the *eye subpath.* The pdf for these subpaths $p(\bar{y})$ and $p(\bar{z})$ can be unambiguously defined as

$$\begin{aligned} p(\bar{y}) &= p(\mathbf{y}_0) \times p(\mathbf{y}_1|\mathbf{y}_0) \times \cdots \times p(\mathbf{y}_{s-2}|\mathbf{y}_{s-3}) \times p(\mathbf{y}_{s-1}|\mathbf{y}_{s-2}), \\ p(\bar{z}) &= p(\mathbf{z}_0) \times p(\mathbf{z}_1|\mathbf{z}_0) \times \cdots \times p(\mathbf{z}_{t-2}|\mathbf{z}_{t-3}) \times p(\mathbf{z}_{t-1}|\mathbf{z}_{t-2}). \end{aligned} \tag{4.17}$$

Now we have a set of path sampling technique based on the two assumptions. We call the path sampling strategy satisfying the two assumptions as the *vertex connection.* In summary, the first assumption determines the order of the vertices and limits the way of path sampling for the dependent vertices. The second assumption determines the number of dependent components of vertices and determines the way to sample the subpaths into the smaller set. Many of the light transport simulation techniques assume these two assumptions, although it might not be stated explicitly. We assume the following discussion relies on the path sampling strategy based on these assumptions, unless specified otherwise.

Finally we note that some of the light transport simulation techniques alleviated these assumptions and achieved the advanced path sampling technique. One notable alleviation is to make it possible to sample the multiple vertices from a single joint density, which is specifically called the *joint importance sampling*.

4.2.3 Sampling Subpaths

In the pdf for subpath sampling (Eq. 4.17), the decomposed vertex sampling pdfs can be categorized into two groups: the unconditional pdf $p(\mathbf{x})$ for the edge vertices, and and the conditional pdf $p(\mathbf{x}|\mathbf{x}')$ for the internal vertices. Typically the edge vertices are directly sampled the surface position representing the sensor or the light source. On the other hand, due to the assumptions, the internal vertices are indirectly sampled by tracing the ray $(\omega, \mathbf{x}')$ based on the randomly-sampled direction ω and the current vertex position $\mathbf{x}'$. The direction ω is actually sampled from the directional density $p_\sigma(\omega|\mathbf{x}', \omega_i)$ with respect to the solid angle measure $d\sigma$, where ω_i is the incoming direction. Because the target density $p(\mathbf{x}|\mathbf{x}')$ is defined on the area measure dA, the density $p_\sigma(\omega|\mathbf{x}', \omega_i)$ must be converted to the area measure. According to the chain rule, the conversion between the corresponding densities can be achieved by the multiplication of the Jacobians:

$$p(\mathbf{x}|\mathbf{x}') = \left|\frac{d\sigma}{dA}\right| p_\sigma(\omega|\mathbf{x}',\omega_i) = \left|\frac{d\sigma^\perp}{dA}\right|\left|\frac{d\sigma}{d\sigma^\perp}\right| p_\sigma(\omega|\mathbf{x}',\omega_i), \tag{4.18}$$

where $d\sigma^\perp$ is the project solid angle measure. According to the discussion in Chapter 2, each Jacobian can be obtained as $\left|d\sigma^\perp/dA\right| = G(\mathbf{x} \leftrightarrow \mathbf{x}')$ (Eq. 2.32) and $\left|d\sigma/d\sigma^\perp\right| = 1/|\mathbf{n}_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}}|$. Substituting these Jacobians, and assuming the vertices $\mathbf{x}$ and $\mathbf{x}'$ are already mutually visible, this equation further becomes

$$p(\mathbf{x}|\mathbf{x}') = \frac{G(\mathbf{x} \leftrightarrow \mathbf{x}')}{|\mathbf{n}_{\mathbf{x}'} \cdot \overrightarrow{\mathbf{x}'\mathbf{x}}|} p_\sigma(\omega|\mathbf{x}',\omega_i) = \underbrace{\frac{|\mathbf{n}_{\mathbf{x}} \cdot \overrightarrow{\mathbf{x}\mathbf{x}'}|}{\|\mathbf{x}-\mathbf{x}'\|^2}}_{G'(\mathbf{x}\leftrightarrow\mathbf{x}')} p_\sigma(\omega|\mathbf{x}',\omega_i), \tag{4.19}$$

where $G'(\mathbf{x} \leftrightarrow \mathbf{x}')$ is called the one-sided geometry term, which removes one cosine terms with respect to the outgoing direction from the geometry term. In practice, the direction ω is obtained by the changing domains from the uniform random numbers $\mathbf{u} \in [0,1]^n$, where n depends on the implementation of the sampler. Typically we use $n = 3$ where two of them are used for sampling direction, and the rest for sampling the type of the materials. In this case, because the pdf $p_{\mathcal{U}}$ for the uniform distribution is always evaluated to one, the path vertex pdf $p(\mathbf{x}|\mathbf{x}')$ can be written as

$$p(\mathbf{x}|\mathbf{x}') = \left|\frac{d\mathbf{u}}{dA}\right| \underbrace{p_{\mathcal{U}}(\mathbf{u}|\mathbf{x}',\omega_i)}_{1} = \left|\frac{d\mathbf{u}}{dA}\right|, \tag{4.20}$$

where $d\mathbf{u}$ is the Lesbegue measure associated to the pdf of the uniform distribution. This implies the vertex pdf can be utilized to obtain the Jacobian for changing the area measure dA and the Lesbegue measure $d\mathbf{u}$. Similarly, we can express the path pdf $p(\bar{x})$ with the Jacobian between the area product measure $d\mu$ and the Lesbegue measure $d\mathbf{u}$ for the corresponding uniform random numbers used to sample the path:

$$p(\bar{x}) = \left|\frac{d\mathbf{u}}{d\mu}\right|. \tag{4.21}$$

We note that, converted from Eq. 2.41, this equation implies another form of light transport equation changing the integration domain to a set of uniform random numbers $\mathcal{U} = [0,1]^{n(k-1)}$, where $k-1$ is the number of vertices in the path $\bar{x}$:

$$I = \int_{\mathcal{U}} f(\bar{x}(\mathbf{u})) \left|\frac{d\mu}{d\mathbf{u}}\right| d\mathbf{u} = \int_{\mathcal{U}} f(\bar{x}(\mathbf{u})) \left|\frac{d\mathbf{u}}{d\mu}\right|^{-1} d\mathbf{u} = \int_{\mathcal{U}} \frac{f(\bar{x}(\mathbf{u}))}{p(\bar{x}(\mathbf{u}))} d\mathbf{u}, \tag{4.22}$$

where we denote the path dependent on $\mathbf{u}$ as $\bar{x}(\mathbf{u})$ and used the identity on the inverse determinant $|d\mu/d\mathbf{u}| = |d\mathbf{u}/d\mu|^{-1}$. We will use this equation in the discussion later in Chapter 6.

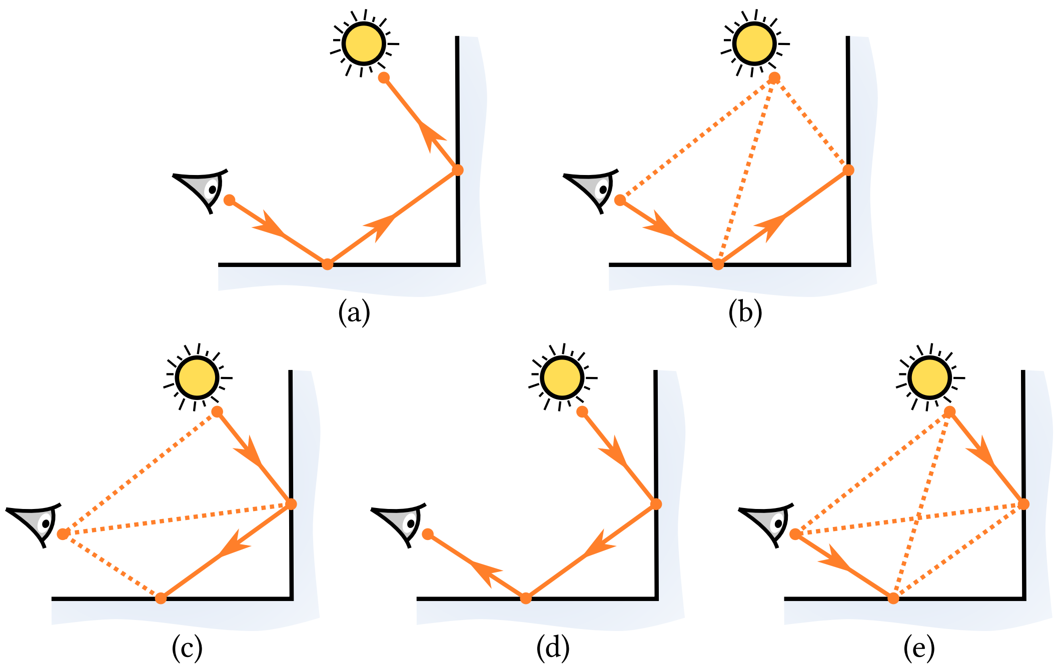


Figure 4.1: Classification of the path sampling techniques. (a) Path tracing traces the rays and construct a path from the side of the sensor until it finds a light source. (b) Path tracing with next event estimation explicitly connect the position on the light sources while traversing the rays. (c) Light tracing is a dual of path tracing with next event estimation, which traces rays from the side of the light source and connect explicitly to the position on the sensor every time the ray intersects the surfaces. (d) A hypothetical technique which is induced as a dual of path tracing. (e) Bidirectional path tracing is the generalization of the four techniques. The technique traces the rays from the both sides and construct a set of paths by connecting the vertices on both sides.

4.2.4 Classification of Path Sampling Techniques

We will discuss on the connection between the path sampling technique introduced above and existing Monte Carlo method based light transport simulation algorithms. We will classify the path sampling technique according to the number of vertices in the eye and light subpaths (s, t), where $0 \leq s, t < k$ and $s + t = k - 1$. The visual overview of the classification is summarized in Fig. 4.1.

Path tracing proposed by Kajiya [58] is the first and the most basic light transport simulation algorithm based on Monte Carlo method (Fig. 4.1a). Although the algorithm is originally formulated in the recursive formulation (Eq. 2.31), the process can be understood in the path space framework. Path tracing constructs the path from the side of the sensor, that is, the vertex $\mathbf{x}_k$ and incrementally construct the path by sampling the directions and tracing the rays depending on the current vertex. The termination of the path is determined stochastically by the technique known as russian roulette. We note that the ray intersection to the light source is not always an stopping condition except for the underlying albedo is zero, meaning the case that the contribution including the vertex turns out to be zero. Using the (s, t) notation, path tracing can be classified with path sampling techniques with $(s, t) = (k - 1, 0)$.

Path tracing with *next event estimation* is a variant of path tracing which is also proposed by Kajiya [58] (Fig. 4.1b). The algorithm is almost similar to path tracing; generates a sequence of vertices by sampling the directions and tracing the rays from the side on the sensor. The different between the ordinary path tracing is whenever the ray intersect a new surface, the algorithm creates an explicit connection to a light source, for instance by sampling the position on the light source, instead of creating an implicit connection by the intersection to the light source itself. In other word, the next event estimation always samples a light subpath with a single vertex and create a complete path by connecting the last vertex in the eye subpath. The next event estimation is classified with path sampling techniques with $(s, t) = (k - 2, 1)$.

Light trancing proposed by Arbo [5] is a dual technique to path tracing with next event estimation (Fig. 4.1c). Instead of tracing the path from the sensor, light tracing traces the path from the light source. That is, a complete path is constructed by initially sampling the position on the light source, and incrementally sample a sequence of samples by tracing the random ray directions. Similar yet different from path tracing with next event estimation, the algorithm creates an explicit connection whenever the ray intersect the surface to the randomly sampled position on the sensor. The classification of light tracing is $(s, t) = (1, k - 2)$.

Although not used frequently, we can consider a dual technique to path tracing (Fig. 4.1d). Similar to light tracing, the technique traces a rays from the side of the light source. Yet instead of finding the explicit connection to the sensor, a path is constructed by finding an implicit intersection to the sensor. This group of techniques is useless in most cases, because the actual camera models we use often assume the small or degenerated aperture. For instance for the pinhole camera model, the probability of finding an intersection to a point is represented by a delta function, hense the path with positive contribution cannot be possible with intersecting to the point itself, which of course results in zero probability — the final image turn out to be always black. We can classify this technique as $(s, t) = (0, k - 1)$.

Bidirectional path tracing proposed independently by Lafortune and Willems [66] and Veach and Guibas [113] facilitates the path constructed *both* from the sensor and the light sources. The idea of bidirectional path tracing is to construct a path by an explicit connection of the two vertices respectively in the eye and light subpaths (Fig. 4.1d). In this sense, we can consider bidirectional path tracing as a generalization of the four path sampling techniques that we introduced above. Also bidirectional path tracing makes use of combining several possible path sampling technique to generate a single image by making use of the multiple importance sampling. The

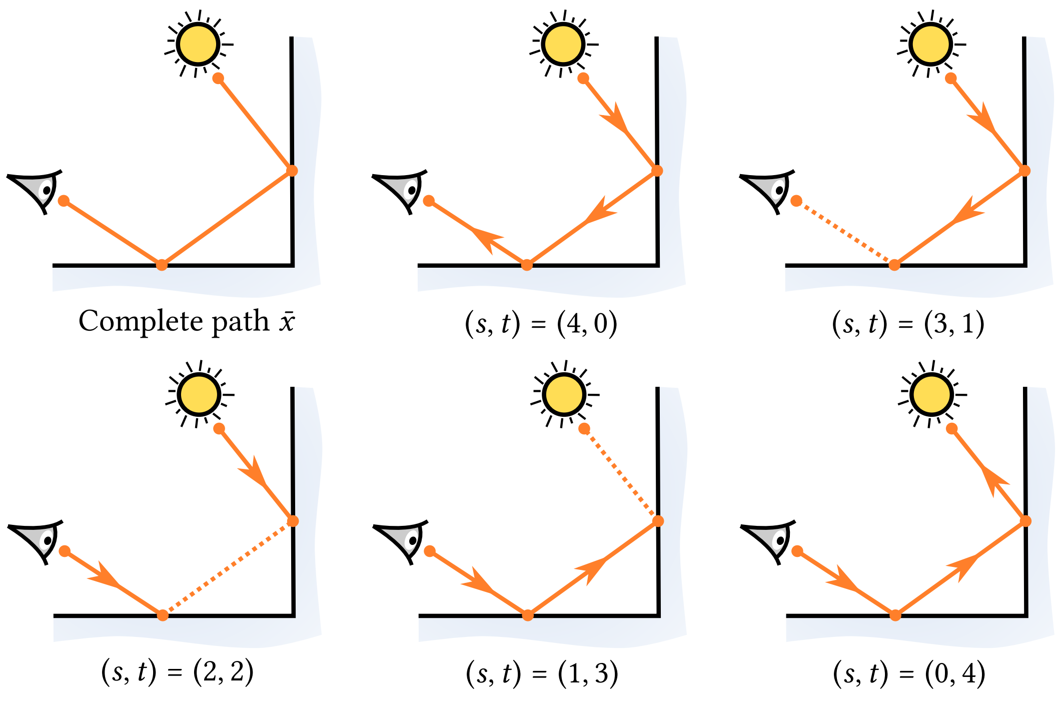


Figure 4.2: Multiple ways to construct a single path $\bar{x}$ for the path with length $k = 3$. In this case, the number of strategies that can generate the path $\bar{x}$ is five. The strategies are classified with all possible combinations of the number of vertices in the eye and light subpaths. The pair (s, t) under the figures illustrates the corresponding strategies.

path sampling by bidirectional path tracing can handle the most general case, so we can classify the technique as all possible combination of paths represented by (s, t).

4.2.5 Combining Multiple Path Sampling Strategies

We can notice that a single path can be generated from the multiple sampling strategies. For instance, in case of bidirectional path tracing, the path with length k has $k + 2$ different ways to construct a single path. Fig. 4.2 illustrates the case with $k = 3$. This means accumulating the contribution from these paths can result to double-count the contribution of the paths with specific length, which eventually becomes a bias of the estimate. In order to compensate the bias, we need to normalize the estimated contribution by multiplying the weights according to the combined path sampling techniques. Based on the idea, Veach [116] proposed *multiple importance sampling* as a general sampling technique to combine multiple sampling strategies for a single estimate (Sec. 3.3.2). In the case of bidirectional path tracing, the multi-sample estimate (Eq. 3.9) for all possible combinations of the techniques $(s, t) \in \mathcal{T}$ can be written as

$$\hat{I}_{\mathrm{MIS}} = \sum_{(s,t)\in\mathcal{T}} w_{s,t}(\bar{x}_{s,t}) \frac{f(\bar{x}_{s,t})}{p_{s,t}(\bar{x}_{s,t})}, \tag{4.23}$$

where we assigns one sample for each strategy explicitly denote the path pdf corresponding to the strategy (s,t) as $p_{s,t}(\bar{x})$ and the path sampled from the density as $\bar{x}_{s,t} \sim p_{s,t}(\bar{x})$. The selection of the weights is arbitrary as long as the two conditions (Sec. 3.3.3) are met. In his paper, Veach suggested to use power heuristics with $\beta = 2$ (Eq. 3.12). Combined with multiple importance sampling, bidirectional path tracing can alleviate the trade-offs in terms of the efficiencies between the different sampling strategies.

4.2.6 Path Reusal

Practically the concept of *path reusal* is important for the efficient path generation. Path reusal is a concept of generating a path based on other paths, which make it possible to generate multiple paths with smaller computation costs, compared to the case that the paths are generated independently from scratch. Path reusal is uniquitous in modern rendering techniques and many techniques explicitly or implicitly utilize the power of path reusal in their implementation. The examples include (bidirectional) path tracing, many light approaches, or photon density estimation based approaches.

Path Reusal in Path Tracing For instance, in case of path tracing without next event estimation, we consider the path $\bar{x}_k$ with path length k is generated sequencially sample the $k+1$ vertices from a sensor. Next when we want to sample a new path $\bar{x}_{k+1}$ with path length $k+1$, instead of resampling again from scratch, we can generate a single vertex $\mathbf{x}_{k+1}$ and construct a new path as $\bar{x}_{k+1} = \bar{x}_k \mathbf{x}_{k+1}$ by *reusing* the path with length k. Thus we can save the computation cost for sampling the $k+1$ vertices again. We note that the two paths $\bar{x}_k$ and $\bar{x}_{k+1}$ are correlated, but these samples contribute to the different estimates as the paths are samples for different path lengths, recalling the pixel intensity in Eq. 2.41 is defined for each path length. We still use a single sample for each estimate so we do not have to consider the error due to the correlation in this case.

Path Reusal in Bidirectioanl Path Tracing The use of path reusal becomes less obvious in the bidirectional case. As we explained the aforementioned sections, the paths in bidirectional path tracing are generated by combining the subpaths sampled from a sensor and a light source. Similar to the case of path tracing, the process of sampling subpaths can facilitate the reusal of paths with different lengths. For instance, we can generate a sequence of light subpaths $(\bar{y}_1, \dots, \bar{y}_{k_L})$ where

$$\bar{y}_1 = \mathbf{y}_0\mathbf{y}_1, \ \bar{y}_2 = \bar{y}_1\mathbf{y}_2, \ \dots, \ \bar{y}_{k_L} = \bar{y}_{k_L-1}\mathbf{y}_{k_L}, \tag{4.24}$$

by just sampling a single batch of the path vertices $(\mathbf{y}_0, \dots, \mathbf{y}_{k_L})$. Similarly, we can generate a sequence of eye subpaths $(\bar{z}_1, \dots, \bar{z}_{k_E})$ utilizing path reusal as

$$\bar{z}_1 = \mathbf{z}_0\mathbf{z}_1,\ \bar{z}_2 = \bar{z}_1\mathbf{z}_2,\ \dots,\ \bar{z}_{k_E} = \bar{z}_{k_E-1}\mathbf{z}_{k_E}. \tag{4.25}$$

Using the set of subpaths, a set of (full) paths with lengths k that can be generated with all strategies can be written as

$$(\underbrace{\bar{y}_k \oplus \emptyset}_{\bar{x}_{k,0}}, \underbrace{\bar{y}_{k-1} \oplus \bar{z}_1}_{\bar{x}_{k-1,1}}, \cdots, \underbrace{\bar{y}_s \oplus \bar{z}_t}_{\bar{x}_{s,t}}, \cdots, \underbrace{\bar{y}_1 \oplus \bar{z}_{k-1}}_{\bar{x}_{1,k-1}}, \underbrace{\emptyset \oplus \bar{z}_k}_{\bar{x}_{0,k}}), \tag{4.26}$$

where $s+t = k$ and $\oplus$ is the concatenation operator. We note that the full combination of paths cannot be generated from the subpaths. For instance, the path with length $k = k_L + k_E$ should generate $k_L + k_E + 1$ number of paths. but we actually get a single path $\bar{y}_{k_L} \oplus \bar{z}_{k_E}$.

Variance of Multi Sample Estimate With Correlated Samples Unlike path tracing, the paths generated by Eq. 4.26 are correlated. This affects the convergence behavior of the variance because the samples in multi-sample estimate (Eq. 3.9) are assumed to be uncorrelated. Fortunately, the effect of correlation of the samples between the different strategies are limiting, because we can show that the correlation between the samples generated from the different strategy is diminishing toward zero as the number of samples is getting larger. In other words, the correlation of multi-sample estimate is mainly influenced by the correlation of the samples generated for the same strategy. In case of bidirectional path tracing, the samples in the same strategy are i.i.d. so we can safely ignore the effect from the correlation.

We will show the correlation between the samples generated from the different strategy becomes zero as the number of samples $N \longrightarrow \infty$, in case of using multi-sample estimate with aforementioned path reusal. We first start from the multi-sample estimate in Eq. 3.9:

$$\hat{I}_{\mathrm{MIS}} = \sum_{i=1}^{M} \frac{1}{N_i} \sum_{j=1}^{N_i} \underbrace{w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})}}_{F_{i,j}} = \sum_{i=1}^{M} \frac{1}{N_i} \sum_{j=1}^{N_i} F_{i,j}, \tag{4.27}$$

where M is the number of strategies and $N_i \leq N$ is the number of samples for the i-th strategy. We also denote the inner term as $F_{i,j}$. In general, the variance of the sum of correlated random variables $(X_1, \dots, X_N)$ can be written as

$$\mathrm{Var}\left(\sum_{i=1}^{N} X_i\right) = \sum_{i=1}^{N}\sum_{j=1}^{N} \mathrm{Cov}(X_i, X_j), \tag{4.28}$$

where $\mathrm{Cov}(\cdot,\cdot)$ is the covariance between two random variables defined as $\mathrm{Cov}(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. We note that $\mathrm{Cov}(X, X) = \mathrm{Var}(X)$ by definition. Using this identity, the variance of the multi-sample estimate can be written as

$$\mathrm{Var}\left(\hat{I}_{\mathrm{MIS}}\right) = \sum_{i_1=1}^{M}\sum_{i_2=1}^{M} \frac{1}{N_{i_1}N_{i_2}} \sum_{j_1=1}^{N_{i_1}}\sum_{j_2=1}^{N_{i_2}} \mathrm{Cov}(F_{i_1,j_1}, F_{i_2,j_2}). \tag{4.29}$$

The first two sums are separated into two components: (1) the component that the strategies are same ($i_1 = i_2$) and (2) the component that the strategies are different ($i_1 \neq i_2$).

$$= \underbrace{\sum_{i=1}^{M} \frac{1}{N_i^2} \sum_{j_1=1}^{N_i}\sum_{j_2=1}^{N_i} \mathrm{Cov}(F_{i,j_1}, F_{i,j_2})}_{(1)} + \underbrace{\sum_{i_1 \neq i_2} \frac{1}{N_{i_1}N_{i_2}} \sum_{j_1=1}^{N_{i_1}}\sum_{j_2=1}^{N_{i_2}} \mathrm{Cov}(F_{i_1,j_1}, F_{i_2,j_2})}_{(2)}. \tag{4.30}$$

In case of bidirectional path tracing, the samples in the same strategy are i.i.d., because each strategy generates only one sample for each epoch, that is, for each j. This implies the correlation between the samples in the same strategy is zero: $\mathrm{Cov}(F_{i,j_1}, F_{i,j_2}) = 0$ when $j_1 \neq j_2$. Therefore we have

$$(1) = \sum_{i=1}^{M} \frac{1}{N_i^2} \sum_{j=1}^{N_i} \mathrm{Cov}(F_{i,j}, F_{i,j}) = \sum_{i=1}^{M} \frac{1}{N_i^2} \sum_{j=1}^{N_i} \mathrm{Var}(F_{i,j}), \tag{4.31}$$

where we used the identity $\mathrm{Cov}(F_{i,j}, F_{i,j}) = \mathrm{Var}(F_{i,j})$. We can further rewrite the equation using the fact that the samples $x_{i,j}$ distributed to the same distribution of i-th strategy, by introducing the single-sample estimate of $F_i \equiv F_{i,0}$:

$$= \sum_{i=1}^{M} \frac{1}{N_i^2} \sum_{j=1}^{N_i} \mathrm{Var}(F_i) = \sum_{i=1}^{M} \frac{1}{N_i^2} \cdot N_i \cdot \mathrm{Var}(F_i) = \sum_{i=1}^{M} \frac{1}{N_i} \cdot \mathrm{Var}(F_i). \tag{4.32}$$

Similarly, in case of the case in the component (2), we can assume the correlation between the different strategies are zero because the path reusal is utilized to generate a set of paths for the strategy in the same epoch and the samples with different epochs are mutually independent. Therefore we have

$$\begin{aligned} (2) &= \sum_{i_1 \neq i_2} \frac{1}{N_{i_1}N_{i_2}} \left(\sum_{j} \mathrm{Cov}(F_{i_1,j}, F_{i_2,j}) + \sum_{j_1 \neq j_2} \underbrace{\mathrm{Cov}(F_{i_1,j_1}, F_{i_2,j_2})}_{=0} \right) \\ &= \sum_{i_1 \neq i_2} \frac{1}{N_{i_1}N_{i_2}} \sum_{j} \mathrm{Cov}(F_{i_1,j}, F_{i_2,j}). \end{aligned} \tag{4.33}$$

Here, the Cauchy-Schwarz inequality for the two random variable X and Y states that $\mathrm{Cov}(X, Y)^2 \leq \mathrm{Var}(X)\mathrm{Var}(Y)$. Using this inequality, the component (2) can be bounded as

$$\begin{aligned}
|(2)| &\leq \sum_{i_1 \neq i_2} \frac{1}{N_{i_1} N_{i_2}} \sum_j \left|\mathrm{Cov}(F_{i_1,j}, F_{i_2,j})\right| \\
&\leq \sum_{i_1 \neq i_2} \frac{1}{N_{i_1} N_{i_2}} \sum_j \left(\mathrm{Var}(F_{i_1,j})\mathrm{Var}(F_{i_2,j})\right)^{1/2} \\
&\leq \sum_{i_1 \neq i_2} \frac{1}{N_{i_1} N_{i_2}} \sum_j V = \sum_{i_1 \neq i_2} \frac{1}{N_{i_1} N_{i_2}} \cdot \min(N_{i_1}, N_{i_2}) \cdot V \\
&= \sum_{i_1 \neq i_2} \min\left(\frac{1}{N_{i_1}}, \frac{1}{N_{i_2}}\right) \cdot V_{i_1,i_2},
\end{aligned} \tag{4.34}$$

where $V_{i_1,i_2} = \max_{j, i \in \{i_1, i_2\}} \mathrm{Var}(F_{i,j})$ and we used the triangle inequality for the first bound. Wrapping up the aforementioned discussion, we finally have an inequality

$$\mathrm{Var}\left(\hat{I}_{\mathrm{MIS}}\right) \leq \sum_{i=1}^{M} \frac{1}{N_i} \cdot \mathrm{Var}(F_i) + \sum_{i_1 \neq i_2} \min\left(\frac{1}{N_{i_1}}, \frac{1}{N_{i_2}}\right) \cdot V_{i_1,i_2}. \tag{4.35}$$

From this inequality, we have $\lim_{N \to \infty} \mathrm{Var}\left(\hat{I}_{\mathrm{MIS}}\right) = 0$ because $N \longrightarrow \infty$ implies $N_i \longrightarrow \infty$. This inequality also implies the convergence ratio of the variance with path reusal is not so worse than the case without correlations.

Chapter 5

Bridging Strategy Spaces of Light Transport Simulations

5.1 Introduction

Rendering based on light transport simulation is a popular approach for photorealistic image synthesis. Since such rendering algorithms solve the same governing equations (e.g., the rendering equation [58]), rendering with light transport simulation should give us the same result regardless of the choice of an algorithm. It is however well known that some algorithms are more efficient at rendering certain light transport effects. For example, photon density estimation [54, 34] is often efficient at rendering caustics, and Markov chain Monte Carlo algorithms [117, 52] are considered efficient at resolving complex occlusions.

Because of the varying efficiency of different algorithms on different light transport effects, it is common practice to select an algorithm based on the type of light transport effect that one wants to render. In the movie industry, an artist often decomposes light transport effects into separate images, renders each with a most efficient algorithm, and composites the resulting images into the final one. Selecting appropriate algorithms and composting the results, however, can be difficult and cumbersome tasks. For selection, an artist either needs to know why some algorithms work well for some effects, or briefly tries all the available algorithms to see which one works well. For composition, an artist also needs to pay attention not to double count a certain type of paths such as caustics.

We propose a framework which automates this selection of the algorithms and composition of the resulting images. Our work is inspired by the superhuman performance of recent machine learning algorithms on classification tasks. We apply the same idea to select and composite two different rendering algorithms based on the classification of light transport effects. To be concrete, we use regression forests [10] to learn the relationship between blending weights that minimize the error and the classification of light transport effects. While multiple importance sampling [116]

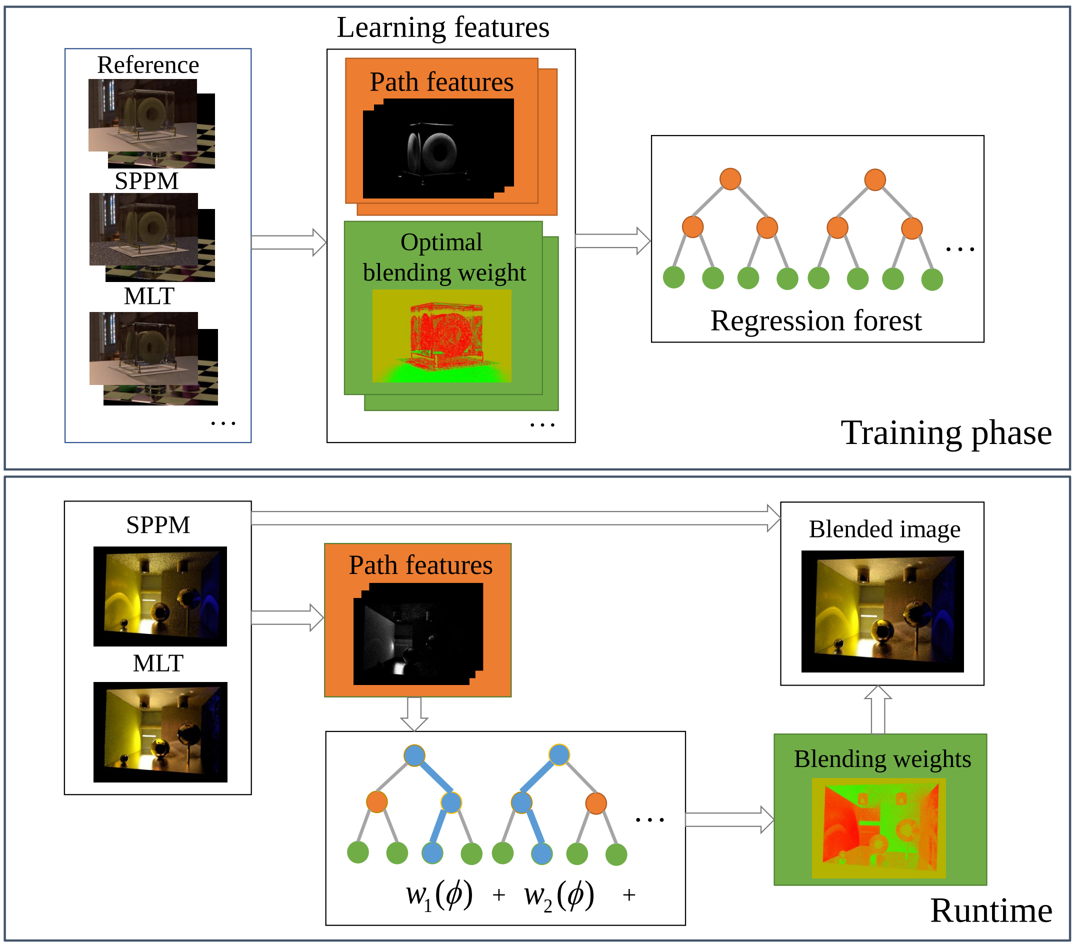


Figure 5.1: General idea of blending the results of two different rendering algorithms using regression forests. In the training phase (top), we first calculate the optimal blending weight per pixel, given the reference image and rendered images with different approaches. These weights and the corresponding path features become one training sample for the regression forest for each scene. We iterate this process for various scenes. Our framework thus learns the relationship between input path features and optimal weights during this learning phase. At runtime (bottom), the trained regression forest returns approximated optimal blending weights based on path features of a new scene.

also allows us to blend results of different rendering techniques, the key difference is that our framework treats each rendering algorithms as a black-box. Accordingly, our framework can be easily applied to very different algorithms such as SPPM and MLT without any algorithmic or theoretical modifications for each. To summarize, our contributions are:

- The use of machine learning to automatically blend the results of different rendering algorithms based on path types.
- A blending framework which is independent from how the underlying rendering algorithms work.

- First successful application of regression forests to light transport simulation.

5.2 Overview

Our goal is to blend the results of two different rendering algorithms such that the error of the blended result is as small as possible. Our algorithm is separated into two phases; the *training* phase and *runtime*. Fig. 7.1 illustrates the algorithm.

In the training phase, we use regression forests [10] (Section 5.4) to learn the relationship between a feature vector of lighting effects extracted from the rendered images and the optimal weights for blending. For each training scene, we render the reference solution, and the two images with both algorithms allocating the same rendering time. Based on the rendered images, we extract *path features* as the relative pixel contributions of different light transport paths according to Heckbert's notation [47]. Modern shader languages often support the same mechanism [30]. We then calculate the *optimal blending weights* based on the reference solution and the results of the two different rendering algorithms. The optimal blending weight is defined such that the error of the blended result is minimized at each pixel. A pair of path features and the optimal blending weight forms one training sample for regression forests. If a rendering algorithm is based on Monte Carlo methods (which is the case in our experiments), we generate multiple training samples for the same scene in order to avoid the influence of the randomness of rendered images.

At runtime, we use the trained regression forest to approximate the optimal blending weights for a given new scene. The path features extracted from the rendered images are used to traverse the regression forest to obtain the blending weights. The final result is a blended image with the obtained weights. Since the trained regression forest expresses the relationship between path features and the optimal weights, a blended image is expected to have small error, even for a scene that was not included in the training phase.

5.3 Automatic Blending with Path Features

5.3.1 Path Features

Our definition of a feature vector for rendering algorithms is inspired by how artists decompose a rendered image into several images with specific lighting effects for each. In order to define the feature vectors, we begin with the formulation of the light transport known as the path integral formulation [112]. According to the formulation, the pixel intensity I observed at each pixel is expressed as

$$I = \int_{\Omega} f(\bar{x}) d\mu(\bar{x}), \tag{5.1}$$

where $\bar{x}$ is a light transport path, f is the measurement contribution function, and μ is the path measure. Ω is the space of paths of all different path lengths.

The path space Ω can be partitioned into a union of disjoint spaces according to the classification by Heckbert [47]:

$$\Omega = \Omega_{\mathrm{LDE}} \cup \Omega_{\mathrm{LSE}} \cup \Omega_{\mathrm{LDSE}} \cup \Omega_{\mathrm{LSDE}} \cup \cdots, \tag{5.2}$$

where each Ω_* is a subspace of Ω defined with the paths represented by the Heckbert's notation $*$. For instance, the subspace Ω_{LDSE} with path length 3 is defined as a set of paths $\bar{x} = \mathbf{x}_0\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$ where $\mathbf{x}_0$ is on a sensor, $\mathbf{x}_1$ is on a diffuse surface, $\mathbf{x}_2$ is on a specular surface, and $\mathbf{x}_3$ is on an emitter. A glossy interaction is classified to either D or S depending on its BRDF.

We thus define a part of the intensity I_* contributed only with the subspace Ω_* as

$$I_* = \int_{\Omega_*} f(\bar{x})d\mu(\bar{x}). \tag{5.3}$$

Since the partition in Equation 5.2 is disjoint, the pixel intensity I is additive:

$$I = I_{\mathrm{LDE}} + I_{\mathrm{LSE}} + I_{\mathrm{LDSE}} + I_{\mathrm{LSDE}} + \cdots. \tag{5.4}$$

We thus define the *path features* ϕ as a vector of the intensities I_* relative to I:

$$\phi \equiv \frac{(I_{\mathrm{LDE}}, I_{\mathrm{LSE}}, I_{\mathrm{LDSE}}, I_{\mathrm{LSDE}}, \dots)}{I}. \tag{5.5}$$

The definition uses relative intensities such that ϕ is independent from the absolute intensity. We fixed the maximum path length to ten, which makes ϕ a $2^{(10-1)} = 512$ dimensional feature vector. The training phase uses an *estimate* $\hat{\phi}$ instead of the analytical value of ϕ for a given rendering time. We selected the number of dimensions such that all the data fits within the main memory. For instance, the scene rendered with 720p resolution requires a storage of $512 \times 1280 \times 720 \times 4$ bytes ≈ 1.8 gigabytes.

5.3.2 Optimal Blending Weights

In the training phase, we need to determine the optimal blending weight. This weight is used as an *answer* associated with a path feature vector. A pair of a path feature vector and the optimal blending weight thus becomes a training sample for supervised learning via regression forests.

We define the optimal blending weight w_{opt} that gives the minimum error as

$$w_{\mathrm{opt}} = \operatorname*{argmin}_{w} \left| \left(w\hat{I}_\alpha + (1-w)\hat{I}_\beta \right) - I \right|, \tag{5.6}$$

where $\hat{I}_\alpha$ and $\hat{I}_\beta$ are the results of two different rendering algorithms α and β respectively, and I is the reference solution. This equation can be easily solved as

$$w_{\text{opt}} = \frac{I - \hat{I}_\beta}{\hat{I}_\alpha - \hat{I}_\beta}. \tag{5.7}$$

If the solution of Equation 5.6 is not in the range of $[0, 1]$, it is clamped to the nearest side such that $w_{\text{opt}} \in [0, 1]$. We apply this clamping such that the blending operation becomes a convex combination of the results. The blended result $w\hat{I}_\alpha + (1 - w)\hat{I}_\beta$ is thus guaranteed to be more accurate than one of $\hat{I}_\alpha$ and $\hat{I}_\beta$ since

$$|w\hat{I}_\alpha + (1 - w)\hat{I}_\beta - I| \leq \max(|\hat{I}_\alpha - I|, |\hat{I}_\beta - I|) \tag{5.8}$$

by definition if $w \in [0, 1]$. Intuitively, this clamping process sets $w_{\text{opt}} = 1$ when $\hat{I}_\alpha$ and $\hat{I}_\beta$ both either underestimate or overestimate I and $\hat{I}_\alpha$ is closer to I (vice versa for $\hat{I}_\beta$). If one of the $\hat{I}_\alpha$ and $\hat{I}_\beta$ underestimates and the other overestimates I, we set w_{opt} such that the blended result is exactly equal to I. Note that Equation 5.8 only guarantees that an error per pixel does not become worse, not the sum of errors over an image. For example, collecting pixels with worse errors (with $w = 0$ or $w = 1$) still satisfies Equation 5.8, but the sum of errors would increase.

The intensities $\hat{I}_\alpha$ and $\hat{I}_\beta$ are the relatively rough estimates of I in practice. If an algorithm is based on Monte Carlo ray tracing, an estimated intensity is an instance of the random variable for each run. Using samples only from a single run of the algorithm causes overfitting to this specific run. For example, it might be that $\hat{I}_\alpha$ happens to be closer to I than $\hat{I}_\beta$ for the single run used in the training phase.

In order to deal with this issue, we use multiple training samples even for the same scene and the same algorithm. In fact, machine learning techniques (including regression forests) are naturally designed for dealing with such variations in the training data.

Problem Statement Given the definitions above, the goal of our algorithm is to find a function w_{approx} such that

$$w_{\text{opt}} \approx w_{\text{approx}}(\phi), \tag{5.9}$$

for given path features ϕ and two rendering algorithms α and β. This function w_{approx} basically expresses the preference of the algorithm α over the other algorithm β for paths with a feature vector of ϕ. In order to learn w_{approx}, we use a machine learning algorithm called *regression forests*.

Difference Between Multiple Importance Sampling Conceptually, our proposed blending approach is similar to multiple importance sampling (MIS) [116]. MIS also combines two or more different estimators to improve the efficiency of the combined estimator.

MIS combines multiple sampling strategies by decomposing the measurement contribution function f in Eq. 6.1 into a weighted sum of M different weights. The estimate of the pixel intensity I by MIS can be written as

$$I = \int_{\Omega} \sum_{t=1}^{M} w_t(\bar{x}) f(\bar{x}) d\mu(\bar{x}) = \sum_{t=1}^{M} \int_{\Omega} w_t(\bar{x}) f(\bar{x}) d\mu(\bar{x}) \tag{5.10}$$

$$\approx \sum_{t=1}^{M} \frac{1}{N_t} \sum_{i=1}^{N_t} w_t(\bar{x}_{t,i}) \frac{f(\bar{x}_{t,i})}{p_t(\bar{x}_{t,i})} \tag{5.11}$$

where $p_i(\bar{x})$ is the pdf of the i-th strategy, $w_i(\bar{x})$ is the weighting function satisfying $\sum_{t=1}^{M} w_i(\bar{x}) = 1$ for all $\bar{x} \in \Omega$ with $f(\bar{x}) \neq 0$, and $w_i(\bar{x}) = 0$ for all $\bar{x} \in \Omega$ with $p_i(\bar{x}) = 0$. In order to use MIS, however, we need to know the probability densities of path sampling techniques for arbitrary sample locations. Such information can be difficult to obtain without modifying an implementation or sometimes impossible due to the formulation of each algorithm.

5.4 Regression Forests

The basic idea of regression forests is to use a set of binary trees for approximating a multivariate function of the feature vector. This multivariate function expresses the relationship between feature vectors and the corresponding value. Each binary tree is called a *regression tree* where the inner nodes (split nodes) express branching conditions on an input feature vector. Each regression tree takes an input feature vector and outputs a value associated with the corresponding leaf node. Regression forests return the average of the outputs of regression trees as the final output.

5.4.1 Construction

For the construction of regression forests, we need a large number of training samples which associate feature vectors (a set of path features) and output values (optimal weights). We generate these samples by rendering several training scenes. We then extract the path features and the corresponding optimal weights for each scene. The regression forest is trained to approximate the optimal weights even for a new scene, based only on the path features.

We define a training sample $t \equiv (\phi^t, w^t_{\text{opt}}) \in \mathcal{T}$ as a tuple of path features ϕ^t and the optimal weight w^t_{opt}. $\mathcal{T}$ is a set of all training samples. The construction process begins from the root node of the regression forest. Each step of the construction process recursively splits training samples into left and right nodes. We denote the subset of the training samples in the currently processed node as $T \subseteq \mathcal{T}$ and we start from $T = \mathcal{T}$. The algorithm is similar to a top-down construction of a kd-tree for ray tracing [20, 43].

Node Splitting The construction process continues splitting the current node until the number of training samples in the current set T is smaller than a threshold, or the depth of the tree has reached the maximum depth. If the recursion terminates, the current node becomes a leaf node. Each leaf node stores the average over the set of the optimal weights in this node as w_{leaf}. This average weight approximates the optimal weight at runtime.

If the recursion continues, we split the current set of samples T into two disjoint subsets T_L and T_R according to a threshold θ and an index k of the path features:

$$T_L(\theta, k) = \{t \in T | \phi^t(k) \geq \theta\} \tag{5.12}$$

$$T_R(\theta, k) = T \setminus T_{\text{L}}(\theta, k) \tag{5.13}$$

where $\phi^t(k)$ is the k-th element of the path features ϕ^t. The threshold θ and the index k at each step are defined as $(\theta, k) = \text{argmax}_{\theta', k'} V(\theta', k', T)$, where $V(\theta', k', T) = \text{Var}(T) - \text{Var}(T_L(\theta', k')) - \text{Var}(T_R(\theta', k'))$. Here $\text{Var}(T)$ is the variance of the optimal weights in T. The function V is used to define the most discriminative pair of the threshold θ and the index of the path feature k according to the variance

5.4.2 Runtime

In our framework, we first render a given new scene with two different algorithms $\hat{I}_\alpha$ and $\hat{I}_\beta$ with the same computation time. We also extract the path features ϕ according to the definition by Equation 5.5. Using these path features, we can now evaluate each trained regression tree by traversing down the tree according to the branching condition defined in Equation 5.12, which eventually reaches a leaf node and the weight w_{leaf} is recorded in the leaf node. By repeating this process for all regression trees in the trained regression forest, we obtain a set of weights w_{leaf} recorded in the leaf nodes for each tree. We define $w_r(\phi)$ as the output of the r-th tree in the trained regression forest, given the path features ϕ. The approximated optimal weight $w_{\text{approx}}(\phi)$ with M trees is given as

$$w_{\text{approx}}(\phi) = \frac{1}{M} \sum_{r=1}^{M} w_r(\phi). \tag{5.14}$$

Blending at each pixel is $w_{\text{approx}}(\phi)\hat{I}_\alpha + (1 - w_{\text{approx}}(\phi))\hat{I}_\beta$. This evaluation process is repeated for all the pixels. The use of forests can alleviate the discontinuity of the resulting weights. Even if one tree suddenly returns a totally different value due to hard classification, it is likely that other trees still return similar weights. As a result, returning weights will be smoothly changing.

5.4.3 Refinement

A trained regression forest is sometimes too optimized for given training samples. In order to reduce overfitting, we follow the refinement technique for regression forests proposed by Ren et al. [93] and Ladický et al. [65]. The main idea is to split a set of training samples into two subsets and use one for constructing the structure of each tree while using the other for defining the outputs. After the construction step, we first discard the values w_{leaf} assigned to the leaf nodes while keeping the tree structure. The refinement process then updates w_{leaf} using the additional training samples.

For each additional training sample ϕ, we execute the evaluation of the tree until the evaluation process reach to the leaf node. After collecting the set of training samples Φ reached to the leaf node, the updated weight w_r^* can be computed as

$$w_r^* = \frac{1}{|\Phi|} \sum_{(\phi, w_{\text{opt}}) \in \Phi} w_{\text{opt}}(\phi). \tag{5.15}$$

We iterate this refinement process for each tree in the regression forest using the sample training set for the refinement. Since the training samples are taken from the different portion of the training set independent of the samples assigned for the initial construction, the final weights associated to the leaf node could become more generic, which alleviates overfitting to the initial training set.

5.5 Results

We selected the two combinations of the rendering algorithms to show the effectiveness of our framework: (1) stochastic progressive photon mapping (SPPM) [32] and Metropolis light transport (MLT) [117] with manifold exploration [52] shown in Fig. 7.7, (2) SPPM and bidirectional path tracing (BDPT) [66, 115] shown in Fig. 5.3. We chose these algorithms because both the algorithm and the performance are distinguishably different. One famous characteristic of SPPM is the ability to handle specular-diffuse-specular paths efficiently. A caustic that can be seen through a water surface is an example of such paths. MLT is based on Markov chain Monte Carlo sampling which utilizes a sequence of correlated samples that forms a Markov chain. The sequence of the samples is generated such that the resulting sample distribution follows an arbitrary user-defined target function such as the measurement contribution function. MLT is known to be effective for the scenes with complex occlu-

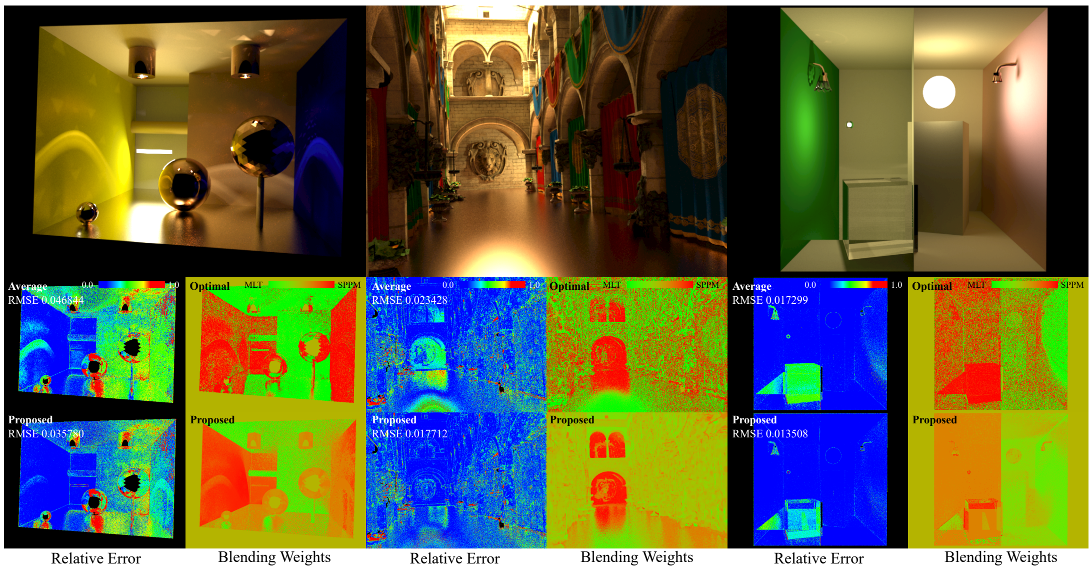


Figure 5.2: Equal-time comparison (20 minutes) of the average and our automatic blending of the images rendered by SPPM [32] and MLT [117] with manifold exploration [52]. We highlighted three scenes with different characteristics from our test cases (box, cryteck-sponza, and water). The top row shows the reference images. The bottom two rows visualize errors, the optimal blending weights, and the output blending weights of our framework. Depending on the types of lighting effects, the optimal blending weights for SPPM and MLT that result in the minimal error vary significantly. Simply taking the average of SPPM and MLT thus produces a suboptimal result in terms of RMS error.

sion. BDPT can utilize various sampling technique by the combination of paths traced from the sensor and the lights. These sampling techniques are combined with multiple importance sampling [116]. The combination of SPPM and BDPT would exhibit the good trade-off because BDPT is not efficient at handling specular-diffuse-specular paths [61] and while being more efficient at rendering diffuse surfaces [36, 26].

For the implementations of rendering algorithms, we used the Mitsuba renderer [51]. Mutation techniques used for MLT are bidirectional, lens, caustic, multi-chain, and manifold perturbation [52]. All the images except for the reference images are rendered on a machine with Intel Core i7-4720HQ at 2.6 GHz. The training phase is computed with a machine with Intel Core i7-3970X at 3.5 GHz and 16 GB of main memory. We utilized only a single core for rendering in order to alleviate the difference of performance between SPPM and MLT according to the parallelization. In order to facilitate the future work, we publish our implementation on our website.

Training Samples Our training set consists of 10 scenes with various characteristics in order to cover as many types of paths as possible. We render all the scenes with each rendering algorithm for 5, 10, 15, and 20 minutes. Each scene is rendered

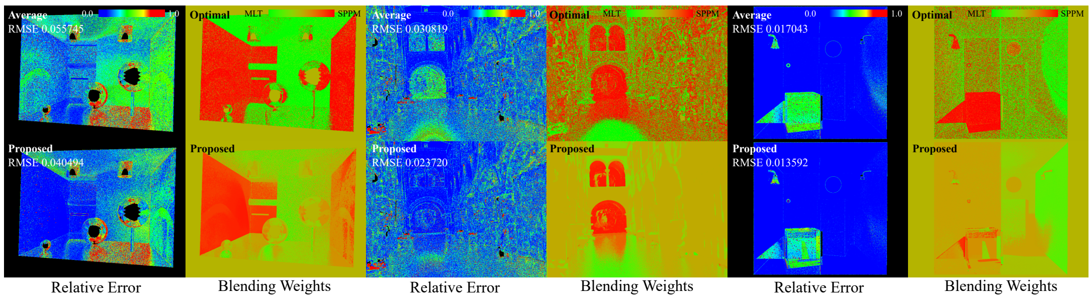
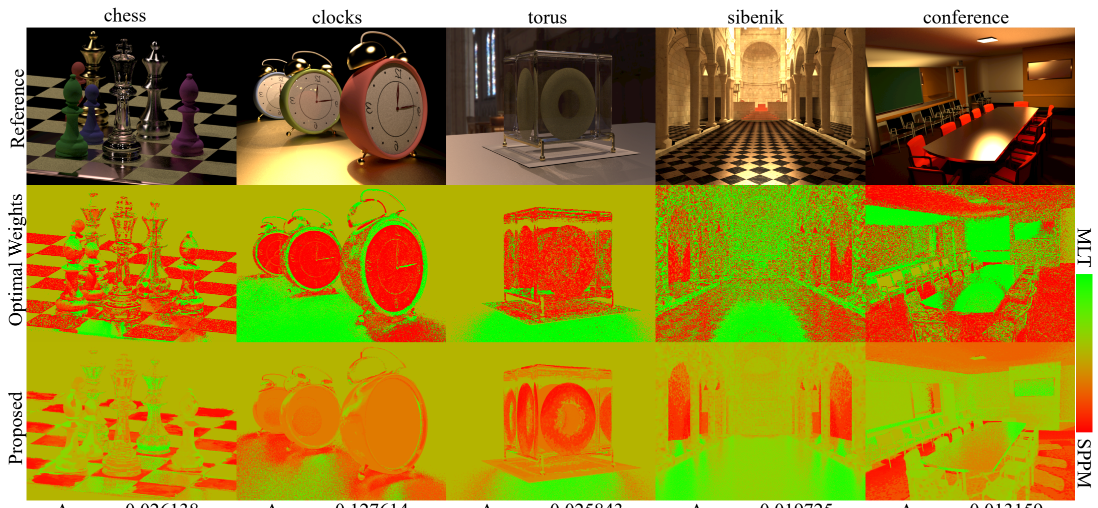


Figure 5.3: Comparison of errors, the optimal weights and the approximations by our framework for the combination of BDPT and SPPM. The selection of the scenes and meaning of the images are same as Fig. 7.7. Similar to the combination of MLT and SPPM, our framework generally captures the preference to the scene according to the characteristics of the scenes, although some difference can be observed, e.g., preference to the scene dominated with specular material is weaker (*box* scene).

five times, in order to alleviate overfitting as discussed in Section 5.3.2. Given this whole training data, we generate a regression forest for each scene by excluding the scene from the training data. We thus have 10 different regression forests as a result. Each forest is tested against the corresponding scene that was excluded from its training. It is essentially leave-one-out cross-validation in machine learning.

While it is possible to have a single forest for all the training scenes and test this forest against the same set of scenes, we found that this kind of experiment is prone to overfit to the training scenes. Our regression forest consists of five trees and the maximum depth of each tree is 15. The construction time of the regression forest is 30 minutes.

Approximated Optimal Weights Fig. 5.4 shows blending weights and RMS errors for selected five scenes with the combination of SPPM and MLT. Fig. 7.7 shows such results with visualization of the error per pixel for three other scenes. We compare approximated optimal weights via a trained regression forest with the average of five different runs for each scene. The blending weight is fixed to 0.5 when a pixel has no information on path features (e.g., background images). We blended two images rendered by SPPM and MLT by taking the average (Average) or by using the approximated optimal weight per pixel (Proposed). The running time of our framework is less than 50 msec for all the scenes. The storage cost of our regression forest is 100 KB. Both the running time and the storage cost are independent of the geometric complexity of the scenes. We can see that optimal weights and weights suggested by our framework are very similar to each other in almost all the cases. Our framework thus successfully learned the preference of an algorithm only based on path features.



RMSE	chess	clocks	torus	sibenik	conference
Average	0.026138	0.127614	0.025843	0.019725	0.013159
Proposed	0.020980	0.114519	0.020510	0.017612	0.011911
MLT	0.043227	0.086604	0.017765	0.016954	0.009494
SPPM	0.018740	0.140037	0.028371	0.026580	0.014602

Figure 5.4: Comparison of the optimal weights and the approximations by our framework for the selected five scenes combining MLT and SPPM. The first row shows the reference images. The bottom two rows visualize the optimal weights and the approximated weights via trained regression forests. For many scenes, our framework largely reproduces the optimal weights, without any information other than rough estimates of path features per pixel. The RMS errors between the blended images and the references are improved compare to taking the average (Average). We also show RMS errors for MLT and SPPM with the same total rendering time.

RMS Errors Fig. 5.5 shows RMS errors for 10 scenes for the combination of MLT and SPPM. We plot RMS errors of MLT, SPPM, their average, and our blended result for each scene with the total rendering time of 20 minutes for all the methods. The plots are scaled such that the values for the average is one. We can observe that our blending is superior to the average in all scenes. The reduction of error by our blending is larger when the difference of RMS errors between SPPM and MLT is large. Moreover, the blended solution by our framework sometimes outperforms a better algorithm with the same total rendering time. Such a result is not trivial since our framework spends only half of the total rendering time for each algorithm. We should also note that just taking the average can in fact increase the error for the same reason (e.g., *Cornell* scene). In contrast, we did not find any such cases using our framework. This result supports that our framework can improve the robustness of light transport simulation in practice.

Effect of Tree Depth The images in Fig. 5.6 show the approximated optimal weights for the *box* scene with different depths of the regression trees in the runtime. As the depth increases, we can observe that the preference to each technique becomes more explicit. Yet another observation is that the approximated weights are converged

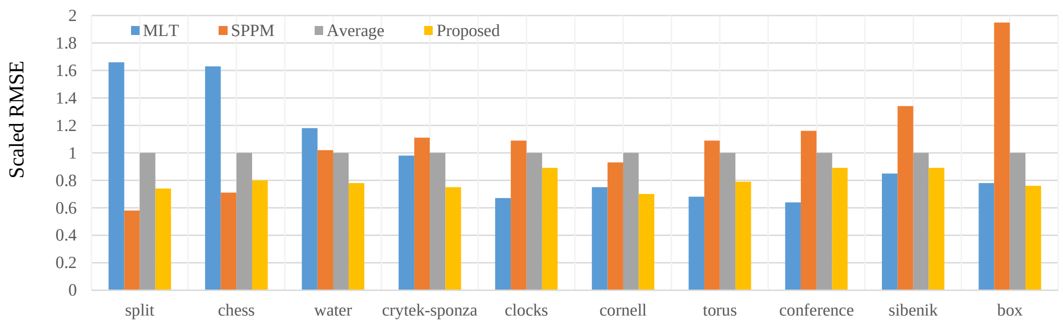


Figure 5.5: Scaled RMS errors of MLT, SPPM, Average, and blending with our framework over 10 scenes. All the methods use the total rendering time of 20 minutes. The average and our blending spends 10 minutes for both MLT and SPPM, keeping the total rendering time equal to 20 minutes. We scaled RMS errors such that the average is always one. The scenes are sorted roughly according to the difference of RMS errors between MLT and SPPM.

around the tree depth of 15. The graph in Fig. 5.6 shows the RMS error between the optimal weight and the approximated weight with our framework for this scene. We can observe that the RMS error converges around the depth of 15, and we found that it is similar for the other scenes as well. Along with the saturation of the weights, we thus conservatively set the tree depth to 15 in our experiments.

5.6 Discussion

5.6.1 Alternative to Blending

While we found that blending is a practical approach to combine different rendering algorithms, it is tempting to try *selecting* one of the different algorithms instead of *blending* such that we can spend all the allocated rendering time to one algorithm. This alternative solution, however, is not feasible for two major reasons. Firstly, as shown in Fig. 7.7, a better algorithm can change even within a single image. Even though MLT looks converged in many regions, it can entirely miss certain lighting effects such as specular reflections of caustics. As such, resolving all the effects by a single algorithm can take a significant amount of rendering time as compared to combining the results of two algorithms. Recent work on robust rendering algorithms are based on the same observation [36, 26].

Secondly, defining useful features for this selection is not trivial and algorithm-dependent. In order to select an efficient algorithm for a specific input scene, we would need a feature vector of a whole configuration of the rendering process. This information includes parameters of each rendering algorithm that affects the performance, which in turn makes the whole framework algorithm-dependent. It is also not obvious how to encode input scenes as feature vectors. Unlike images, which

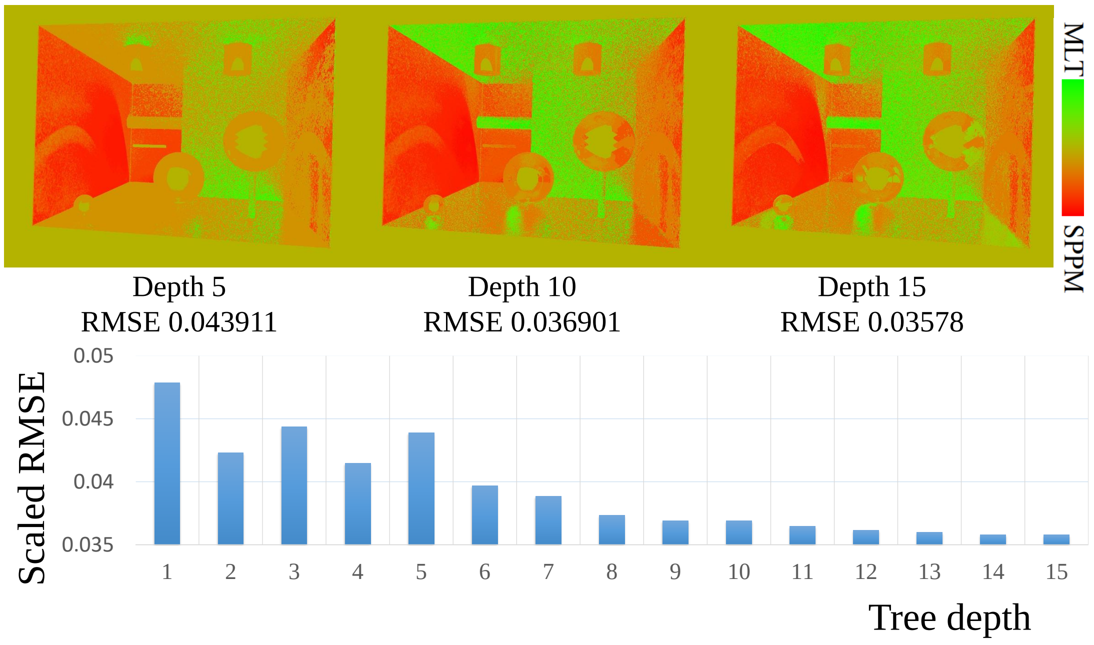


Figure 5.6: Visualization of approximated optimal weights for the *box* scene with different tree depth (top) and the corresponding plot of the approximation errors (bottom). The RMS errors of the blended images are shown under each image. As the depth of the tree increases, the color indicating the preference to MLT becomes a bit more explicit, but not significantly after a certain depth. The plot of the variance shows how approximation errors of the optimal weights change according to the tree depth, which also stops converging around the depth of 15.

contain a set of pixels in a structured manner, scene data contains a set of very different information such as material data, textures, and triangle meshes. There is no single data structure common to all of data necessary to define input scenes. This lack of a common structured input form is a striking differences to applications of machine learning for images.

One might also consider finding a distribution of total rendering time, such that we do not spend too much computation for an algorithm with small weights. This deceivingly obvious improvement, however, is not possible since our regression forest is trained under the assumption that each algorithm spends the same rendering time. Even if we can find such a distribution of rendering time somehow, optimal blending weights are now different from those at the training phase since rendering time for each algorithm is also different. To implement this idea, we would need to have multiple regression forests for all the possible distributions of total rendering time, which is likely infeasible.

5.6.2 Comparison to Neural Networks

We used regression forests as a machine learning technique to learn the relationship between path features and the optimal blending weights. One possible option is to

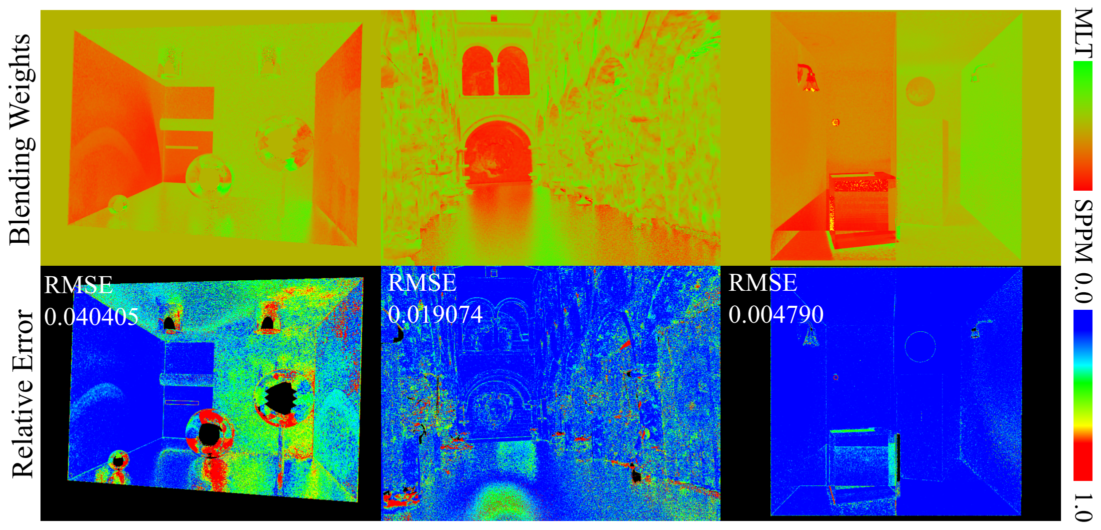


Figure 5.7: Approximated blending weights (top) and the relative errors (bottom) for the selected three scenes (*box*, *crytek-sponza*, and *water*) by replacing regression forests via a neural network.

replace it by neural networks. Given its success in the computer vision community, a deep neural network [49] is a possible candidate. We tested replacing regression forests by a fully-connected four layer's neural network using Caffe [56] on GPU as additional experiments. As shown in Fig. 5.7, we found that a neural network can achieve similar performance to regression forests. We discarded this approach in the end since even its running time is multiple orders of magnitudes slower (three minutes) than regression forests (60 msec) without much improvement in terms of RMS errors.

5.6.3 Limitations

Preparing Training Scenes In general, a machine learning technique needs a large number of training samples to avoid overfitting. While we carefully designed a set of training scenes, it is not guaranteed that the prepared training scenes are indeed sufficient for learning. This situation is in contrast to the computer vision community; there are several standardized large datasets such as ImageNet [94]. Although we used some standard models and scenes often seen in other rendering research, it would be interesting as future work to generate training scenes based on procedural modeling. This procedural modeling should include not only shapes, but also materials, lighting, and camera parameters.

Dependency on Training Scenes We found that our method works especially well if there are only slight differences between training scenes and test scenes. Fig. 5.8 shows the *torus2* scene which uses the same geometry and materials as the *torus* scene in Figure 7.7, but with a slightly different camera configuration and an

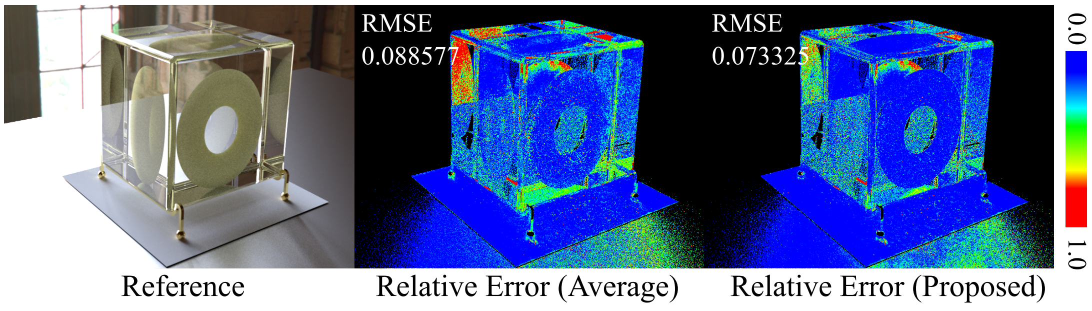


Figure 5.8: RMS errors for a test scene that is only slightly different from a training scene. This test scene is made by changing the environment light and the camera configuration while retaining the geometry and materials of the *torus* scene in Figure 7.7. For this experiment, we used the original *torus* scene for training, and the modified *torus* scene at runtime.

environment map. For this experiment, we used only the *torus* scene for the training phase, and rendered the *torus2* scene. We can observe that reduction of RMS error is significant in this case. This experiment indicates an interesting use case of our framework in practice: when an artist is modeling a new scene based on existing ones, we can train a regression forest with existing scenes beforehand.

5.7 Related Work

Light Transport Simulation in Rendering Since the development of path tracing [58], the number of light transport simulation algorithms have been developed. Among many rendering algorithms, we used the two representative approaches in our tests: SPPM [32] and MLT [117] with manifold exploration [52]. We chose these two approaches because their algorithms are completely different and have different characteristics as rendering algorithms. SPPM works by tracing a number of light paths and estimates density of light path vertices at a visible point through each pixel. MLT on the other hand traces a whole path by a Markov chain from the previously generated path and estimates the histogram of this Markov chain at all the pixels. SPPM is generally considered good at rendering caustics, while MLT is considered efficient at resolving complex visibilities from light sources. Our framework however is not restricted to use very different algorithms, since it is independent of how each algorithm works internally.

Machine Learning in Rendering Several researchers have already applied machine learning to rendering. One popular application of machine learning in rendering is regression models. Among others, Jacob et al. [53] utilized unsupervised online-learning of a Gaussian mixture model (GMM) to represent a radiance distribution in participating media. Vorba et al. [118] also used online learning of GMM to represent probability density functions for importance sampling. Ren et al. [92] introduced a realtime rendering algorithm using non-linear regression to represent precomputed radiance data. The precomputed radiance data is modeled as a multi-layered neural network [46]. The idea is to learn the relationship between scene configurations and the resulting radiance distribution based on off-line rendering with random attributes. While we also use machine learning for regression, we propose to use machine learning to combine existing rendering algorithms without any modification to them. Our framework thus can be applied on top of any of the previous work mentioned above. More recently, Nalbach et al. [80] showed how to use CNN to approximate screen-space shaders. While the goal of their work is completely different from ours, their work demonstrate the powerful potential of applying machine learning to rendering.

Kalantari et al. [59] recently proposed a image filtering technique to reduce Monte Carlo rendering noise based on the multilayer perception [46]. The idea is to learn the relationship between the scene features such as a shading location or texture values and a set of filtering parameters. Our work is inspired by their successful application and we also use machine learning to find the relationship between path features and the optimal blending weights. The difference is that their work focuses to improve the result of a single image by filtering, while we consider a situation where there are multiple rendering algorithms available for a user.

The aim of our work is to use machine learning to blend the results of different rendering algorithms. Such blending is often done by multiple importance sampling [116], and there have been many recent works on this approach [36, 26]. Our work differs from multiple importance sampling in that we treat each rendering algorithm as a black-box and does not require any detailed algorithmic information such as path probability densities.

Regression Forests Regression forests [10] are actively used in many applications. One famous example is Kinect body segmentation [99]. By simply fetching neighboring depth values and parse the regression forest, this algorithm can label each pixel by 31 different body parts quite accurately in realtime. For face recognition, Ren et al. [93] showed that regression forests can be used to detect major features such as eyes, a mouth, and a nose. Tang et al. [108] used regression forests to extract a skeletal hand model from an RGB-depth image.

For applications in computer graphics, Ladický et al. [65] used regression forest for fluid simulation and achieved x200 speed up. They trained a regression forest via position-based fluid simulation by defining several features around each particle. The trained regression forest is used to update the state of particles at the next time step, without relying on costly simulation. Inspired by the success of regression forests in many applications, we also utilize regression forests instead of a more popular convolution neural network [49]. As far as we know, our work is the first application of regression forests in rendering.

5.8 Summary

We presented a framework to automatically blend results of different light transport simulation algorithms. The key idea is to learn the relationship between a class of light transport paths and the performance of each algorithm on each class. For classification of paths, we introduced a feature vector based on relative contributions from different types of paths according to Heckbert's notation. We then calculate optimal blending weights such that a resulting image has minimal errors on average after blending. Using regression forests, we approximate a function that takes a feature vector of light transport paths and outputs the optimal blending weight per pixel. The resulting framework is independent from how each algorithm works, which makes it easily applicable to different rendering algorithms.

Chapter 6

Bridging State Spaces of Markov Chain Monte Carlo Rendering

6.1 Introduction

Physically-based rendering with light transport simulation is widely used nowadays. One class of simulation algorithms is based on Markov chain Monte Carlo (MCMC) [114]. MCMC rendering generates a Markov chain of light transport paths to follow an arbitrary user-defined target distribution. This target distribution is defined according to the contributions of light transport paths to the image, which allows MCMC algorithms to efficiently focus computation on contributing paths. In rendering, a technique for proposing the next state of a Markov chain is typically called a mutation strategy. Recent work has proposed a variety of sophisticated mutation strategies [52, 60, 33, 69].

The MCMC rendering algorithms use either the path space [114] or the primary sample space [62] as the state space of a Markov chain. The path space defines a path by a sequence of vertices, and the algorithms mutate a path by *directly* modifying its vertices. The primary sample space instead defines a path by a sequence of numbers used to generate the path, and *indirectly* mutates a path by modifying the corresponding sequence of numbers. This sequence corresponds to numbers generated by a pseudo-random number generator in regular Monte Carlo path samplers, and can be mapped to a path by path tracing [58] or bidirectional path tracing [67, 113] for example. Due to this fundamental difference between the two state spaces, a mutation strategy for one space is not applicable to the other space. For example, it is impossible to use manifold exploration [52] in the primary sample space as it is designed to work on path vertices in the path space. This situation prohibits us to take full advantage of all the advanced mutation strategies within a single rendering algorithm.

We propose a framework to *fuse* the different state spaces in MCMC rendering for the first time. The main idea is the use of an *inverse path sampler* which acts as the inverse operation of a regular path sampler. The inverse path sampler takes a complete path as input and maps it to a corresponding sequence of numbers in a unit hypercube. Since this mapping uses the inverse of the inverse cumulative distribution functions (CDFs) used in importance sampling, it requires just the CDFs themselves which are often readily available. Our formulation does not impose any modification to mutation strategies themselves, making it easy to use it with existing implementations. We tested our formulation for the combination of manifold exploration [52] and multiplexed Metropolis light transport [33]. The results demonstrate that this combination robustly handles scenes with different characteristics where using only one of the algorithms fails. To summarize, our contributions are:

- A novel framework which fuses the two state spaces currently used for Markov chain Monte Carlo rendering,
- Introduction of the concept of an inverse path sampler,
- Demonstration of MCMC rendering with a combination of mutation strategies mixing both state spaces.

6.2 Related Work

Light Transport Simulation Rendering algorithms based on Monte Carlo integration are primarily characterized by how they generate paths connecting a light source to a sensor. Path tracing [58] generates a path starting from the sensor, light tracing [5] traces from a light source, and bidirectional path tracing [67, 113] from both sides with deterministic connections of subpaths.

Another family of approaches is based on *photon density estimation* [98], such as (progressive) photon mapping [55, 35], which estimates illumination using the density of light subpath vertices. Recent work [26, 36] combines Monte Carlo integration and photon density estimation into a single rendering algorithm.

We employ existing Monte Carlo integration approaches as *path samplers*, which can generate a light transport path from a sequence of numbers. A path sampler in our formulation can be any of the existing approaches as long as they are based on the path integral formulation [112]. We introduce the inverse of such a path sampler and show how it can be used in Markov chain Monte Carlo (MCMC) rendering with fused state spaces.

MCMC in Path Space Veach and Guibas [114] introduced Markov chain Monte Carlo methods to rendering. The resulting algorithm, Metropolis Light Transport (MLT), perturbs the vertices of a path and generates a history of paths based on the Metropolis-Hastings algorithm. Since MLT directly manipulates vertices, it works within the path space of the path integral formulation.

Manifold exploration [52] is also built on the original MLT framework and extends its mutation strategy to efficiently handle a chain of specular and highly glossy events. Half vector space light transport [60, 39] represents paths by their endpoints and half vectors at the interactions between them. This representation has been shown to flatten the target sampling distribution, which makes it easier to sample by Markov chain Monte Carlo algorithms.

Path space algorithms are often efficient at rendering certain effects since we can explicitly consider characteristics of such effects in mutations (e.g., caustics in the caustic mutation [114]). We show how to incorporate such efficient mutations into the other class of MCMC rendering algorithms.

MCMC in Primary Sample Space Kelemen et al. [62] introduced an alternative formulation of MLT based on the primary sample space. The algorithm, primary sample space MLT (PSSMLT), indirectly mutates paths by perturbing a vector of (random) numbers that is used to generate paths. They showed how this formulation significantly simplifies the MCMC process and flattens the target distribution by utilizing the information of the probability density function of a given path sampler. Li et al. [69] showed how to achieve locally adaptive anisotropic mutations in the primary sample space based on the approximation of Hessian-Hamiltonian dynamics. The resulting algorithm demonstrates robust sampling even in the presence of complex light transport paths.

If PSSMLT is used with bidirectional path tracing as a path sampler, it generates a family of bidirectional paths, rather than a single path such as MLT. This difference makes the connection between the primary sample space and the path space ambiguous since one sample in the primary sample space corresponds to a family of paths. Hachisuka et al. [33] showed that this ambiguity can be resolved simply by extending the primary sample space by another dimension, encoding the type of the used bidirectional technique. They showed how this multiplexed primary sample space leads to a method that can distribute samples based on the weighted primary sample spaces according to multiple importance sampling [113]. We show how to combine the path space MLT techniques and the primary sample space MLT techniques for the first time.

Bridging Sampling Spaces The half vector space [60] can be considered as yet another space and one possibility to bridge the half vector space and the path space. We instead consider bridging the primary sample space and the path space. Concurrent works [87, 9] also use an inverse mapping from the primary sample space to the path space. Although their methods are different from ours, the underlying concept is equivalent.

6.3 State Spaces of MCMC Rendering

This section briefly introduces the path space and primary sample space variants of MLT. We also recapitulate the multiplexed primary space, as it has beneficial properties that lead to our framework.

6.3.1 Path Space

Light transport simulation computes the solution of the *path integral* [112]. It determines the intensity of the j-th pixel in the image as the integral over the measurement contribution function $f_j(\bar{x})$ with respect to the product area measure μ:

$$I_j = \int_{\mathcal{P}} f_j(\bar{x}) d\mu(\bar{x}), \tag{6.1}$$

where $\mathcal{P}$ is the *path space* which comprises all paths of all possible lengths ($\mathcal{P} \equiv \cup_{k=2}^{\infty} \mathcal{P}_k$) and $\mathcal{P}_k$ denotes a sub-space containing paths with $k = [2, \dots, \infty)$ vertices. An individual *path* $\bar{x} \equiv (\mathbf{x}_1, \dots, \mathbf{x}_k) \in \mathcal{P}_k$ is defined as a sequence of points on the scene's surfaces. Note that we do not consider participating media in this work.

MLT [114] uses the Metropolis-Hastings algorithm [42] to sample paths: the path space $\mathcal{P}$ serves as the state space and MLT generates a sequence of samples that follow f^*/b as the target distribution. Here f^* is a scalar contribution function (typically luminance) proportional to the measurement contribution f, and $b \equiv \int_{\mathcal{P}} f^*(\bar{x}) d\mu(\bar{x})$ is the normalization constant, which is estimated with regular Monte Carlo techniques. Using this sequence of samples, the estimate $\hat{I}_j$ becomes

$$I_j \approx \hat{I}_j \equiv \frac{b}{N} \sum_{i=1}^{N} \frac{f(\bar{x}_i)}{f^*(\bar{x}_i)}. \tag{6.2}$$

A tentative sample $\bar{y}$ is generated based on the current path $\bar{x}_i \in \mathcal{P}$ and according to the transition kernel $T_{\mathcal{P}}$, i.e., $\bar{y} \sim T_{\mathcal{P}}(\bar{x}_i \longrightarrow \cdot)$. For a Metropolis-Hastings update, the path $\bar{y}$ is accepted as the next state $\bar{x}_{i+1}$ with an *acceptance probability* of $\min(1, a(\bar{x}_i \longrightarrow \bar{y}))$ where

$$a(\bar{x}_i \longrightarrow \bar{y}) = \frac{f^*(\bar{y}) T_{\mathcal{P}}(\bar{y} \longrightarrow \bar{x}_i)}{f^*(\bar{x}_i) T_{\mathcal{P}}(\bar{x}_i \longrightarrow \bar{y})}. \tag{6.3}$$

Otherwise $\bar{x}_i$ is kept as the current state ($\bar{x}_{i+1} = \bar{x}_i$).

6.3.2 Primary Sample Space

PSSMLT [62] simplified the original MLT algorithm by using the space of uniform random numbers as the state space, based on the observation that paths are sampled by a sequence of random numbers (Fig. 6.1, left). This state space is denoted as the

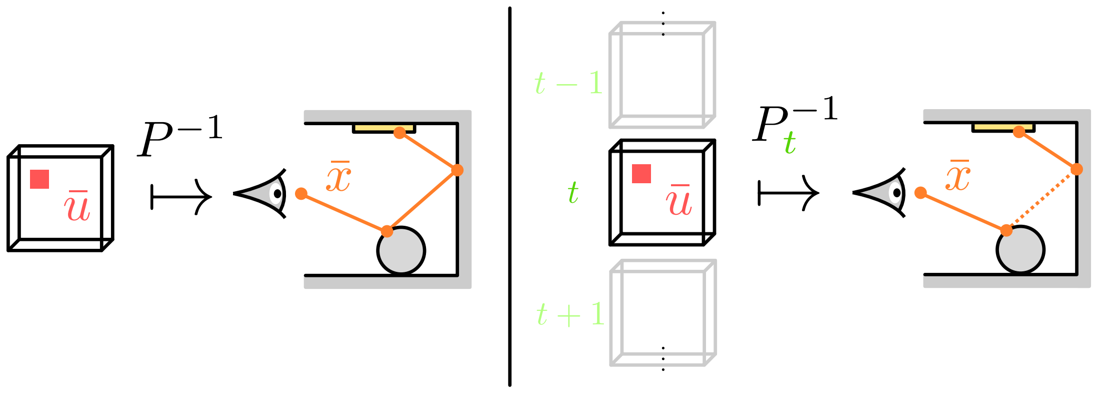


Figure 6.1: Relationship between the path space and the primary sample space (left), and the multiplexed primary sample space (right). For primary sample space, the random number $\bar{u}$ is mapped to the path $\bar{x}$ using the inverse cumulative distribution function P^{-1}. Multiplexed primary sample space extends the primary sample space by the index of the sampling strategy t corresponding to the strategy of the bidirectional path samplers. In this example, a path with length three is mapped from $\bar{u}$ with path connection between the second and third vertices ($t = 2$).

primary sample space, described as the unit hypercube $\mathcal{U} = [0,1]^{O(k)}$ ($O(k)$ random numbers are normally required to define a path of length k).

The relationship between a sample in the state space $\bar{u} \in \mathcal{U}$ and a path $\bar{x} \in \mathcal{P}$ can be written as the inverse of the cumulative distribution function $P^{-1} : \mathcal{U} \longrightarrow \mathcal{P}$. The actual mapping of $\bar{u}$ to a path $P^{-1}(\bar{u})$ is obtained by using the underlying path sampler and using the random number sequence $\bar{u}$. With P^{-1} Eq. 6.1 can be rewritten as

$$I = \int_{\mathcal{U}} \tilde{f}(\bar{u}) \left| \frac{d\mu(\bar{x})}{d\bar{u}} \right| d\bar{u} = \int_{\mathcal{U}} \tilde{f}(\bar{u}) \left| \frac{dP^{-1}(\bar{u})}{d\bar{u}} \right| d\bar{u} = \int_{\mathcal{U}} \tilde{C}(\bar{u}) d\bar{u}, \tag{6.4}$$

where $\tilde{C}$ is the path throughput in primary sample space. We use the tilde to explicitly express the dependence on random numbers $\bar{u}$ instead of the path space vertices $\bar{x}$, i.e., $\tilde{C}(\bar{u}) = C(P^{-1}(\bar{u})) = C(\bar{x})$, where $C(\bar{x}) = f(\bar{x})/p(\bar{x})$ is the path contribution in path space.

Now using an appropriate scalar target function $\tilde{C}^*(\bar{u}_i)$ (the luminance of $\tilde{C}(\bar{u})$), MCMC can generate a sequence of samples distributed according to $\tilde{C}(\bar{u})$. That is, using a Markov chain $\bar{u}_i \in \mathcal{U}$ with N samples, we can compute an estimate $\langle I \rangle$ of I (Eq. 6.4) as

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{\tilde{C}(\bar{u}_i)}{\tilde{C}^*(\bar{u}_i)/b} = \frac{b}{N} \sum_{i=1}^{N} \frac{\tilde{C}(\bar{u}_i)}{\tilde{C}^*(\bar{u}_i)}, \tag{6.5}$$

where $b = \int_{\mathcal{U}} \tilde{C}^*(\bar{u}) d\bar{u}$ is the normalization constant, which again is estimated with regular Monte Carlo techniques.

PSSMLT also uses the Metropolis-Hasting algorithm to generate the next sample.

Given a current state $\bar{u}_i$, we first generate the next tentative state $\bar{v}$ from the proposal distribution q, i.e., $\bar{v} \sim T_{\mathcal{U}}(\bar{u}_i \longrightarrow \bar{v})$, which is accepted as $\bar{u}_{i+1}$ with the probability:

$$a(\bar{u}_i \longrightarrow \bar{v}) = \frac{\tilde{C}^*(\bar{v}) T_{\mathcal{U}}(\bar{v} \longrightarrow \bar{u}_i)}{\tilde{C}^*(\bar{u}_i) T_{\mathcal{U}}(\bar{u}_i \longrightarrow \bar{v})}, \tag{6.6}$$

and $\bar{u}_{i+1} = \bar{u}_i$ otherwise. A symmetric proposal distribution avoids the computation of the transition probabilities and Eq. 6.6 becomes

$$a(\bar{u}_i \longrightarrow \bar{v}) = \frac{\tilde{C}^*(\bar{v})}{\tilde{C}^*(\bar{u}_i)}. \tag{6.7}$$

In general, there is no one-to-one mapping between primary space and a path since one primary space sample can be mapped to a set of paths (e.g., with bidirectional path tracing).

Multiplexed Primary Sample Space Hachisuka et al. [33] extended the formulation of PSSMLT to facilitate multiple importance sampling (MIS) [113] within this framework (Fig. 6.1, right). MIS combines multiple sampling strategies $p_t(\bar{x})_{t=1,\dots,M}$ into an estimate I:

$$\begin{aligned} I &= \int_{\mathcal{P}} \sum_{t=1}^{M} w_t(\bar{x}) f(\bar{x}) d\mu(\bar{x}) = \sum_{t=1}^{M} \int_{\mathcal{P}} w_t(\bar{x}) f(\bar{x}) d\mu(\bar{x}) && (6.8)\\ &\approx \sum_{t=1}^{M} \frac{1}{N_t} \sum_{i=1}^{N_t} w_t(\bar{x}_{t,i}) C_t(\bar{x}_{t,i}), && (6.9) \end{aligned}$$

where M is the number of techniques and $w_t(\bar{x})$ are the MIS weights satisfying $\sum_{t=1}^{M} w_t(\bar{x}) = 1$, and $C_t(\bar{x}_{t,i}) = f(\bar{x}_i)/p_t(\bar{x})$ is the throughput of the path $\bar{x}_i$ generated with a technique t.

Given the sampling technique t, we can write the relationship between $\bar{u}$ and $\bar{x}$ with the mapping P_t^{-1} associated with the technique t. Similar to PSSMLT, Eq. 6.8 can be written using this mapping as:

$$\begin{aligned} I &= \sum_{t=1}^{M} \int_{\mathcal{U}} \tilde{w}_t(\bar{u}) \tilde{f}(\bar{u}) \left| \frac{dP_t^{-1}(\bar{u})}{d\bar{u}} \right| d\mu(\bar{u}) \\ &= \sum_{t=1}^{M} \int_{\mathcal{U}} \tilde{w}_t(\bar{u}) \tilde{C}_t(\bar{u}) d\mu(\bar{u}) = \int_{\mathcal{U}} \sum_{t=1}^{M} \tilde{w}_t(\bar{u}) \tilde{C}_t(\bar{u}) d\mu(\bar{u}). \end{aligned} \tag{6.10}$$

Instead of using $\mathcal{U}$ for each t as a state space, MMLT utilizes an extended space named *multiplexed primary sample space* $\mathcal{U} \times \mathcal{T}$ where $\mathcal{T} = \{1, \dots, M\}$, facilitating the idea of *serial tempering* [74]. This method explores the state spaces $\mathcal{U}$ parameterized by the parameter t, and also facilitates Markov chain updates between the two different parameters t and $\bar{u}$. Using this method, we can sample the states according to $\sum_{t=1}^{M} \tilde{w}_t(\bar{u}) \tilde{C}_t(\bar{u})$ and estimate the last expression in Eq. 6.10 using a single

Markov chain.

The Metropolis-Hasting update for MMLT considers two types of mutations: (1) a mutation within the same technique, and (2) a mutation among the different techniques. Both types of mutations can be considered in a *single* update with the acceptance ratio $\min(1, a([\bar{u}, t] \longrightarrow [\bar{v}, t']))$ using

$$a([\bar{u}, t] \longrightarrow [\bar{v}, t']) = \frac{\tilde{w}_{t'}(\bar{v})\tilde{C}_{t'}(\bar{v})T_{V'}([\bar{v}, t'] \longrightarrow [\bar{u}, t])}{\tilde{w}_t(\bar{u})\tilde{C}_t(\bar{u})T_{V'}([\bar{u}, t] \longrightarrow [\bar{v}, t'])}. \tag{6.11}$$

6.3.3 Discussion

MLT based on the path space is good for local explorations, as it can selectively recreate parts of a path. PSSMLT variants usually need to re-trace all path segments after updating the random numbers.

On the other hand, path space MLT has problems with global discovery of important "islands" in path space. This is because the only mutation strategy designed to fulfill ergodicity, the bidirectional mutation, needs to fix the number of eye and light subpath vertices upfront. This is required to be able to evaluate the transition kernel $T(\bar{x}_i \longrightarrow .)$, and often leads to very low acceptance rates.

This a-priori decision for one particular technique is similar in spirit to MMLT, but MMLT also performs a local exploration of the state space. The MMLT formulation has one more property which is important to us: we are able to give a mapping from random numbers to path vertices.

6.4 Fusing the State Spaces

Our goal is to enable the use of mutation strategies from different state spaces, in particular from the (multiplexed) primary sample space and the path space as in MLT, in a single framework. We explain how to incorporate path space mutations into MMLT by introducing the concept of an *inverse path sampler*.

6.4.1 Paths to Numbers

An *inverse path sampler* returns a sequence of numbers $\bar{u}$ from a given path $\bar{x}$. While such a mapping cannot be uniquely determined in PSSMLT with the BDPT sampler, the multiplexed primary sample space leads to a straightforward derivation as follows.

Revisiting Path Samplers In order to find such an inverse mapping, we need to revisit the precise meaning of a path sampler $P_t^{-1}(\bar{u}) = \bar{x}$. In general, one can factorize

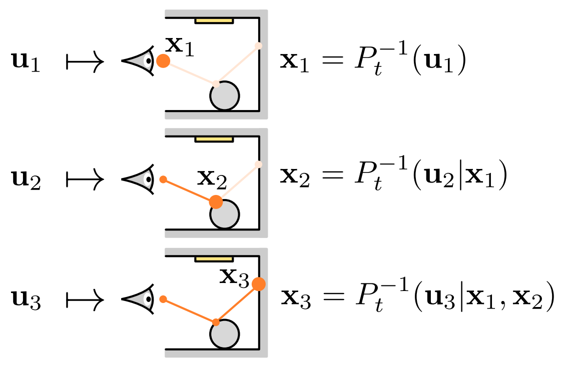

$\mathbf{u}_1 \mapsto$
$\mathbf{x}_1$
$\mathbf{x}_1 = P_t^{-1}(\mathbf{u}_1)$
$\mathbf{u}_2 \mapsto$
$\mathbf{x}_2$
$\mathbf{x}_2 = P_t^{-1}(\mathbf{u}_2|\mathbf{x}_1)$
$\mathbf{u}_3 \mapsto$
$\mathbf{x}_3$
$\mathbf{x}_3 = P_t^{-1}(\mathbf{u}_3|\mathbf{x}_1, \mathbf{x}_2)$

Figure 6.2: Illustration of path sampling. A sequence of numbers $\bar{u} = (\mathbf{u}_1, \mathbf{u}_2, ...)$ is successively transformed by inverse CDFs P_t^{-1}. This process creates the path vertices $\mathbf{x}_1, \mathbf{x}_2, ...$ one after the other, depending on the chosen technique t and the path vertices that have already been sampled so far.

a multivariate CDF $P_t(\bar{x}) = P_t(\mathbf{x}_1, \cdots, \mathbf{x}_k)$ into a product of conditional CDFs:

$$\begin{aligned} &P_t(\mathbf{x}_1, \cdots, \mathbf{x}_k) = \\ &P_t(\mathbf{x}_1)P_t(\mathbf{x}_2|\mathbf{x}_1) \cdots P_t(\mathbf{x}_k|\mathbf{x}_1, \cdots, \mathbf{x}_{k-1}), \end{aligned} \tag{6.12}$$

where $P_t(\mathbf{x}_k|\mathbf{x}_1, \cdots, \mathbf{x}_{k-1})$ is a CDF of $\mathbf{x}_k$ given $\mathbf{x}_1, \cdots, \mathbf{x}_{k-1}$. Since all the conditional CDFs are different CDFs, $P_t(\mathbf{x}_2|\mathbf{x}_1)$ strictly should be written as $P_{t,\mathbf{x}_1}(\mathbf{x}_2|\mathbf{x}_1)$ for example. We use the notation $P_t(\mathbf{x}_2|\mathbf{x}_1)$ throughout this paper for brevity.

Using this factorization, the multivariate variant of inverse transform sampling generates $(\mathbf{x}_1, \cdots, \mathbf{x}_k) \sim p_t(\mathbf{x}_1, \cdots, \mathbf{x}_k)$ by a sequence of inverse transform sampling (see Fig. 6.2):

$$\begin{aligned} \mathbf{x}_1 &= P_t^{-1}(\mathbf{u}_1) \\ \mathbf{x}_2 &= P_t^{-1}(\mathbf{u}_2|\mathbf{x}_1) \\ &\cdots \\ \mathbf{x}_k &= P_t^{-1}(\mathbf{u}_k|\mathbf{x}_1, \cdots, \mathbf{x}_{k-1}). \end{aligned} \tag{6.13}$$

This factorization approach, in general, is not very practical since an analytical form of the inverse of a conditional CDF is not often available. In typical Monte Carlo rendering systems, however, we can rely on this exact approach by generating each vertex $\mathbf{x}_i$ based on the previous vertices $\mathbf{x}_{i-1}, \cdots, \mathbf{x}_1$ (usually only on $\mathbf{x}_{i-1}$ and $\mathbf{x}_{i-2}$) according to a PDF proportional to the BRDF at $\mathbf{x}_{i-1}$.

This approach is often denoted as $P_t^{-1}(\bar{u}) = \bar{x}$, but it is not precise. The exact meaning of $P_t^{-1}(\bar{u}) = \bar{x}$ is the above sequential approach. We folded common cases where we need to use multiple numbers to sample a vertex in each CDF for the sake of brevity, but one can generally also factorize them into conditional univariate CDFs similar to above.

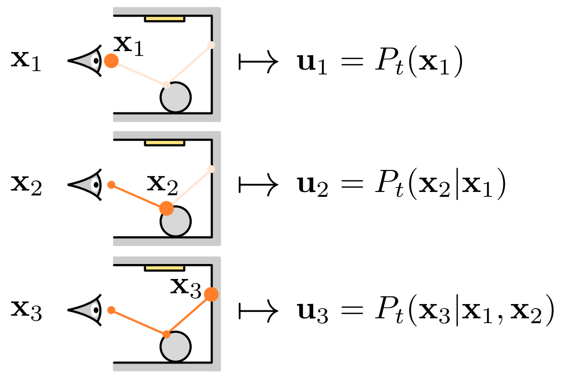

$\mathbf{x}_1$
$\mathbf{x}_1$
$\mapsto \mathbf{u}_1 = P_t(\mathbf{x}_1)$
$\mathbf{x}_2$
$\mathbf{x}_2$
$\mapsto \mathbf{u}_2 = P_t(\mathbf{x}_2|\mathbf{x}_1)$
$\mathbf{x}_3$
$\mathbf{x}_3$
$\mapsto \mathbf{u}_3 = P_t(\mathbf{x}_3|\mathbf{x}_1, \mathbf{x}_2)$

Figure 6.3: Illustration of inverse path sampling. Analogous to creating a path from random numbers (Fig. 6.2), we can invert the process and successively compute random numbers which would have created the given path via path sampling. This is possible if the CDFs to create the vertices are bijections.

Inverse Path Samplers This precise definition leads to the following inverse sampler from $\bar{x}$ to $\bar{u} = (\mathbf{u}_1, \cdots, \mathbf{u}_k)$ (see Fig. 6.3):

$$\begin{aligned}
&P_t(\mathbf{x}_1) = \mathbf{u}_1 \\
&P_t(\mathbf{x}_2|\mathbf{x}_1) = \mathbf{u}_2 \\
&\cdots \\
&P_t(\mathbf{x}_k|\mathbf{x}_1, \cdots, \mathbf{x}_{k-1}) = \mathbf{u}_k.
\end{aligned} \tag{6.14}$$

We assume that each CDF is already available and can be evaluated easily. This assumption holds in practice since we usually define the CDF first in order to derive its inverse in path samplers. Note that the technique index t is unaffected due to the use of the multiplexed primary sample space.

While the definition of inverse path samplers might look trivial after the fact, such CDFs have not been used for practical purposes in rendering so far. Our work shows that they can be used for fusing the state spaces in MCMC rendering. We provide some examples of CDFs and how they can be trivially defined.

6.4.2 Examples of Inverse Path Samplers

Example 1: Cosine distribution The cosine distribution is used to sample a direction according to the cosine of the angle from the surface normal. Since there are multiple algorithms to sample such a direction, we provide an example based on Malley's method [71]. The method uses the polar coordinates (r, θ) of the projected direction onto the tangent plane around the normal. The PDF of the cosine distribution in this case is $p(r, \theta) = r/\pi$, so the marginal and conditional densities for r and

θ are

$$p(r) = \int_0^{2\pi} p(r,\theta)d\theta = 2r \qquad p(\theta|r) = \frac{p(r,\theta)}{p(r)} = \frac{1}{2\pi}. \tag{6.15}$$

The mapping from (r,θ) to (u_1,u_2) is

$$\begin{aligned} u_1 &= P(r) = \int_0^r p(r)dr = r^2, \\ u_2 &= P(\theta|r) = \int_0^\theta p(\theta|r)d\theta = \frac{\theta}{2\pi}, \end{aligned} \tag{6.16}$$

Example 2: Uniform sampling on a triangle Uniform sampling on a triangle is often used to generate an initial vertex of a light path. Similar to the previous example, we sample a point in different coordinates and convert to the surface point. Typically, we transform to the barycentric coordinates of the isosceles right triangle for this purpose. Given a point with barycentric coordinates (b_1,b_2), the mapping from (b_1,b_2) to (u_1,u_2) is

$$\begin{aligned} u_1 &= P(b_1) = \int_0^{b_1} p(b_1)db_1 = 2b_1 - b_1^2, \\ u_2 &= P(b_2|b_1) = \int_0^{b_2} p(b_2|b_1)db_2 = \frac{b_2}{1-b_1}. \end{aligned} \tag{6.17}$$

Example 3: Tabulated PDF In some cases, such as sampling a direction according to an environment map, we generate samples according to a tabulated PDF. Such a tabulated PDF $p(x)$ can be defined as a piecewise-constant function with N bins over $[x_0,x_N]$:

$$p(x) = \begin{cases} p_j & x \in [x_0, x_N] \\ 0 & \text{otherwise} \end{cases}, \tag{6.18}$$

where $j = \lfloor N\frac{x-x_0}{x_N-x_0} \rfloor + 1$. We consider a one dimensional CDF for simplicity. The CDF in this case becomes a piecewise-linear function, and the mapping from x to u is

$$u = P(x) = \begin{cases} P_j + p_j N(x - \frac{j-1}{N}) & x \in [x_0, x_N] \\ 0 & x \in x < x_0 \\ 1 & x \in x > x_N \end{cases}, \tag{6.19}$$

where $P_j = \sum_{k=1}^{j-1} \frac{p_k}{N}$. The summation for P_j can be accelerated by precomputing P_j and finding a bin index using binary search.

Example 4: GGX distribution There are various microfacet normal distributions and the shape of the distributions can be complex. The distributions, however, are well designed to make it possible to derive CDFs for importance sampling. The GGX

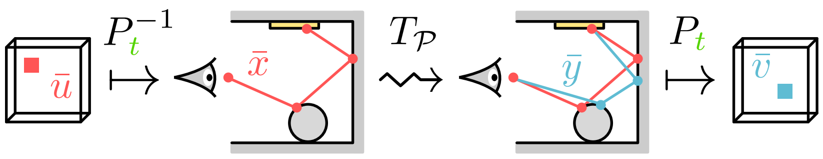


Figure 6.4: We perform path space mutations by first converting the Markov chain state from our base space (multiplexed primary sample space) to path space. This is done using the standard Monte Carlo path sampler, applying P_t^{-1} to the random number sequence. We then mutate the state in path space via $T_{\mathcal{P}}$ as regular MLT does, without any modifications. Lastly, we convert the resulting path back to the base state space by inverting the path sampling process and applying P_t to the path to compute the random number sequence.

distribution [120] controls the width of the distribution with a parameter α. The PDF in spherical coordinates (θ, ϕ) is

$$p(\theta, \phi) = \frac{\alpha^2 \sin\theta}{\pi \cos^3\theta(\alpha^2 + \tan^2\theta)^2}. \tag{6.20}$$

Following similar steps as previous examples, we obtain the marginal and conditional CDFs for θ and ϕ, and the mapping from (θ, ϕ) to $\mathbf{u} = (u_1, u_2)$ is

$$\begin{aligned} u_1 &= P(\theta) = \int_0^\theta p(\theta, \phi) d\theta = \frac{\tan^2\theta}{\alpha^2 + \tan^2\theta}, \\ u_2 &= P(\phi|\theta) = \int_0^\phi p(\phi|\theta) d\phi = \frac{\phi}{2\pi}. \end{aligned} \tag{6.21}$$

6.4.3 Fusing State Spaces with Inverse Path Samplers

With the definition of inverse path samplers, we now have a mapping between the path space $\mathcal{P}$ and the primary sample space $\mathcal{U}$ for simple unidirectional samplers. Since we keep the technique t fixed up front, we can now also formulate a mapping between the two state spaces. In the following we consider paths with equal lengths since the change of path lengths would complicate the inverse mapping. Similar to the implementation of MMLT by Hachisuka et al. [33], we use a separate Markov chain for each path length. In the following discussion, we opt to use the multiplexed primary sample space as the base space since we found that it naturally incorporates our formulation.

Fig. 6.4 illustrates this process: in order to perform a mutation in the path space, we first map the current state $(\bar{u}, t)$ to a path $\bar{x} = P_t^{-1}(\bar{u})$ using the current sampling strategy t and the path sampler. We then apply a mutation and obtain the new path $\bar{y} \in \mathcal{P}$ according to $T_{\mathcal{P}}(\bar{x} \longrightarrow \bar{y})$. Using the inverse path sampler, we convert it back to $\mathcal{U}$ with $\bar{v} = P_t(\bar{y})$. The transition kernel in the path space $T_{\mathcal{P}}(\bar{x} \longrightarrow \bar{y})$ is transformed to

$$T_{\mathcal{P}}(\bar{x} \longrightarrow \bar{y}) = \tilde{T}_{\mathcal{P},t}(\bar{u} \longrightarrow \bar{v}) \left| \frac{dP_t^{-1}(\bar{v})}{d\bar{v}} \right|, \tag{6.22}$$

where $\tilde{T}_{\mathcal{P},t}(\bar{u} \longrightarrow \bar{v}) = T_{\mathcal{P}}(P_t^{-1}(\bar{u}) \longrightarrow P_t^{-1}(\bar{v}))$. The acceptance ratio for the mapped paths then becomes

$$a([\bar{u}, t] \longrightarrow [\bar{v}, t]) = \frac{\tilde{w}_t(\bar{v})\tilde{C}_t(\bar{v})\tilde{T}_{\mathcal{P},t}(\bar{v} \longrightarrow \bar{u}) \left| \frac{dP_t^{-1}(\bar{u})}{d\bar{u}} \right|}{\tilde{w}_t(\bar{u})\tilde{C}_t(\bar{u})\tilde{T}_{\mathcal{P},t}(\bar{u} \longrightarrow \bar{v}) \left| \frac{dP_t^{-1}(\bar{v})}{d\bar{v}} \right|}, \tag{6.23}$$

where $\left|dP^{-1}(\bar{u})/d\bar{u}\right| = 1/\tilde{p}(\bar{u})$ and $\left|dP^{-1}(\bar{v})/d\bar{v}\right| = 1/\tilde{p}(\bar{v})$ by the definition of CDFs. Eq. 6.23 is in fact equal to

$$a([\bar{u}, t] \longrightarrow [\bar{v}, t]) = \frac{\tilde{w}_t(\bar{v})\tilde{f}(\bar{v})\tilde{T}_{\mathcal{P},t}(\hat{v} \longrightarrow \hat{u})}{\tilde{w}_t(\bar{u})\tilde{f}(\bar{u})\tilde{T}_{\mathcal{P},t}(\hat{u} \longrightarrow \hat{v})} \tag{6.24}$$

since we have

$$\tilde{C}_t(\bar{v}) \left| \frac{dP^{-1}(\bar{u})}{d\bar{u}} \right| = \frac{\tilde{C}_t(\bar{v})}{\tilde{p}(\bar{u})} = \frac{\tilde{f}(\bar{v})}{\tilde{p}(\bar{u})\tilde{p}(\bar{v})} \tag{6.25}$$

$$\tilde{C}_t(\bar{u}) \left| \frac{dP^{-1}(\bar{v})}{d\bar{v}} \right| = \frac{\tilde{C}_t(\bar{u})}{\tilde{p}(\bar{v})} = \frac{\tilde{f}(\bar{u})}{\tilde{p}(\bar{u})\tilde{p}(\bar{v})}. \tag{6.26}$$

We thus do not need to modify existing implementations of mutation strategies in order to use them in conjunction with another space.

6.4.4 Handling Deterministic Cases

Deterministic distributions such as perfect reflectors become a PDF with a delta function. We cannot uniquely determine the inverse mapping in this case, since no component of $\bar{u}$ was actually used to generate a sample from such distributions. To solve this issue, we use the lower dimensional subspace $\mathcal{U}_* \subset \mathcal{U}$ excluding all the components related to the delta distributions, which corresponds to a specular manifold in the path space [52]. Using this subspace, the function $P(\bar{u}_*)$ defined on $\mathcal{U}_*$ becomes invertible. We fill the rest of the components with uniform random numbers, which is a special case of reversible jump MCMC [29]. This approach is also equivalent to lazy mutation in PSSMLT, which also accounts for the case where there is a difference in the number of dimensions between MCMC states.

Algorithm 1 The mutation step with the combined proposal distribution. The mutation is limited to maintain the path length. Given $[\bar{u}, t] \in \mathcal{U} \times \mathcal{T}$ as the current state, this algorithm computes the next state $[\bar{u}_*, t_*]$.

```
if primary space space mutation is selected then
    $[\bar{v}, t'] \sim T_{\mathcal{U}}([\bar{u}, t] \longrightarrow \cdot)$
    $a \leftarrow \min\left(1, \frac{\tilde{w}_{t'}(\bar{v})\tilde{C}_{t'}(\bar{v})T_{\mathcal{U}}([\bar{v},t'] \longrightarrow [\bar{u},t])}{\tilde{w}_t(\bar{u})\tilde{C}_t(\bar{u})T_{\mathcal{U}}([\bar{u},t] \longrightarrow [\bar{v},t'])}\right)$        ▷ Eq. 6.11
    if Random() < $a$ then
        $[\bar{u}_*, t_*] \leftarrow [\bar{v}, t']$
    else
        $[\bar{u}_*, t_*] \leftarrow [\bar{u}, t]$
    end if
else
    $\bar{x} \leftarrow P_t^{-1}(\bar{u})$
    $\bar{y} \sim T_{\mathcal{P}}(\bar{x} \longrightarrow \cdot)$
    $a \leftarrow \min\left(1, \frac{\tilde{w}_t(\bar{v})\tilde{f}(\bar{v})\tilde{T}_{\mathcal{P},t}(\hat{v} \longrightarrow \hat{u})}{\tilde{w}_t(\bar{u})\tilde{f}(\bar{u})\tilde{T}_{\mathcal{P},t}(\hat{u} \longrightarrow \hat{v})}\right)$        ▷ Eq. 6.24
    if Random() < $a$ then
        $[\bar{u}_*, t_*] \leftarrow [P_t(\bar{y}), t]$
    else
        $[\bar{u}_*, t_*] \leftarrow [\bar{u}, t]$
    end if
end if
```

The transition kernel in this case becomes

$$\begin{aligned} T_{\mathcal{P}}(\bar{x} \longrightarrow \bar{y}) &= \tilde{T}_{\mathcal{P},t}(\bar{u} \longrightarrow \bar{v}) \left| \frac{dP_t^{-1}(\bar{v}_*)}{d\bar{v}_*} \right| \left| \frac{d(\bar{v}_*, \bar{r})}{d\bar{v}} \right| \\ &= \tilde{T}_{\mathcal{P},t}(\bar{u} \longrightarrow \bar{v}) \left| \frac{dP_t^{-1}(\bar{v}_*)}{d\bar{v}_*} \right|. \end{aligned} \tag{6.27}$$

The conversion from $(\bar{v}_*, \bar{r})$ to $\bar{v}$ is just a permutation so we have $\left| \frac{d(\bar{v}_*, \bar{r})}{d\bar{v}} \right| = 1$. This eventually yields the same equation as Eq. 6.24.

It is also possible to handle layered materials in a similar way as described above. Instead of reflection or transmission for specular surfaces, we can assign a selected lobe, e.g., diffuse or glossy, as a material type. When we want to convert a path to random numbers, we can then choose a uniformly distributed random number in the appropriate range that maps to this lobe. This treatment is similar in spirit to different mathematical formulations of concurrent works [87, 9].

6.5 Implementation

We implemented our proposed method in our renderer, including primary sample space mutation techniques as well as the path space mutation techniques in MLT [114] and manifold exploration [52]. As for manifold exploration, we only implemented the variant for specular surfaces and omitted handling of glossy surfaces. The path space mutation techniques do not contain any code specific to our technique, so we can reuse them for MLT without modifications. Similar to MMLT, our method separates the integral into path lengths and generates one Markov chain for each path length. Therefore the bidirectional mutation which involves a change in path length cannot be used in combination with our technique. Instead we implemented a fixed length variant of the bidirectional mutation. Similar to MLT, our method combines various mutation techniques and thus the selection of the mutation strategy has a great influence on the efficiency. Following the suggestion in Section 5.3.4 in the MLT paper [114], we randomly select a *valid* mutation strategy given the current state to avoid meaningless selections which would always be rejected. We parallelize the algorithm by computing several Markov chains assigned to different threads.

The core of the technique is almost the same as MMLT. We maintain the state in the multiplexed primary sample space and initialize it with random numbers. In our method, on the other hand, the mutation process in MMLT is replaced by Algorithm 1. The entire flow of the algorithm is a mixture of two Metropolis-Hastings updates for each state space. One difference is that we need to multiply MIS weights in addition to the measurement contribution function in Eq. 6.24. Obviously, the mapping functions introduce additional computational overhead for the path space mutations. To alleviate this issue we maintain the mapped state $P^{-1}(\bar{u})$ as a cache as well as the state $\bar{u}$ in the process of the mutations.

When going from path space to primary sample space, we need to decide on a technique t. The choice does not affect the correctness of the algorithm. It would, for instance, be possible to importance sample t based on the MIS weight of a given path, i.e., the acceptance probability of the tentative sample in the Markov chain. In our current implementation, we always initialize the Markov chain with a multiplexed primary sample space state, such that it comes with a valid t. If we choose a path space mutation, we will leave the technique t and the path length untouched, such that t is still valid when going back to multiplexed primary sample space.

6.6 Results

Setup We implemented MMLT [33], MLT [114], and the proposed algorithm in the same rendering system. For mutation techniques, we implemented small and large step mutations in the primary sample space, and the bidirectional mutations, lens/caustic/multi-chain perturbations, and manifold exploration [52] as the path space mutations. The reference images are rendered using BDPT or vertex connection and merging [26]/unified path sampling [36] with more than six hours of computation for each scene. We conducted all the experiments on a machine with

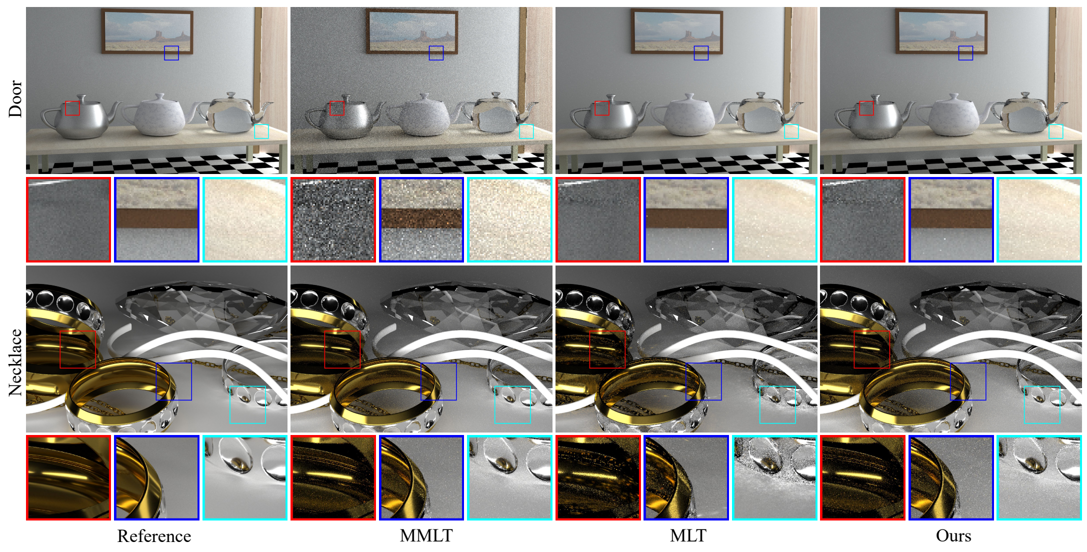


Figure 6.5: Equal-time rendering (20 minutes) of two scenes. The door scene contains glossy and specular materials and is illuminated by indirect lights with difficult visibility. This scene can be effectively rendered with MLT [114], but exhibits sub-optimal performance with multiplexed MLT (MMLT) [33]. On the other hand, the necklace scene, which is characterized by glossy interreflections, shows the opposite behavior. Our framework makes it possible to combine mutation strategies using the two different state spaces of MLT and MMLT, enabling the combination of specialized mutation strategies, and resulting in a general algorithm that works robustly in many cases.

an Intel Xeon E5-2698 v3 at 2.3 GHz using 32 threads.

The comparisons are all equal-time (20 minutes) with the maximum path length between 9–20 depending on the scenes. We manually tuned parameters for mutation techniques to achieve the best results for MMLT and MLT. We then used the same set of best parameters for our methods. The code for our proposed method will be available on our website.

Experiments Fig. 7.7 compares rendered images by MMLT, MLT, and our combination of both. We rendered two different scenes with different characteristics to highlight the need for fusing the state spaces. The *door* scene contains diffuse, specular, and glossy surfaces indirectly illuminated by a light leaking through the opening door. MLT handles this scene well because it can partially regenerate a path depending on the selection of the mutation strategies. The performance of MMLT is worse for this scene because MMLT needs to re-trace the complete path for every mutation. The *necklace* scene consists mainly of glossy surfaces illuminated by light sources of varying sizes causing complex interreflection between glossy surfaces. MMLT is effective for this scene because the target distribution includes the MIS weights. On the other hand, MLT tends to get stuck in local subspaces and generates non-uniform artifacts in the rendered image. In both cases, the result shows that our

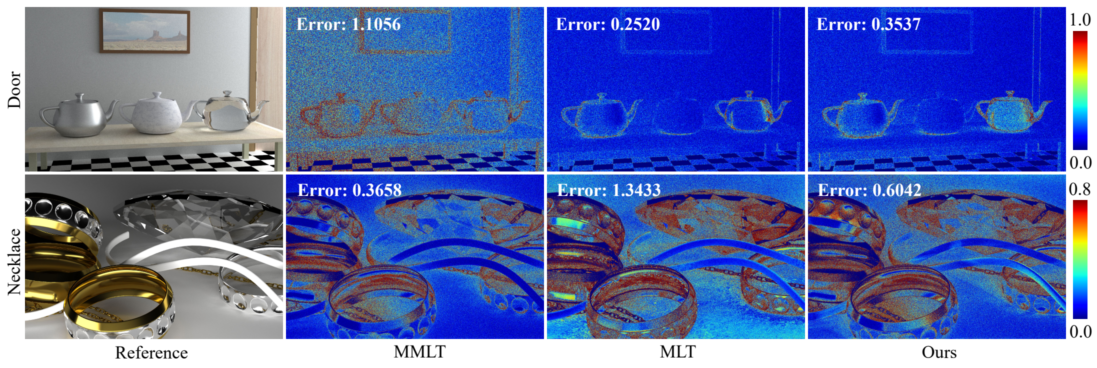


Figure 6.6: Comparisons of the errors for the scenes in Fig. 7.7. We show the relative RMSE for each images, and also show the pixel-wise relative error. We can observe that the combined mutation techniques with our method exhibit the error distribution similar to the better one of MMLT or MLT. Although the errors are suboptimal compared to the better one for the respective situation, this figure indicates our proposed method can moderately alleviate trade-offs between the two techniques, also in terms of the errors.

method can alleviate this robustness issue by combining mutation techniques from the two different state spaces.

We show two more scenes in Figs. 6.7 and 6.8. The *Salle de bain* scene contains diffuse and glossy surfaces illuminated with an area light source. Specifically the scene contains mirror models with low glossiness. This scene successfully captures the trade-off between MMLT and MLT in a single image. MMLT can render the part of the image visible from the mirror model more efficiently than MLT. On the other hand, MLT can render the part containing diffuse surfaces better than MMLT. The *Grey and White Room* scene consists of diffuse and glossy surfaces illuminated with several area light sources. This scene shows similar characteristic as the *door* scene and MLT outperforms MMLT. Again in this scene, our method can render the image with similar performance as MLT, the better one.

Error Analysis Fig. 6.6 shows the pixel-wise error distributions for the images in Fig. 7.7 compared to the references. We also show the error images for the scenes in Figs. 6.7 and 6.8. We used relative root mean square error (rRMSE) as an error metric. Our method exhibits similar error distributions as the better one in both scenes. In both cases, however, rRMSE values are larger than the respective better method, and some regions of the images display higher error than the better one.

We want to stress that achieving the best result among all is not our claim. The practical benefit is the generality to provide good performance for a wide spectrum of scenes, namely improved *robustness*, which is possible only by combining various mutation techniques. This context is different from multiple importance sampling which aims at an optimal combination of different sampling techniques. While

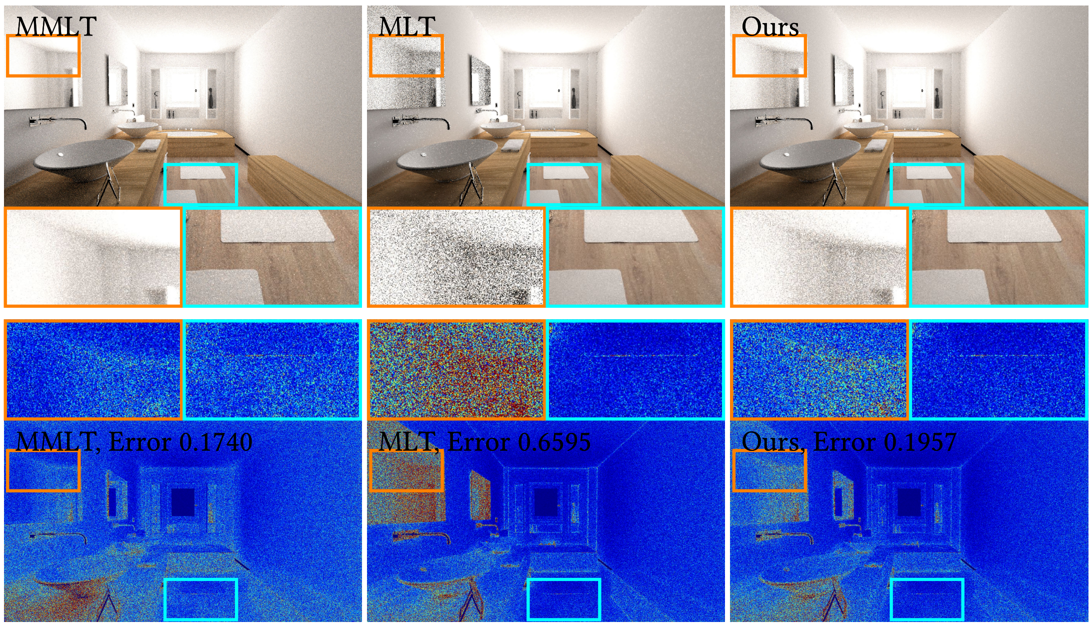


Figure 6.7: The *Salle de bain* scene (equal time comparison 20 min). A set of mutation strategies employed in MLT has problems with the mirror (orange inset) while MMLT is more noisy on the floor. Our combination alleviates both issues. Also note the reduced clumping of samples in the edges of the room as compared to MLT and the reduced noise in front of the first washing basin as compared to MMLT.

it would be ideal if we could select an optimal mutation strategy (or sets thereof) and the optimal space for a given scene, a combination of mutation strategies in different spaces has been just *impossible* to begin with, prior to our work. The optimal combination of different mutation strategies also remains as an interesting direction of the future work.

6.7 Discussion and Limitations

6.7.1 Discussion

Computational cost When switching from one space to another, our algorithm requires extra computational cost to convert a sequence of numbers in the primary sample space to a path, and then convert a path back to a sequence of numbers. This computational cost is, however, negligible compared to the entire render time. For instance, the conversion takes only 0.03% of the rendering time for the *necklace* scene in Fig. 7.7. The expected number of required transitions was 0.131 per single mutation for this scene. In addition, as was discussed in Section 6.5, it is possible to cache some of these computations.

Combination with other state spaces We focused on combining the primary sample space and the path space as they are major state spaces used in MCMC ren-

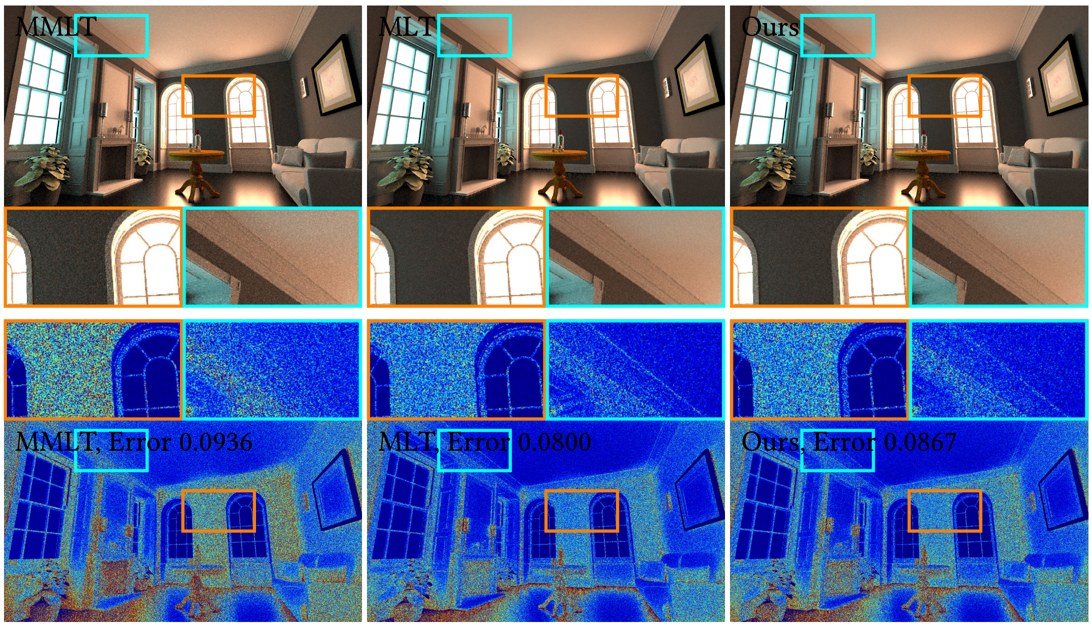


Figure 6.8: The *Grey and White Room* scene (equal time comparison 20 min). The scene has similar noise characteristics as the *door* scene and MLT generates better result than MMLT. Similar to the *door* scene, our method inherits the noise characteristics from the better method.

dering. The half-vector space [60] is another state space that has its own advantages over the path space. While we have not implemented their techniques in the rendering system we used for this paper, it should be straightforward to combine them by a composition of the transformations between domains (e.g., half vector space to path space to primary sample space, and vice versa).

Non-bijective sampling procedures Our solution of handling deterministic cases generalizes to more cases where the primary sample space formulation consumes more numbers than the resulting dimensions in the path space. A concurrent work [87] provides a mathematical formulation of a similar process. There is also an interesting connection to using expected values in the measurement contribution when evaluating the acceptance probability [64], which might lead to a more efficient approach for handling trans-dimensional moves in general.

Numerical precision When working in the primary sample space, extra care should be taken to avoid problems with numerical precision. For example, when mapping an outgoing direction to random numbers for BRDF sampling, numerical errors can accumulate and introduce bias. It is essential to work in double precision and add numerical guards against error accumulation and drifting. We implemented two techniques to assess numerical precision.

Firstly, during initial path sampling, we use only the valid samples based on the *round-trip values*. Given the candidate of the initial state u, we compute $\bar{x} = P_t^{-1}(\bar{u})$

and the round-tripped initial state $P_t(\bar{x})$. We decided not to use $\bar{u}$ as an initial state if $\tilde{C}(P_t(\bar{x}))$ has zero contribution. This test in the initialization phase can introduce MCMC start-up bias, which will vanish while converging as the Markov chain runs for longer. Without this test, the chain might get stuck in the same state, depending on the combination of the mutation strategies. Second, in the case of path space mutation, we also check the round-trip values for the mutated path $\bar{x}$. We check if $C(P_t^{-1}(P_t(\bar{x})))$ is non-zero, and the proposal is rejected immediately if the condition is not met.

This verification process can be optimized because $P_t^{-1}(P_t(\bar{x}))$ can be reused if the next mutation is a path space mutation. Similarly, this early rejection in the MH update can introduce bias.

6.7.2 Limitation

One limitation of our formulation is the requirement of an analytical mapping between a path to a sequence of numbers. This requirement is particularly violated for rejection sampling such as Woodcock tracking. In this case, we do not have an analytical form of (inverse) cumulative distribution functions since rejection sampling does not rely on the existence of such analytical solutions. It is still unclear how to handle cases where an analytical form of such a mapping is not easily accessible.

6.8 Summary

We proposed a framework to fuse the path space and the primary sample space of MCMC rendering for the first time. We explained how to formulate the connection between these two state spaces by introducing a novel mapping from a path to a sequence of numbers. This mapping is the inverse of existing path samplers and we thus named them as inverse path samplers. We show how such an inverse mapping can be formulated with cumulative distribution functions of samples which are oftentimes readily available as we already use inverse cumulative distribution functions for path samplers. Inverse path samplers allow us to use mutations designed only for one state space in another state space without modifying the mutation algorithms themselves. The results demonstrate that the fusion of two state spaces brings a practical benefit of robustness to different scene configurations, even when the use of one state space alone fails. Our framework should be immediately useful for existing MCMC rendering systems since it essentially introduces new mutation strategies in each state space without modifying each implementation. We believe that our framework leads to a new family of MCMC rendering methods since it provides a well-defined connection between two state spaces.

Chapter 7

Bridging Chromatic Spaces For Spectral Reflectance Reconstruction

7.1 Introduction

Spectral rendering simulates light transport considering spectral radiance per each wavelength. The material description in spectral rendering thus needs to be a full spectrum representation instead of a tristimulus representation (e.g., RGB colors). For example, reflectance becomes spectral reflectance defined over all (visible) wavelengths.

This difference causes two major inconvenient aspects for using spectral rendering in practice. Firstly, a full spectrum representation is more expensive in terms of memory footprint than tristimulus values. Unlike tristimulus colors, spectral values are usually needed at more than three wavelengths. Secondly, obtaining spectral data can be difficult and time consuming. Commonly available measuring devices and image editing systems often support only tristimulus values. It is thus more practical to use tristimulus values as inputs of a spectral rendering system.

This requirement leads to a problem of obtaining a spectrum given a tristimulus color. The main difficulty is that a tristimulus color can correspond to several spectra. This concept is known as *metamerism* in color science and such spectra are called *metamers*. Due to the metamerism, converting a tristimulus color to a spectrum is an ill-posed problem.

Smits [103] showed that smoothness of reconstructed spectra can be used to resolve the metamerism. This heuristic is based on the observation that measured spectra tend to be smooth [72]. A recent work [76] also uses the same heuristic. Reconstructed spectra with this heuristic, however, have little connection with measured spectra except for being smooth. The difference will show up as visible color shift in spectral rendering due to the presence of interreflections. Iterative multiplications of spectral distributions via interreflections can cause color shift even if reconstructed spectra have the same corresponding tristimulus colors (Fig. 7.7).

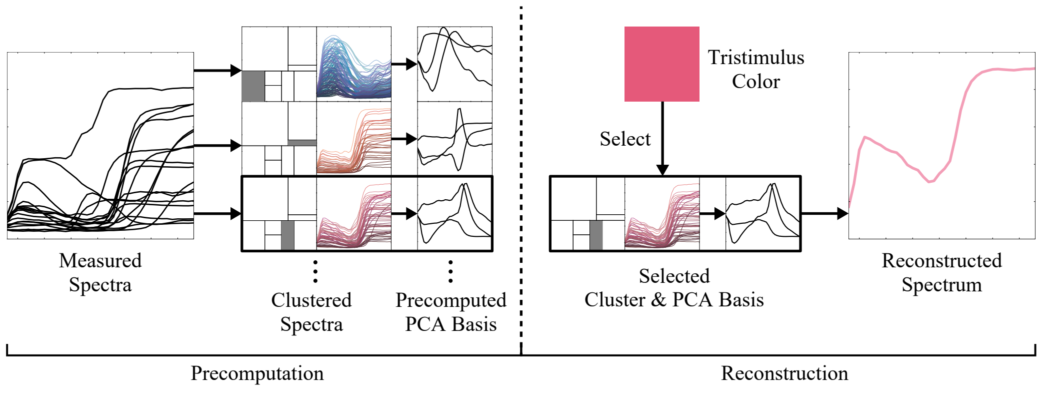


Figure 7.1: Overview of our method. The algorithm is separated into two phases: the precomputation phase and the runtime phase. In the precomputation phase, measured spectra are separated into several clusters. For each set of clustered spectra, we apply PCA to obtain the basis functions. In the runtime phase, an input tristimulus color is used to determine the corresponding cluster computed in the precomputation phase. Using the basis functions associated with the cluster, we reconstruct a spectrum.

We propose a data-driven approach to reproduce the measured spectra based only on tristimulus colors. The key observation is that principal component analysis (PCA) reveals that typical measured spectra can be accurately reproduced with only a few principle components [15, 23]. Given measured spectra that satisfy this property, we can reformulate conversion from tristimulus colors to spectra as a well-defined matrix equation by considering the first three components. Unlike the previous methods in computer graphics, our approach does not assume that spectra are smooth. Since we use measured spectra in the conversion process, we are able to reconstruct spectra that match well with actual measured spectra. Unlike similar work in color science [6, 37], we use greedy clustering of spectra to directly minimize the difference between the reconstructed spectra and the measured spectra. The runtime algorithm is as simple as one matrix multiplication with an input tristimulus color, which is suitable for spectral rendering. In summary, our contributions are:

- Reformulation of the relationship between tristimulus colors and spectra based on PCA.
- Greedy clustering method which numerically minimizes the reconstruction error.
- Simple runtime algorithm which converts tristimulus colors into spectral reflectances.

7.2 Related Work

Reconstruction of a spectrum from various information is a major topic in color science. The problem setting varies according to the information available for reconstruction. We focus on related work only on reconstruction from tristimulus colors.

Optimization Approaches As presented in the pioneering work by Smits [103], conversion from a tristimulus color to a spectrum can be formulated as a numerical optimization problem. This formulation tries to find a spectrum that optimizes a certain cost function such as the difference between converted tristimulus values and input tristimulus values. Smits formulated this problem as a linear optimization of discretized spectra. This method tries to find the smoothest possible spectrum that retains the input tristimulus color which is based on the observation that natural spectra are often smooth [72]. Dupont [22] tested various optimization methods for finding a metameric spectrum similar to the method proposed by Smits, and concluded that the Hawkyard method [45] is among the best.

One issue of this approach is that we need to perform a heavy optimization process for a given input tristimulus color, which is too costly for certain applications. Smits [103] thus proposed to use precomputed spectra only on some primal colors and to rely on linear interpolation for other colors at runtime. Meng et al. [76] later reported that we in fact need more precomputed samples to cover the XYZ tristimulus color space as input. Our method is free from such interpolation and intrinsically supports inputs from the XYZ color space. Another issue is that reconstructed spectra with this approach can be very different from measured spectra, if the smoothness heuristic does not hold. We avoid using smoothness heuristics via a data-driven approach.

Data-driven Approaches Since measured spectra data are publicly available, one can also analyze those data and reformulate the conversion process as a problem of finding the relationship between tristimulus colors and measured spectra. Such approaches usually express spectra as a weighted sum of basis functions, especially based on PCA. Cohen [15] is the first to perform PCA on measured spectra, and he found that spectral reflectances can be accurately represented by a small number of basis functions. Since then, a weighted sum of basis functions is widely used as a compact representation of measured spectra [88, 23, 109] for the purpose of compressing measured data.

Building upon this idea, if one can find a well-defined relationship between tristimulus colors and the weights of basis functions, we are able to reconstruct spectra from tristimulus colors. While a direct application of PCA for this purpose is ap-

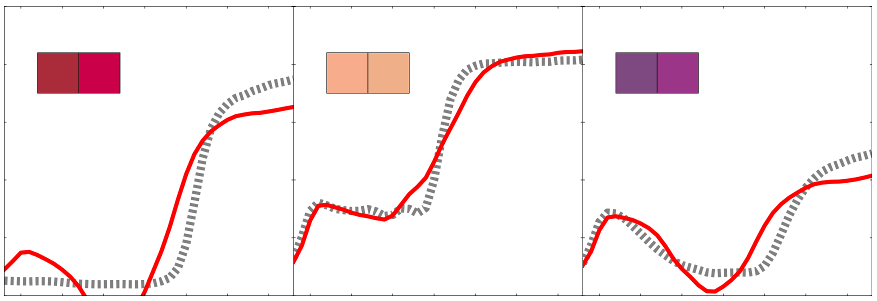


Figure 7.2: Approximation (red) of a measured spectrum (dotted) via direct PCA. Although the approximated spectrum is generally similar to the original spectrum, the corresponding colors (left: original, right: reconstructed) are visually different.

pealing, research in color science concluded that it does not work well. Marimont et al. [73] thus proposed to use a different set of basis functions including the effect of sensor responses by changing the optimization process in PCA. Other researchers suggested to use variants of PCA such as weighted PCA [2] and adaptive NMF [3]. Our method is also built upon PCA, but we propose to cluster spectra in the input color space before performing PCA. Unlike similar work in color science [6, 126], our clustering algorithm directly minimizes reconstruction errors at runtime. As far as we know, we are the first to take this data-driven approach in computer graphics.

Abed et al. [1] proposed to use a set of measured spectra directly and interpolate them in the input color space for reconstruction. Some other methods suggest to use non-linear mapping between a tristimulus color and a spectrum such as a feed-forward neural network [97], a radial basis function network [84], and nonlinear PCA with an auto-associative neural network [7]. While these methods improve the accuracy of reconstruction, the tristimulus colors of the reconstructed spectra are not guaranteed to match with the input tristimulus color (called *round-trip error*). Our formulation theoretically guarantees zero round-trip errors (Eq. 7.9), which is important for applications in computer graphics.

Representation of Spectral Distributions In addition to discretized samples over wavelengths, one could represent a spectral distribution as a set of basis functions such as polynomials [91, 106], Fourier basis [90], Gaussian quadrature [78], or wavelets [14]. Since our goal is not to find a representation of spectral distributions, but to generate spectral distributions themselves, these works are orthogonal to ours.

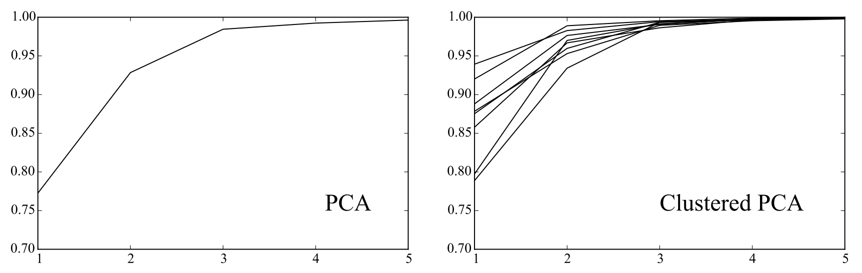


Figure 7.3: Accumulated contribution with respect to the number of components for direct PCA applied to the entire dataset (left) and the per-cluster contributions (right) with our clustered PCA. These plots essentially show how accurately the entire dataset is reproduced for each number of components.

7.3 Overview

Our goal is to reconstruct a spectral reflectance from a tristimulus color. We determined the following design criteria for this goal.

1. *Reproduction of spectra*: The reconstructed spectra should match the measured spectra as well as possible.
2. *Recovery of tristimulus colors*: The reconstructed spectrum should represent the input color.
3. *Fast reconstruction*: The runtime computational cost of reconstruction should not be too high.

Note that achieving the criterion #1 does not necessarily mean satisfying the criterion #2 automatically. For example, Fig. 7.2 shows L2-error minimized reconstruction of spectra with three bases by PCA, but their resulting tristimulus colors are generally different from the corresponding input colors. We thus need to account for these points separately.

Fig. 7.1 shows the overview of our method. In the precomputation phase, we first cluster a set of measured spectra into several disjoint subsets. The clustering algorithm is designed to minimize the difference between the reconstructed spectra and the measured spectra (criterion #1). For each cluster, we precompute a set of basis functions based on PCA over the measured spectra within the cluster. A direct application of PCA to our problem, however, is impossible since we do not know the original measured spectra at runtime to compute weights for the basis functions. We thus introduce a practical approximation to calculate those weights only from input tristimulus colors. In order to alleviate additional error introduced by this approximation, we also propose to use a clustered PCA with greedy minimization of reconstruction error.

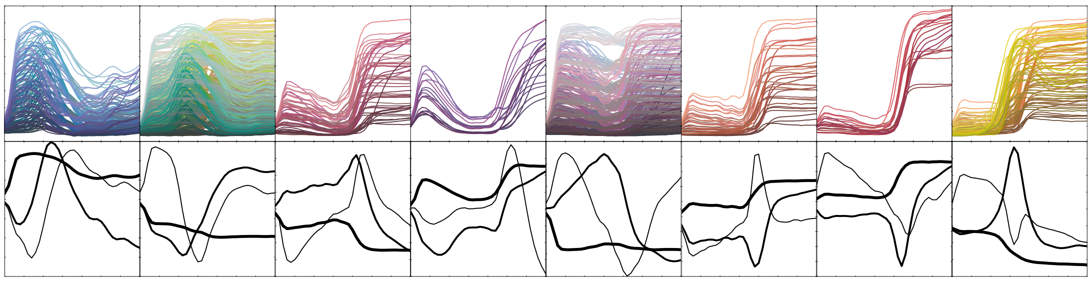


Figure 7.4: Measured spectra plotted with the corresponding colors (top) and the corresponding PCA basis functions (bottom) for each cluster. The thick, normal, and thin lines show the first, second, and third principal components respectively. Our clustering method can group spectra with similar shapes and the basis functions are indeed very different for each cluster.

In the reconstruction phase, we first select the corresponding cluster based on the given tristimulus color. We then reconstruct a spectrum as a weighted sum of the precomputed basis functions associated with the selected cluster. This formulation guarantees the recovery of the input tristimulus color after reconstruction (criterion #2). The conversion process is just a matrix-vector multiplication, which is fast enough for many applications (criterion #3).

7.4 Method

7.4.1 Tristimulus Colors

A color visible to the human eye can be represented by a spectrum $S(\lambda)$ defined for the wavelength λ. We use the range $\Lambda = [380, 730]$ nm in the following. In order to describe various aspects of the human vision system, many color spaces have been developed. Among them, the CIE XYZ color space is often used as the reference color space among other color spaces. The conversion process from a spectrum $S(\lambda)$ to tristimulus values $\mathbf{c} = [X\ Y\ Z]^T$ is

$$X = \int_\Lambda S(\lambda)\bar{x}(\lambda)\mathrm{d}\lambda \quad Y = \int_\Lambda S(\lambda)\bar{y}(\lambda)\mathrm{d}\lambda \quad Z = \int_\Lambda S(\lambda)\bar{z}(\lambda)\mathrm{d}\lambda, \tag{7.1}$$

where $\bar{x}$, $\bar{y}$, and $\bar{z}$ are the color matching functions that describe chromatic responses of the standard observer. While our framework is not limited to this choice, we used the analytical approximation of the color matching functions by Wyman et al. [124] for the CIE 1931 standard observer. The RGB color space we used is the sRGB color space (Rec.709) [110].

7.4.2 Representation of Measured Spectra

We represent a spectrum as a weighted sum of a fixed set of basis functions via PCA. Similar to previous work, we first discretize a spectrum $S(\lambda)$ into n fixed-width bins. The width of each bin in our experiments is $\Delta = 10\,\text{nm}$ ($n = 36$). We denote this discretized spectrum as an n dimensional vector $\mathbf{s} \in \mathbb{R}^n$. Running PCA over a set of measured spectral reflectance data gives us a set of basis functions $\{\mathbf{b}_j\}_{j=1,\ldots,n}$. Each measured spectrum under these new basis functions then becomes

$$\mathbf{s} = \sum_{j=1}^{n} w_j\,\mathbf{b}_j, \tag{7.2}$$

where $w_j = \mathbf{b}_j \cdot \mathbf{s}$ is the weight for the jth basis function. Research in color science shows that each measured spectrum can be well approximated by a few basis functions [15, 72, 23].

This observation is confirmed by the accumulated contribution in Fig. 7.3. In PCA, the accumulated contribution is utilized to assess how much of the information is kept with a given number of components. This value for k-th component is defined as a cummulative sum $\sum_{i=1}^{k} r_i$ of the explained variance ratio $r_i = \sigma_i / \sum_{j=1}^{n} \sigma_j$ where σ_j is the variance of the data around j-th principal component. The accumulated contributions quickly approach to one only with a few basis functions regardless of clustering. We thus use only three basis functions in the following.

This choice also means that our method is not applicable to the cases where measured spectra cannot be accurately reconstructed just by three bases. While reproduction of such general measured spectra from tristimulus colors remains challenging, being a data-driven approach, our method allows us to take advantage of measured spectra unlike the previous approaches in computer graphics [103, 76].

By truncating the sum to three components, we obtain a compact representation of a spectrum $\mathbf{s}$ via three weights w_1, w_2, w_3 using the common three basis functions $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ as

$$\mathbf{s} \approx w_1\mathbf{b}_1 + w_2\mathbf{b}_2 + w_3\mathbf{b}_3. \tag{7.3}$$

This direct approximation, however, may not recover the input tristimulus color. In other words, the approximated spectrum and the original spectrum $\mathbf{s}$ may not be metameric spectra. Fig. 7.2 shows an example of this problem: while the approximation is similar to the original spectrum, it represents a different tristimulus color.

This property is not suitable for rendering because we eventually convert the obtained spectrum to tristimulus colors to display final images and the difference might be visually noticeable. We introduce a novel formulation that achieves a compact presentation and the recovery of the input color at the same time. As such, our work is not about merely having a compact representation of measured spectra for given tristimulus colors.

7.4.3 Reconstruction of Spectra via PCA

We can circumvent the above issue by considering a full conversion process between spectra and tristimulus colors. We start with the relationship between **c** and **s** as in Eq. 7.1. By discretizing the color matching functions as n dimensional vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$, we have

$$X \approx \Delta\left(\mathbf{x} \cdot \mathbf{s}\right) \quad Y \approx \Delta\left(\mathbf{y} \cdot \mathbf{s}\right) \quad Z \approx \Delta\left(\mathbf{z} \cdot \mathbf{s}\right). \tag{7.4}$$

Instead of representing a spectrum directly with basis functions as Eq. 7.3, we approximate $\mathbf{s} - \mu$ with basis functions where μ is the average of the measured spectra:

$$\mathbf{s} \approx \tilde{\mathbf{s}} = w_1\mathbf{b}_1 + w_2\mathbf{b}_2 + w_3\mathbf{b}_3 + \mu. \tag{7.5}$$

This modification is often adopted in PCA [79], and also used in color science [57, 11, 122]. Therefore we can approximate the corresponding tristimulus color **c** as

$$\mathbf{c} \approx \Delta \begin{bmatrix} \mathbf{x}^T \\ \mathbf{y}^T \\ \mathbf{z}^T \end{bmatrix} \tilde{\mathbf{s}} = \underbrace{\Delta \begin{bmatrix} \mathbf{x}^T \\ \mathbf{y}^T \\ \mathbf{z}^T \end{bmatrix} \begin{bmatrix} \mathbf{b}_1\mathbf{b}_2\mathbf{b}_3 \end{bmatrix}}_{M} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + \mathbf{c}_\mu = M \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} + \mathbf{c}_\mu, \tag{7.6}$$

where $\mathbf{c}_\mu = \Delta \begin{bmatrix} \mathbf{x}\,\mathbf{y}\,\mathbf{z} \end{bmatrix}^T \mu$. The matrix M is a 3×3 matrix where each element is defined by a dot product of a basis function and a color matching function. The matrix is a constant given the basis functions.

While each weight is given by $w_j = \mathbf{b}_j \cdot (\mathbf{s} - \mu)$, it is impossible to obtain weights using this definition since the input is **c**, not **s**. We thus use the (pseudo) inverse of M to approximate the weights as

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \approx M^{-1}(\mathbf{c} - \mathbf{c}_\mu) = \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \hat{w}_3 \end{bmatrix} \tag{7.7}$$

and obtain the reconstructed spectrum **r** as

$$\mathbf{s} \approx \tilde{\mathbf{s}} \approx \mathbf{r} = \hat{w}_1\mathbf{b}_1 + \hat{w}_2\mathbf{b}_2 + \hat{w}_3\mathbf{b}_3 + \mu. \tag{7.8}$$

Remind that an approximated spectrum $\tilde{\mathbf{s}}$ via PCA may not recover the input tristimulus color **c**. The reconstructed spectrum **r**, however, converts back to **c** by con-

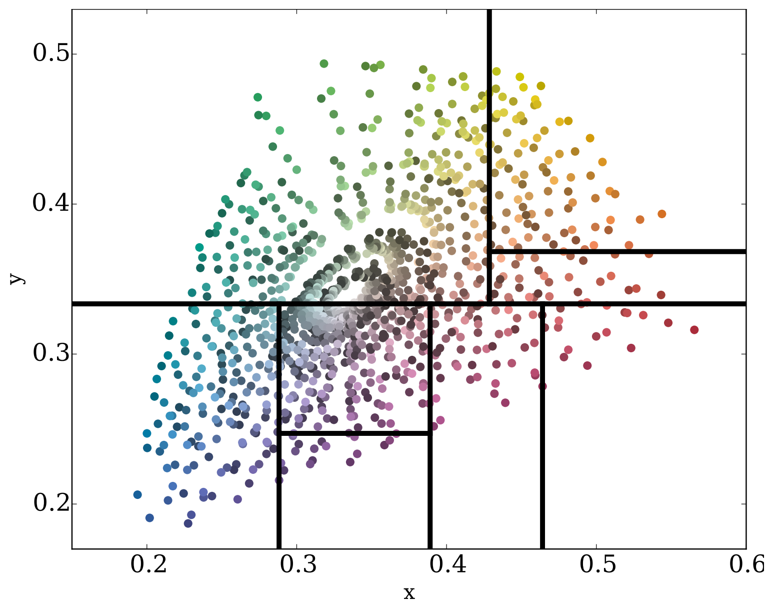


Figure 7.5: Visualization of the precomputed clusters and measured spectra in the xy-plane. The color of each point is the converted RGB color. The subdivision is done for the entire xy plane, but we zoomed into the region where the measured spectra exist.

struction since we have

$$\Delta \begin{bmatrix} \mathbf{x}^T \\ \mathbf{y}^T \\ \mathbf{z}^T \end{bmatrix} \mathbf{r} = \underbrace{\Delta \begin{bmatrix} \mathbf{x}^T \\ \mathbf{y}^T \\ \mathbf{z}^T \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \mathbf{b}_2 \mathbf{b}_3 \end{bmatrix}}_{M} \begin{bmatrix} \hat{w}_1 \\ \hat{w}_2 \\ \hat{w}_3 \end{bmatrix} + \mathbf{c}_\mu = M \underbrace{M^{-1}(\mathbf{c} - \mathbf{c}_\mu)}_{\left[\hat{w}_1 \hat{w}_2 \hat{w}_3\right]^T} + \mathbf{c}_\mu = \mathbf{c}. \tag{7.9}$$

Therefore the reconstructed spectrum $\mathbf{r}$ and the original spectrum $\mathbf{s}$ are metamers by construction in this formulation. Since we can precompute $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3, M$, and M^{-1}, the runtime conversion is just one matrix vector multiplication $M^{-1}(\mathbf{c} - \mathbf{c}_\mu)$ as in Eq. 7.7. This formulation of conversion from tristimulus colors to spectra no longer relies on smoothness heuristics.

7.4.4 Hierarchical Chromatic Clustering

While the reconstruction approach above works, Eq. 7.8 reveals that this approach involves two different approximations: $\mathbf{s} \approx \tilde{\mathbf{s}}$ and $\tilde{\mathbf{s}} \approx \mathbf{r}$. Even though PCA minimizes the L2 error $\|\mathbf{s} - \tilde{\mathbf{s}}\|^2$, due to another approximation $\tilde{\mathbf{s}} \approx \mathbf{r}$, the reconstructed spectrum $\mathbf{r}$ does not necessarily well approximate the original spectrum $\mathbf{s}$.

We address this issue by directly minimizing the L2 reconstruction error $\|\mathbf{s} - \mathbf{r}\|^2$. Our idea is inspired by that natural spectra with similar colors tend to be similar [25]. Related work in color science [6, 126] suggest that similar approximation via PCA on such spectra indeed improves the reconstruction accuracy, although they did not fa-

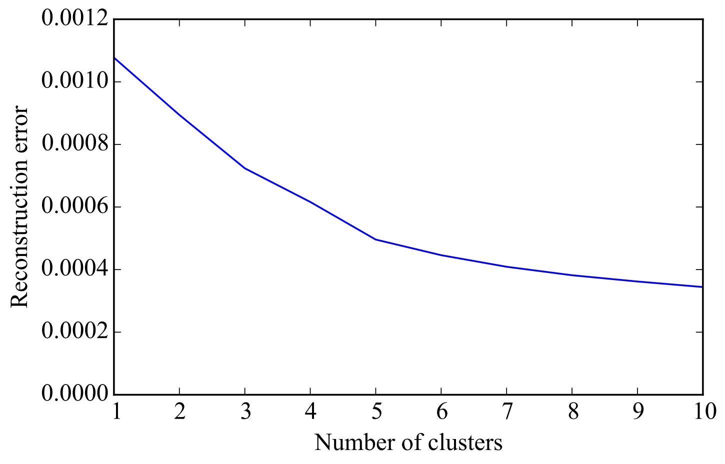


Figure 7.6: Relashionship between the number of clusters and the reconstruction error defined in Eq. 7.13. We can observe that the reconstruction error decreases as the number of clusters increases, eventually becomes flat. Based on this observation, we empirically fixed the number of clusters to eight.

cilitate the clustering scheme to directly minimize the reconstruction error. Building upon this idea, we cluster the spectra in the xy plane of the CIE xyY color space such that the resulting L2 reconstruction error $\|\mathbf{s} - \mathbf{r}\|^2$ over all the spectra is numerically minimized.

To be concrete, we iteratively split the measured spectra to construct a kD-tree on the xy plane. Each split tries to minimize the L2 reconstruction error $\|\mathbf{s} - \mathbf{r}\|^2$ in a greedy manner. This procedure is inspired by a top-down kD-tree construction with SAH [44]; our approach tries to minimize the L2 reconstruction error, while the top-down kD-tree construction tries to minimize SAH.

Splitting Spectra We explain the process of splitting a set of spectra $S \equiv \{\mathbf{s} \in \mathbb{R}^n\}$ into two subsets S_1 and S_2. We determine the splitting plane perpendicular to each axis and compute the splitting position so that the sum of the L2 reconstruction error for each subset is minimized. Given the splitting axis $p \in \{\mathrm{x}, \mathrm{y}\}$ and the splitting position $v \in [0, 1]$, the subsets S_1 and S_2 can be written as

$$S_1(p, v) = \{\mathbf{s} \in S \mid \mathbf{c}.p \geq v\}, \tag{7.10}$$

$$S_2(p, v) = S \setminus S_1(i, v), \tag{7.11}$$

where $\mathbf{c}.p$ returns x or y coordinates of the measured spectrum $\mathbf{s}$ in the xyY color space according to the axis p. Using this definition, the splitting axis and position is

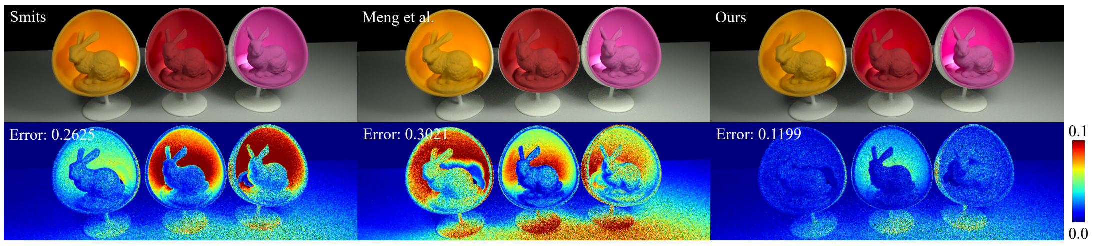


Figure 7.7: Rendering using reconstructed spectral reflectances. We rendered a scene with Lambertian surfaces illuminated by a black-body illuminant at 5500 K using a spectral rendering system. For spectral reflectances, we use the method by Smits [103], the method by Meng et al. [76], and our method to reconstruct them from tristimulus colors. Tristimulus colors are from the corresponding reference measured spectra [63]. The bottom row visualizes errors from the reference rendering using the measured spectra. The existing methods suffer from color shifts in indirect illumination due to the difference between the reconstructed spectra and the measured spectra. Our method faithfully reproduces measured spectra only from tristimulus colors, which has been considered challenging in computer graphics.

written as

$$(i^*, v^*) = \underset{i,v}{\operatorname{argmin}} \left(\Delta E(S_1(i, v)) + \Delta E(S_2(i, v)) \right). \tag{7.12}$$

$\Delta E(S)$ is the sum of the reconstruction error for a set of spectra S:

$$\Delta E(S) = \sum_{\mathbf{s} \in S} \|\mathbf{s} - \mathbf{r}\|^2 , \tag{7.13}$$

where $\mathbf{r}$ is obtained by using the approach described in the last section with PCA over the subset S. We note that the splitting axis and position is only determined by the optimization process, thus our approach does not depends on the starting condition.

At each leaf node, we store the basis functions, the conversion matrix, and the mean values of the corresponding subset of spectra. At runtime, we traverse the tree using the input tristimulus color and select the corresponding leaf node to reconstruct the spectrum. Since the reconstruction process still recovers input tristimulus colors, a spectrum and a color has a bijective mapping. A set of reconstructed spectra at each leaf node minimizes L2 errors as in Eq. 7.13.

7.5 Results

Setup We used the two datasets of the matte Munsell color chips [63]. The dataset I is a collection of 1269 measurements ranging from 380 nm to 800 nm with 1 nm interval. The dataset II is a collection of 1250 measurements ranging from 400 nm to

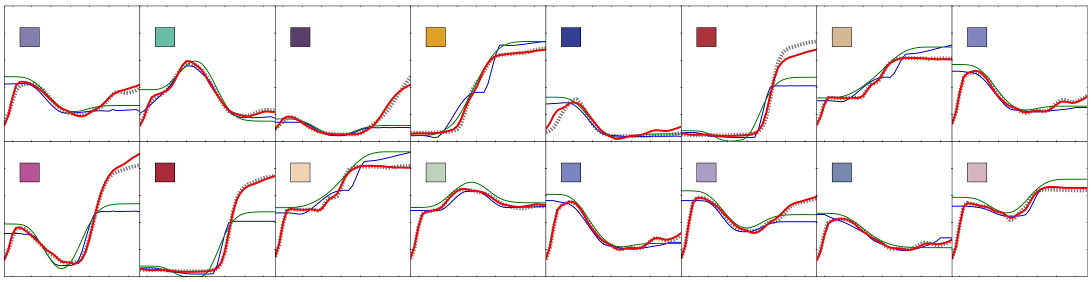


Figure 7.8: Comparison of the reference spectrum (dotted) and the reconstructed spectra for the dataset I with the method by Smits [103] (blue), the method by Meng et al. [76] (green), and the proposed approach (red) as well as the RGB color represented by the reference spectrum. All reconstructed spectra are converted from the tristimulus colors of the reference spectra.

700 nm with 5 nm interval. The precomputation and the evaluation are conducted respectively on the dataset I and II. The range of the vertical axis is $[0, 1]$ for all the plots in this section unless otherwise specified. The range of the horizontal axis is $[380, 730]$ nm or $[400, 700]$ nm respectively for the results using the dataset I or II. We integrated our method into a spectral rendering system based on bidirectional path tracing [66, 115]. All images are rendered with equal time (1 hour) on the same environment. We conducted all the experiments on a machine with an Intel Core i7-5960X at 3.0 GHz using 16 threads. Both the rendered images and visualized colors are in the sRGB color space under the D65 illuminant. The algorithm itself is independent of a choice of the color space and the illuminant. We plan to distribute an implementation of our method.

Clustering of Measured Spectra Fig. 7.4 shows the measured spectra in each cluster for the dataset I. Fig. 7.5 visualizes the distribution of measured spectra and the clusters in the xy-plane for the precomputation using the same dataset. We used eight clusters so that the amount of precomputed data is roughly equal to that of the method by Smits [103]. We also experimentally verified that the reconstruction error does not decrease much for the number of clusters more than eight (Fig. 7.6).

We can observe that our algorithm successfully clusters spectra that have similar shapes with different overall magnitudes. Ignoring the magnitudes is desirable in our case since the reconstruction via PCA will take care of overall scaling. Fig. 7.4 also shows the corresponding three PCA basis functions at the bottom. Different clusters have quite different basis functions, validating the use of clustered PCA instead of PCA over an entire dataset.

Reconstructed Spectra Fig. 7.8 and Fig. 7.9 compare several measured spectra [63] and reconstructed spectra by three different methods; the method by Smits [103]

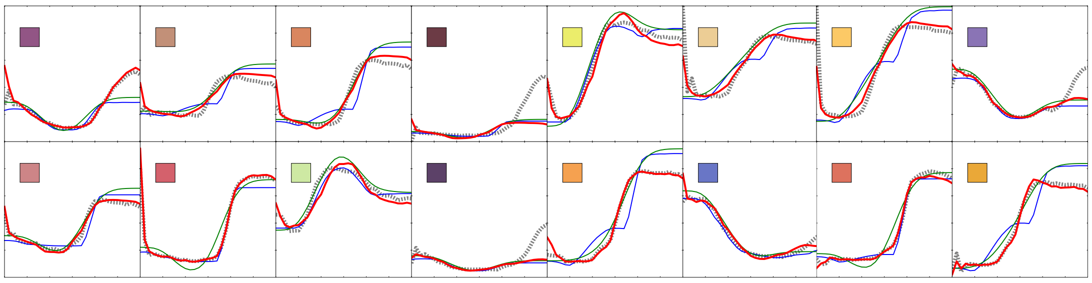


Figure 7.9: Comparison of the reference and reconstructed spectra for the dataset II in similar configuration to Fig. 7.8.

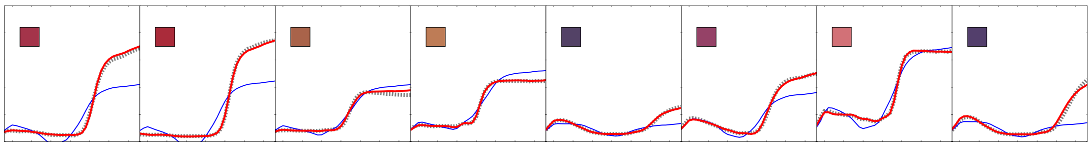


Figure 7.10: Comparison of the reconstructed spectra with (red) and without (blue) clustering for the dataset I. Similar to Fig. 7.8, we used the measured spectra for the reference (dotted), and reconstructed them. We can observe that clustering improves the accuracy of reconstruction, especially for longer wavelengths.

(blue line), the method by Meng et al. [76] (green line), and ours (red line). The reference RGB color is shown in the top left of each plot. We can observe that the reconstructed spectra with our method match well with the measured spectra in many cases. The two previous methods, while producing metameric spectra like ours, result in very different spectra. It is because real-world spectra are not simply characterized as smooth in general, and a data-driven approach like ours is necessary to capture more complex characteristics of real-world spectra. Our method is the first to achieve such results in computer graphics. Tab. 7.1 summarizes the reconstruction errors for each reconstruction method. Δe is the L2 reconstruction error for each reference spectrum $\|\mathbf{s} - \mathbf{r}\|^2$. Our method outperforms the other methods in terms of most metrics in the table.

We note that the purpose of our approach is to achieve a compact representation of the spectrum based on the specific data and the accurate recovery of the input color at the same time. In this sense, because we are not focus on the reconstruction of the general reflectance spectra, our approach can be considered as data compression rather than learning. In other words, we do not focus on applying the precomputed data to reconstruct the features in the different datasets. This explains why, in our experiments, we evaluated the reconstruction methods only with the precomputed data obtained from the same dataset, instead of applying the precomputed data obtained from the different dataset.

Fig. 7.10 highlights the importance of clustering, illustrating the reconstructed

Dataset	Method	Δe Mean	Δe Med.	Δe Max.
I	Smits	0.1904	0.0819	1.8439
	Meng et al.	0.1633	0.0710	1.1410
	PCA only	0.0388	0.0156	**0.9744**
	Ours	**0.0138**	**0.0053**	0.9875
II	Smits	0.1297	0.0633	1.1825
	Meng et al.	0.1566	0.0985	1.0305
	PCA only	0.0024	0.0014	**0.0214**
	Ours	**0.0022**	**0.0012**	0.0220

Table 7.1: Reconstruction errors of the four reconstruction methods: Smits [103], Meng et al. [76], PCA without clustering, and the proposed method. We show the mean, the median, and the maximum of the reconstruction errors for each dataset computed from the reference spectra. In almost all cases, our method is the most accurate.

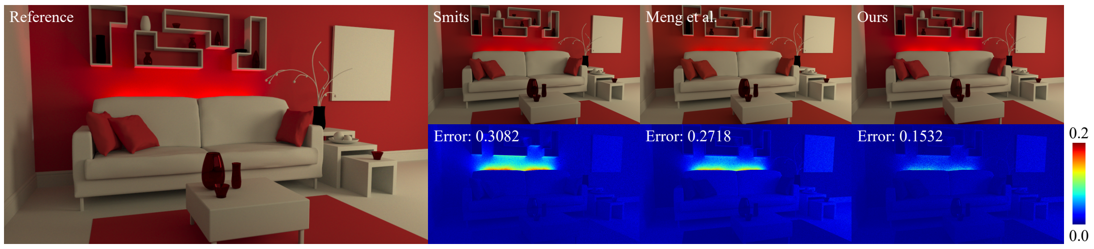


Figure 7.11: Rendering using reconstructed spectral reflectances with a less naive scene. The scene contains Lambertian surfaces and illuminated by black-body illuminants at 4000K and 5500K. Several light sources are placed behind the sofa. Similar to Fig. 7.7, we render the images with three different methods and visualize the error images as well as the corresponding error values, compared to the reference obtained with the measured spectra.

spectra with and without clustering for the dataset I. For the results without clustering, we just apply PCA over all the measured spectra and use the same set of basis functions for any input. While based on the same formulation via PCA (Eq. 7.8), clustering dramatically improves the accuracy of reconstruction. We found that similar approaches in color science [57, 11, 122, 79] are not significantly different from our PCA without clustering in terms of error since they do not directly minimize reconstruction error.

Rendering with Reconstructed Spectra We tested the importance of reproducing of measured spectra in spectral rendering. The scene shown in Fig. 7.7 is rendered with black-body illuminants of 5500 K. All the materials except for the light source are Lambertian with spectral reflectances. We also render a less naive scene in Fig. 7.11. The scene is illuminated by black-body illuminants of 4000 K and 5500 K.

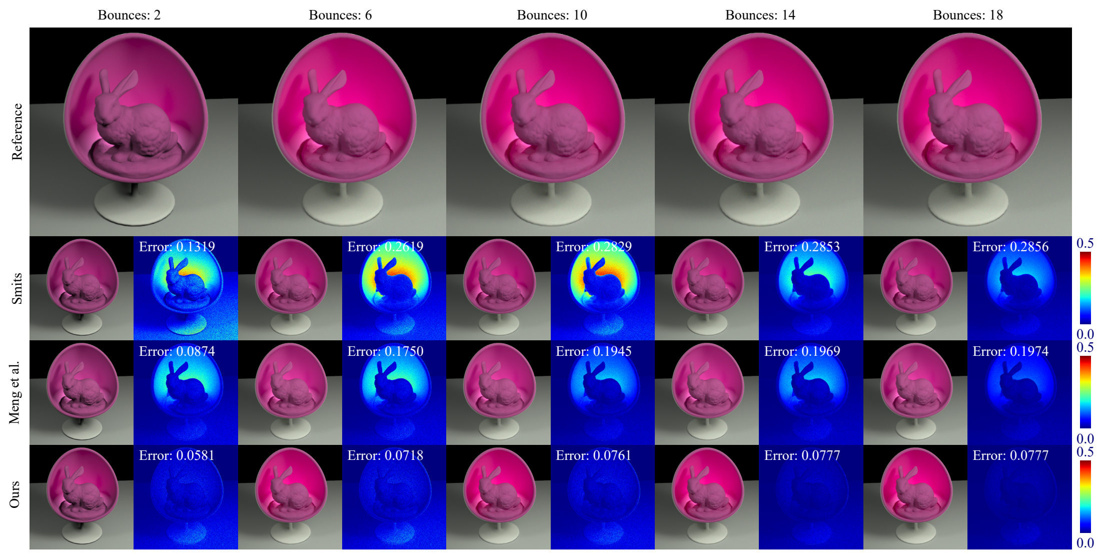


Figure 7.12: Rendering with different number of bounces using the spectra converted from the same tristimulus color with the three different methods: the method by Smits [103], the method by Meng et al. [76], and our method. As well as the rendered images we also show error images and the error values correspoinding to the images. We note that, because we use different references for each column, the error images and the error values are irrelevant with respect to the different number of path length, although we used the same error scale for the visualization.

Several light sources casts indirect lighting to the red wall. In both scenes, the top row shows the resulting images using spectral reflectances produced by the methods of Smits [103], Meng et al. [76], and ours. The bottom row visualizes per-pixel error from the reference rendering using the measured spectra. We also show the error values (rRMSE) correspoinding to the error images. The color map next to the error images is linear according to the error values because we compute the errors *before* tone mapping. We determine the range of the color map between zero and the threshold near the maximum error among the images. The tristimulus color used for reconstruction is obtained from the corresponding measured spectrum. We can observe that the final color in the rendered images with the previous reconstruction methods suffer from color shifts especially for the indirect illumination. This result implies that different metameric spectra can in fact lead to visible differences because the reflectances are multiplied by many times. This observation underlines the importance of accurate reproduction of measured spectra.

We also show rendered images with different number of bounces using the three different methods (Fig. 7.12). Similarly, we also show the error images obtained by the comparison with the references using the corresponding measured spectra for each path length. We can observe the similar tendency that our method can generate the better results.

Fig. 7.13 shows how the number of clusters affects the rendered images. The

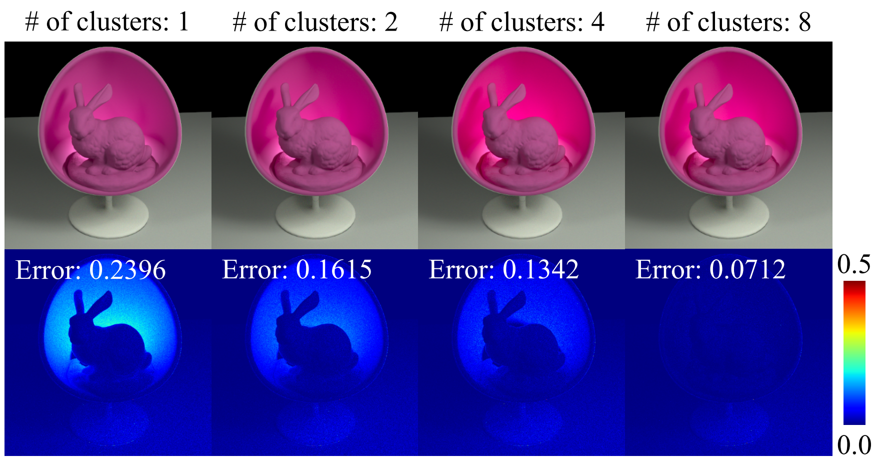


Figure 7.13: Rendered images using the reconstructed spectra from the multiple number of clusters, converted from the same tristimulus color. The top row shows the rendered images using the spectrum with the corresponding number of clusters. The bottom row shows the pixel-wised error compared to the reference rendered with the original spectrum corresponding to the tristimulus color, as well as the error values.

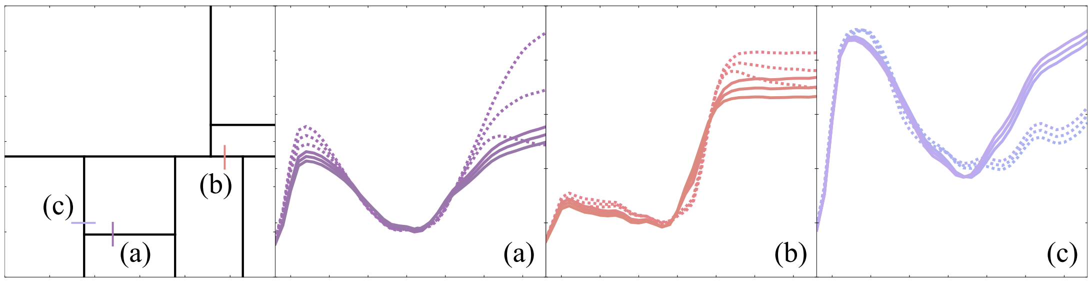


Figure 7.14: Some sets of spectra across cluster boundaries. The left-most plot is the xy plane with three different lines across the cluster boundaries. The right three figures are corresponding reconstructed spectra to the samples points on the lines. We assigned the different line types (solid, dashes) for the spectra in the different clusters.

bottom line shows the error compared to the reference obtained with the measured spectra corresponding to the input tristimulus color. Due to the inaccurate reconstruction, the rendered image with a small number of clusters suffers from the color shift compared to the reference. We can observe that the color shift can be alleviated by adopting sufficient number of clusters.

7.6 Limitations

Reconstruction in Cluster Boundaries Fig. 7.14 illustrates the change of the reconstructed spectra in the cluster boundaries. The left-most plot shows the xy plane with cluster boundaries. The sampled points on each of lines (a) - (c) correspond to the set of the spectra in the right three sets of spectra. Fig. 7.15 shows some rendered

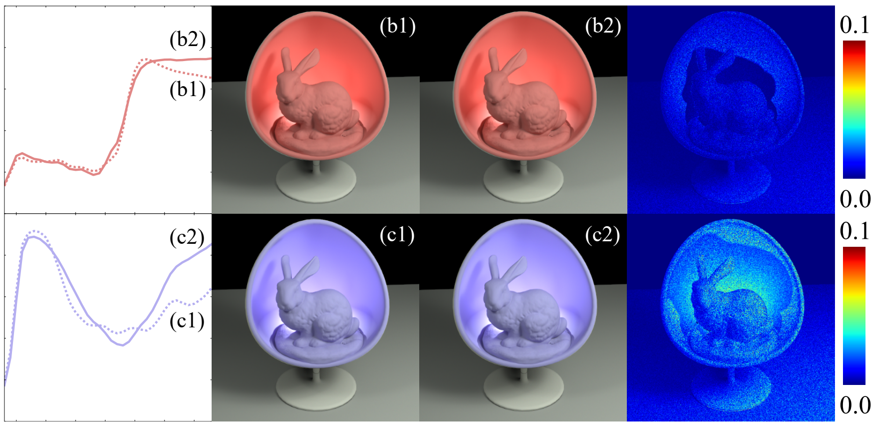


Figure 7.15: Rendered images using the reconstructed spectra with the two consecutive colors for the lines (b) and (c) in Fig. 7.15. We selected the consecutive colors so that the differences of x or y values are within 10^{-5}. We rendered the scene containing indirect illumination with the spectral reflectances using two reconstructed spectra from the two colors. The right-most images are the difference of two images.

Dataset	Method	Δe Mean	Δe Med.	Δe Max.
Glossy	Smits	0.0050	0.0023	0.0523
	Meng et al.	0.0061	0.0028	0.0484
	PCA only	0.0014	0.0006	**0.0441**
	Ours	**0.0006**	**0.0002**	0.0455
Natural Colors	Smits	0.0057	0.0036	0.0643
	Meng et al.	0.0054	0.0037	0.0602
	PCA only	0.0027	0.0017	0.0285
	Ours	**0.0021**	**0.0013**	**0.0223**

Table 7.2: Reconstruction errors of the four reconstruction methods similar to Tab. 7.1 for the two different datasets: glossy and natural colors [63].

images for the closest two samples in the lines (b) and (c) within the difference of 10^{-5} in the value of x or y. Since our clustering algorithm does not explicitly consider the continuity across the boundaries, we found that the shapes of the spectra do not necessarily vary smoothly across the boundaries. For example, while the cases (a) and (b) show smooth changes, the case (c) shows some discontinuous changes in the shape of the spectrum. For the case (c), the visual difference in the rendered images (Fig. 7.15, bottom) is small, but noticeable for interreflection.

Reconstruction of Saturated Colors To obtain physically plausible results in spectral rendering, spectral reflectances must be in the range of [0, 1] for every wavelength. Our formulation using PCA does not guarantee this property. That is, the reconstructed spectrum can be less than zero or more than one at certain wavelengths. We thus need to clamp the output spectra to the range of [0, 1] to guarantee this property. Clamped spectra, however, may not convert back to the input tristimulus color. Fig. 7.16 illustrates some examples of the case. We reconstructed the spectrum from the saturated colors, constraining each component of the RGB color to either zero or one. The values of the spectra reconstructed from such colors tend to be outside the range of $[0, 1]$. Meng et al. [76] pointed out that some colors are impossible to be represented within the range of $[0, 1]$, and suggested some adjustment techniques to produce physically plausible spectra in such cases. Since our formulation intrinsically supports the XYZ color space, we can use their technique.

We also visualize the ratio of the successful reconstruction in the range of $[0, 1]$ in the xy-plane (Fig. 7.17). The region enclosed by the solid line shows the range of visible chromacities and the region enclosed by the dotted line shows the coutor line of the measured spectra for the dataset I. The color map represents the ratio of the successful reconstruction. The color is assigned for each bin representing a small part in the xy-plane and the ratio is estimated by computing the number of successful and unsuccessful reconstructions for each bin. The red color means all reconstructed spectra are in the range of $[0, 1]$ and the blue color represents the spectra are outside the range of $[0, 1]$, or the visible region. We can observe that the reconstruction succeeds in a large part of the domain defined by the measurements yet fails in the outside of the domain.

The maximum error in Tab. 7.1 shows that there are cases where our method still fails to reproduce measured spectra. We do not claim that our method is capable of reproducing *any* natural spectra since some natural spectra can have very different characteristics than typical ones [125]. Our method is certainly not perfect and metamerism may still occur if we consider all possible spectra. In the paper, we tested our method with the measured spectra of Munsell Matte color chips. While those spectra certainly do not represent *all* the natural spectra, our method is still applicable when measured spectra has a few degree of freedoms as was confirmed firmly in color science [15, 72, 23] for different datasets. Our method thus can be seen as a general approach to find a compact and reversible (in the sense that it recovers the input colors) representation of a given measured spectra.

For example, if measured spectra for various metals are given and we use the input color to specify the color of a metal, our method reproduces spectra of various metals, *not*, Munsell Matte color chips. Tab. 7.2 shows such examples where the reconstruction errors are calculated for the two different datasets: *glossy* and *natural colors* [63]. Our method indeed generates more accurate results than the other existing methods even in these examples.

Reconstruction of General Spectra Our method does not support spectra with intrinsic distributions such as the spectra measured from the material with the flu-

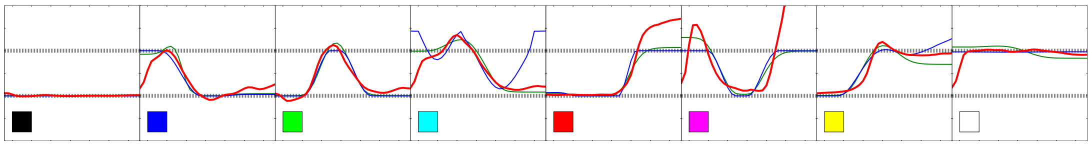


Figure 7.16: Comparison of the reconstructed spectra for the saturated colors with the method by Smits (blue), the method by Meng et al. (green), and our approach (red). We reconstructed the spectrum from the eight combinations of the saturated colors whose components of the color are equal to one or zero. The dotted lines illustrates the range of $[0, 1]$. The color plate in each plot shows the color converted from the reference spectrum.

orescence or the structural colors. For such spectra, principle component analysis might not lead to an efficient approximation. Our method does not directly support emission spectra either. One technique to use emission spectra within our method is scaling down the input emission tristimulus color into $[0, 1]^3$, converting it to an emission spectrum using our method, then scaling up the resulting spectrum. This technique, however, is not guaranteed at all to reproduce a measured emission spectrum with the same emission tristimulus color.

7.7 Summary

We introduced a novel method to reproduce spectral reflectances from tristimulus colors. Our key idea is to use prior knowledge on the actual measured spectra as studied in color science. Unlike existing methods in computer graphics, the use of measured spectra allows us to formulate the conversion process without smoothness heuristics. Our formulation also clarifies that a similar formulation in color science would have uncontrolled approximation error. In order to address this problem, we proposed a clustering technique to directly minimize the approximation error. At runtime, an input tristimulus color is converted to a spectrum using the precomputed basis functions and the conversion matrix to weights for those basis functions. Our experiments demonstrate that the proposed method can faithfully reproduce measured spectra without relying on existing heuristics or heavy optimizations at runtime. We expect that our method can be used for many different applications with spectral data. In addition, since our work introduces the mapping between RGB values a certain set of measured spectra, it might lead to more accurate color reproduction in printing, given RGB images.

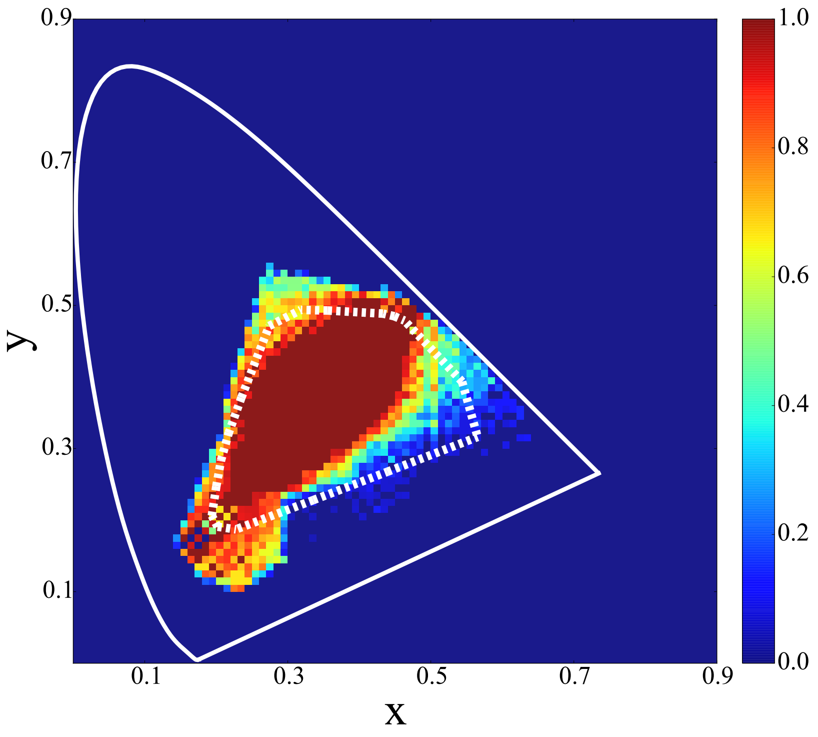


Figure 7.17: Visualization of the ratio of the reconstructed spectra in the range of $[0, 1]$ in the xy-plane. The solid line shows the region of the visible spectra and the dotted line shows the contor line of the measured spectra in the dataset I. We separated the xy-plane into a set of small regions and assigned the color for each region according to the estimated ratio of the spectra in the range.

Chapter 8

Conclusion

In this thesis, we presented several novel methods to achieve the efficiency and the accuracy in the simulation and the modeling in the light transport simulation. We focus on the two specific problems in light transport simulation. One is to improve the efficiency of the (Markov chain) Monte Carlo rendering, and the other is to improve the accuracy of the accuracy in the spectral reflectance reconstruction. In order to achieve the goal, we facilitated the general concept of bridging different spaces inherent to the problems in light transport simulation. In this thesis, we explored the three different techniques that each focuses on the corresponding representative spaces: strategy spaces, state spaces, and chromatic spaces. Each of the spaces represents the fundamental concepts corresponding to the problems of interest. We attempted to resolve the problem by facilitating the relationship between the spaces as components. Specifically, in this thesis, we developed the following three different methods appeared from the different contexts in light transport simulation:

First, we presented the method to bridge the strategy spaces of light transport simulations. The solution space of the rendering algorithms is called the strategy spaces. In this method, we bridge the strategy spaces by blending the results of different light transport simulation algorithms via machine learning. The key idea of the approach is to learn the relationship between a class of light paths based on the classification of the lighting effects and the performance of each algorithm. Unlike the existing techniques, the method does not require the underlying knowledge of the underlying combined rendering algorithms, which makes it easy to apply the approach to the different rendering algorithms.

Next, we presented the method to bridge the state spaces of Markov chain Monte Carlo rendering. We focus on bridging the two different state spaces with different mathematical background: the path space and the primary sample space. We provided the formulation to connect between the two state spaces by introducing a novel mapping from a path to a sequence of numbers, named as the inverse path sampler. This allowed us to combine different mutation algorithms designed for each state space. The inverse path sampler can be derived easily and we need not to modify the existing mutation algorithms. The numerical experiments demonstrate that this combination of the mutation techniques faithfully resolves the trade-off, where either one of the algorithms works better for some part of the scene.

Finally, we presented the method to bridge the chromatic spaces which determines the representation of spectra in order to improve the accuracy in the spectral reflectance reconstruction. Unlike the existing approaches in the computer graphic field, we incorporated the prior knowledge on the actual measured spectra to reconstructed the spectrum. The proposed method is achieved by facilitating the relationship between the original and lower dimensional representation of the spectra obtained by the principal component analysis, which has been originally studied in the color science field. In consequence, the reconstructed spectrum using the proposed approach can faithfully reproduce the shape of the spectra similar to the measured spectra, as well as retaining the original tristimulus values as possible.

References

[1] Abed, F. M., Amirshahi, S. H., and Abed, M. R. M. Reconstruction of reflectance data using an interpolation technique. *J. Opt. Soc. Am. A 26*, 3 (2009).

[2] Agahian, F., Amirshahi, S. A., and Amirshahi, S. H. Reconstruction of reflectance spectra using weighted principal component analysis. *Color Res. Appl. 33*, 5 (2008), 360–371.

[3] Amirshahi, S. H., and Amirhahi, S. A. Adaptive non-negative bases for reconstruction of spectral data from colorimetric information. *Optical Review 17*, 6 (2010), 562–569.

[4] Appel, A. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference* (New York, NY, USA, 1968), AFIPS '68 (Spring), ACM, pp. 37–45.

[5] Arvo, J. Backward ray tracing. In *Developments in Ray Tracing, ACM SIGGRAPH Course Notes* (1986), pp. 259–263.

[6] Ayala, F., Echavarri, F., Renet, P., and Negueruela, A. I. Use of three tristimulus values from surface reflectance spectra to calculate the principal components to reconstruct these spectra by using only three eigenvector. *J. Opt. Soc. Am. A 23*, 8 (Sep 2006), 2020–2026.

[7] Barakzehi, M., Amirshahi, S. H., Peyvandi, S., and Afjeh, M. G. Reconstruction of total radiance spectra of fluorescent samples by means of nonlinear principal component analysis. *J. Opt. Soc. Am. A 30*, 9 (2013), 1862–1870.

[8] Bekaert, P. *Hierarchical and stochastic algorithms for radiosity.* PhD thesis, Department of Computer Science, KU Leuven, Celestijnenlaan 200A, 3001 Heverlee, December 1999.

[9] Bitterli, B., Jakob, W., Novák, J., and Jarosz, W. Reversible jump Metropolis Light transport using inverse mappings. Tech. rep., arXiv preprint arXiv:1704.06835v1, 2017.

[10] Breiman, L. Random forests. *Machine Learning 45*, 1 (2001), 5–32.

[11] Brill, M. H. A non-PC look at principal components. *Color Res. Appl. 28*, 1 (2002).

[12] Campbell, III, A. T., and Fussell, D. S. Adaptive mesh generation for global diffuse illumination. *SIGGRAPH Comput. Graph. 24*, 4 (Sept. 1990), 155–164.

[13] Chandrasekhar, S. *Radiative Transfer.* 1960.

[14] Chern, J. R., and Wang, C. M. A novel progressive refinement algorithm for full spectral rendering. *Real-Time Imaging 11*, 2 (2005), 117–127.

[15] Cohen, J. Dependency of the spectral reflectance curves of the munsell color chips. *Psychonomic Science 1*, 1 (1964), 369–370.

[16] Cohen, M., Chen, S. E., Wallace, J. R., and Greenberg, D. P. A progressive refinement approach to fast radiosity image generation. *Computer Graphics (Proc. of SIGGRAPH 88) 22*, 4 (1988), 75–84.

[17] Cohen, M., Greenberg, D., Immel, D., and Brock, P. An efficient radiosity approach for realistic image synthesis. *IEEE Comput. Graph. Appl. 6*, 3 (Mar. 1986), 26–35.

[18] Cohen, M., and Greenberg, D. P. The hemi-cube: A radiosity solution for complex environments. *Computer Graphics (Proc. of SIGGRAPH 85) 19*, 3 (1985), 31–40.

[19] Cook, R. L., Porter, T., and Carpenter, L. Distributed ray tracing. *Computer Graphics (Proc. of SIGGRAPH 84) 18*, 3 (1984), 137–145.

[20] D., M. J., and S., B. K. Heuristics for ray tracing using space subdivision. *The Visual Computer 6*, 3 (1990), 153–166.

[21] Davis, P. J., and Rabinowitz, P. *Methods of Numerical Integration: Second Edition.* Academic Press, 1984.

[22] Dupont, D. Study of the reconstruction of reflectance curves based on tristimulus values: Comparison of methods of optimization. *Color Res. Appl. 27*, 2 (2002), 88–99.

[23] Fairman, H. S., and Brill, M. H. The principal components of reflectances. *Color Res. Appl. 29*, 2 (2004), 104–110.

[24] Fairman, H. S., Brill, M. H., and Hemmendinger, H. How the CIE 1931 color-matching functions were derived from Wright-Guild data. *Color Research & Application 22*, 1 (1997), 11–23.

[25] García-Beltrán, A., Nieves, J. L., Hernández-Andrés, J., and Romero, J. Linear bases for spectral reflectance functions of acrylic paints. *Color Res. Appl. 23*, 1 (1998).

[26] Georgiev, I., Krivanek, I., Davidovic, T., and Slusallek, P. Light transport simulation with vertex connection and merging. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), Article 192.

[27] Goral, C. M., Torrance, K. E., Greenberg, D. P., and Battaile, B. Modelling the interaction of light between diffuse surfaces. *Computer Graphics (Proc. of SIGGRAPH 84) 18*, 3 (1984), 212–222.

[28] Gortler, S. J., Schröder, P., Cohen, M. F., and Hanrahan, P. Wavelet radiosity. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1993), SIGGRAPH '93, ACM, pp. 221–230.

[29] Green, P. J. Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika 82* (1995), 711–732.

[30] Gritz, L., Stein, C., Kulla, C., and Conty, A. Open shading language. *ACM SIGGRAPH 2010 Talks* (2010).

[31] Guild, J. The colorimetric properties of the spectrum. 149–187.

[32] Hachisuka, T., and Jensen, H. W. Stochastic progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 28*, 5 (2009), Article 141.

[33] Hachisuka, T., Kaplanyan, A. S., and Dachsbacher, C. Multiplexed metropolis light transport. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4 (2014).

[34] Hachisuka, T., Ogaki, S., and Jensen, H. W. Progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008), Article 130.

[35] Hachisuka, T., Ogaki, S., and Jensen, H. W. Progressive photon mapping. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 27*, 5 (2008), 130.

[36] Hachisuka, T., Pantaleoni, J., and Jensen, H. W. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31*, 6 (2012), Article 191.

[37] Hajipour, A., and Shams-Nateri, A. Effect of classification by competitive neural network on reconstruction of reflectance spectra using principal component analysis. *Color Res. Appl.* (2016).

[38] Halmos, P. R. *Measure Theory.* Springer-Verlag, 1974.

[39] Hanika, J., Kaplanyan, A. S., and Dachsbacher, C. Improved half vector space light transport. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 34*, 4 (2015), 65–74.

[40] Hanrahan, P., Salzman, D., and Aupperle, L. A rapid hierarchical radiosity algorithm. *SIGGRAPH Comput. Graph. 25*, 4 (July 1991), 197–206.

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

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

[43] Havran, V. *Heuristic Ray Shooting Algorithms.* Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[44] Havran, V. *Heuristic Ray Shooting Algorithms.* Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[45] Hawkyard, C. Synthetic reflectance curves by additive mixing. *Journal of the Society of Dyers and Colourists 109*, 10 (1993), 323–329.

[46] Haykin, S. *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1998.

[47] Heckbert, P. S. Adaptive radiosity textures for bidirectional ray tracing. *Computer Graphics (Proc. SIGGRAPH) 24*, 4 (1990), 145–154.

[48] Heckbert, P. S. Discontinuity meshing for radiosity. In *Third Eurographics Workshop on Rendering* (1992), pp. 203–226.

[49] Hinton, G. E., Osindero, S., and Teh, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation 18*, 7 (2006), 1527–1554.

[50] Immel, D. S., Cohen, M. F., and Greenberg, D. P. A radiosity method for non-diffuse environments. *SIGGRAPH Comput. Graph. 20*, 4 (Aug. 1986), 133–142.

[51] Jakob, W. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org.

[52] Jakob, W., and Marschner, S. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31*, 4 (2012).

[53] Jakob, W., Regg, C., and Jarosz, W. Progressive expectation–maximization for hierarchical volumetric photon mapping. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 30*, 4 (2011).

[54] Jensen, H. W. Global illumination using photon maps. *Proc. Eurographics Symposium on Rendering* (1996), 21–30.

[55] Jensen, H. W. Global illumination using photon maps. In *Proc. Eurographics Workshop on Rendering* (1996), pp. 21–30.

[56] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. *CoRR abs/1408.5093* (2014).

[57] Judd, D. B., MacAdam, D. L., Wyszecki, G., Budde, H. W., Condit, H. R., Henderson, S. T., and Simonds, J. L. Spectral distribution of typical daylight as a function of correlated color temperature. *J. Opt. Soc. Am. A 54*, 8 (Aug 1964), 1031–1040.

[58] Kajiya, J. T. The rendering equation. *Computer Graphics (Proc. SIGGRAPH)* (1986), 143–150.

[59] Kalantari, N. K., Bako, S., and Sen, P. A Machine Learning Approach for Filtering Monte Carlo Noise. *ACM Transactions on Graphics (TOG) (Proceedings of SIGGRAPH 2015) 34*, 4 (2015).

[60] Kaplanyan, A., Hanika, J., and Dachsbacher, C. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4 (2014), 1–13.

[61] Kaplanyan, A. S., and Dachsbacher, C. Path space regularization for holistic and robust light transport. *Computer Graphics Forum (Proc. of Eurographics) 32*, 2 (2013), 63–72.

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

[63] Kohonen, O., Parkkinen, J., and Jaaskelainen, T. Databases for spectral color science. *Color Res. Appl. 31*, 5 (2006).

[64] Kronander, J., Schön, T. B., and Unger, J. Pseudo-marginal Metropolis light transport. In *SIGGRAPH Asia 2015 Technical Briefs* (2015), pp. 13:1–13:4.

[65] Ladický, L., Jeong, S., Solenthaler, B., Pollefeys, M., and Gross, M. Data-driven fluid simulations using regression forests. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015), 199.

[66] Lafortune, E., and Willems, Y. D. Bi-directional path-tracing. *Proc. Compugraphics* (1993), 145–153.

[67] Lafortune, E. P., and Willems, Y. D. Bi-directional path tracing. In *Compugraphics '93* (1993), pp. 145–153.

[68] Lessig, C. *Modern Foundations of Light Transport Simulation.* PhD thesis, University of Toronto, 2012.

[69] Li, T.-M., Lehtinen, J., Ramamoorthi, R., Jakob, W., and Durand, F. Anisotropic Gaussian mutations for Metropolis light transport through Hessian-Hamiltonian dynamics. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 34*, 6 (2015), 209:1–209:13.

[70] Lischinski, D., Tampieri, F., and Greenberg, D. P. Discontinuity meshing for accurate radiosity. *IEEE Computer Graphics and Applications 12*, 6 (Nov 1992), 25–39.

[71] Malley, T. A shading method for computer generated images. Master's thesis, University of Utah, 1988.

[72] Maloney, L. T. Evaluation of linear models of surface spectral reflectance with small numbers of parameters. *J. Opt. Soc. Am. A 3*, 10 (1986), 1673–1683.

[73] Marimont, D. H., and Wandell, B. A. Linear models of surface and illuminant spectra. *J. Opt. Soc. Am. A 9*, 11 (1992), 1905–1913.

[74] Marinari, E., and Parisi, G. Simulated tempering: a new Monte Carlo scheme. *EPL (Europhysics Letters) 19*, 6 (1992), 451.

[75] McKay, M. D., Beckman, R. J., and Conover, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics 21*, 2 (1979), 239–245.

[76] Meng, J., Simon, F., Hanika, J., and Dachsbacher, C. Physically meaningful rendering using tristimulus colours. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering) 34*, 4 (2015), 31–40.

[77] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. Equation of state calculations by fast computing machines. *J. Chem. Phys. 21*, 6 (1953).

[78] Meyer, G. W. Wavelength selection for synthetic image generation. *Comput. Vision Graph. Image Process. 41*, 1 (1988), 57–79.

[79] Miranda, A. A., Le Borgne, Y.-A., and Bontempi, G. New routes from minimal approximation error to principal components. *Neural Processing Letters 27*, 3 (2008), 197–207.

[80] Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H.-P., and Ritschel, T. Deep shading: Convolutional neural networks for screen-space shading.

[81] Neumann, A., Neumann, L., Bekaert, P., Willems, Y. D., and Purgathofer, W. *Importance-driven Stochastic Ray Radiosity.* Springer Vienna, Vienna, 1996, pp. 111–121.

[82] Neumann, L., Feda, M., Kopp, M., and Purgathofer, W. *A New Stochastic Radiosity Method for Highly Complex Scenes.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 201–213.

[83] Neumann, L., Purgathofer, W., Tobler, R. F., Neumann, A., Eliás, P., Feda, M., and Pueyo, X. *The Stochastic Ray Method for Radiosity.* Springer Vienna, Vienna, 1995, pp. 206–218.

[84] Nguyen, R. M. H., Prasad, D. K., and Brown, M. S. Training-based spectral reconstruction from a single rgb image. *ECCV* (2014), 186–201.

[85] Niederreiter, H. *Random Number Generation and quasi-Monte Carlo Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[86] Nishita, T., and Nakamae, E. Continuous tone representation of three dimensional objects taking account of shadow and interreflection. *Computer Graphics (Proc. of SIGGRAPH 85)* (1985), 23–30.

[87] Pantaleoni, J. Charted Metropolis light transport. SIGGRAPH '17, ACM.

[88] Parkkinen, J. P. S., Hallikainen, J., and Jaaskelainen, T. Characteristic spectra of munsell colors. *J. Opt. Soc. Am. A 6*, 2 (1989), 318–322.

[89] Pattanaik, S. N., and Mudur, S. P. Adjoint equations and random walks for illumination computation. *ACM Trans. Graph. 14*, 1 (Jan. 1995), 77–102.

[90] Peercy, M. S. Linear color representations for full speed spectral rendering. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, pp. 191–198.

[91] Raso, M., and Fournier, A. A piecewise polynomial approach to shading using spectral distributions. *Graphics Interface* (1991), 40–46.

[92] Ren, P., Wang, J., Gong, M., Lin, S., Tong, X., and Guo, B. Global illumination with radiance regression functions. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32*, 4 (2013), 130:1–130:12.

[93] Ren, S., Cao, X., Wei, Y., and Sun, J. Face alignment at 3000 fps by regressing local binary features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014).

[94] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV) 115*, 3 (2015), 211–252.

[95] Sbert, M. Error and complexity of random walk monte carlo radiosity. *IEEE Transactions on Visualization and Computer Graphics 3*, 1 (Jan 1997), 23–38.

[96] Sbert, M. Optimal source selection in shooting random walk monte carlo radiosity. *Computer Graphics Forum 16*, 3 (1997), C301–C308.

[97] Sharma, G., and Wang, S. Spectrum recovery from colorimetric data for color reproductions. *Color Imaging: Device-Independent Color, Color Hardcopy, and Applications VII. Proc. SPIE 4663* (2002), 8–14.

[98] Shirley, P., Wade, B., Hubbard, P. M., Zareski, D., Walter, B., and Greenberg, D. P. Global illumination via density-estimation. In *Rendering Techniques 1995 (Proceedings of the Eurographics Workshop on Rendering)* (1995), pp. 219–230.

[99] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from a single depth imaget. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2011).

[100] Sillion, F., and Puech, C. A general two-pass method integrating specular and diffuse reflection. *SIGGRAPH Comput. Graph. 23*, 3 (July 1989), 335–344.

[101] Sillion, F. X., Arvo, J. R., Westin, S. H., and Greenberg, D. P. A global illumination solution for general reflectance distributions. *SIGGRAPH Comput. Graph. 25*, 4 (July 1991), 187–196.

[102] Smith, T., and Guild, J. The C.I.E. colorimetric standards and their use. *Transactions of the Optical Society 33*, 3 (1931), 73.

[103] Smits, B. An rgb-to-spectrum conversion for reflectances. *Journal of Graphics Tools 4*, 4 (1999), 11–22.

[104] Smits, B., Arvo, J., and Greenberg, D. A clustering algorithm for radiosity in complex environments. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, ACM, pp. 435–442.

[105] Stokes, M., Anderson, M., Chandrasekar, S., and Motta, R. A standard default color space for the internet — sRGB, November 1996.

[106] Sun, Y., Fracchia, F. D., Drew, M. S., and Calvert, T. W. A spectrally based framework for realistic image synthesis. *The Visual Computer 17*, 7 (2001), 429–444.

[107] Tampieri, F., and Lischinski, D. The constant radiosity assumption syndrome. 83–92.

[108] Tang, D., Yu, T.-H., and Kim, T.-K. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *The IEEE International Conference on Computer Vision (ICCV)* (2013).

[109] Tzeng, D.-Y., and Berns, R. S. A review of principal component analysis and its applications to color technology. *Color Res. Appl. 30*, 2 (2005), 84–98.

[110] Union, I. T. Recommendation itu-r bt.709: Parameter values for the hdtv standards for production and international programme exchange, 2002.

[111] Union, I. T. Recommendation ITU-R BT.709-6: Parameter values for the HDTV standards for production and international programme exchange, 2015.

[112] Veach, E. *Robust Monte Carlo methods for light transport simulation.* PhD thesis, Stanford University, USA, 1998. AAI9837162.

[113] Veach, E., and Guibas, L. Bidirectional estimators for light transport. In *Proc. Eurographics Workshop on Rendering* (1994), pp. 147–162.

[114] Veach, E., and Guibas, L. Metropolis light transport. In *SIGGRAPH '97* (1997), pp. 65–76.

[115] Veach, E., and Guibas, L. J. Bidirectional estimator for light transport. *Proc. Eurographics Symposium on Rendering* (1994), 147–162.

[116] Veach, E., and Guibas, L. J. Optimally combining sampling techniques for Monte Carlo rendering. *Proc. SIGGRAPH '95* (1995), 419–428.

[117] Veach, E., and Guibas, L. J. Metropolis light transport. *Proc. of SIGGRAPH 97* (1997), 65–76.

[118] Vorba, J., Karlík, O., Šik, M., Ritschel, T., and Křivánek, J. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33*, 4 (aug 2014).

[119] Wallace, J. R., Cohen, M. F., and Greenberg, D. P. A two-pass solution to the rendering equation: A synthesis of ray tracing and radiosity methods. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1987), SIGGRAPH '87, ACM, pp. 311–320.

[120] Walter, B., Marschner, S. R., Li, H., and Torrance, K. E. Microfacet models for refraction through rough surfaces. *Proc. Eurographics Symposium on Rendering* (2007), 195–206.

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

[122] Worthey, J. A., and Brill, M. H. Principal components applied to modeling: Dealing with the mean vector. *Color Res. Appl. 29*, 4 (2004).

[123] Wright, W. D. A re-determination of the trichromatic coefficients of the spectral colours. *Transactions of the Optical Society 30*, 4 (1929), 141.

[124] Wyman, C., Sloan, P.-P., and Shirley, P. Simple analytic approximations to the CIE XYZ color matching functions. *Journal of Computer Graphics Techniques 2*, 2 (2013), 1–11.

[125] Wyszecki, G., and Stiles, W. S. *Color Science: Concepts and Methods, Quantitative Data and Formulae*, 2nd ed. Wiley-Interscience, 2000.

[126] Zhang, X., and Xu, H. Reconstructing spectral reflectance by dividing spectral space and extending the principal components in principal component analysis. *J. Opt. Soc. Am. A 25*, 2 (2008), 371–378.