UNIVERSITY OF CALIFORNIA, SAN DIEGO

Robust Light Transport Simulation using Progressive Density Estimation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Toshiya Hachisuka

Committee in charge:

Professor Henrik Wann Jensen, Chair
Professor James Arvo
Professor Serge Belongie
Professor Samuel Buss
Professor Sanjoy Dasgupta

2011

The dissertation of Toshiya Hachisuka is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California, San Diego

2011

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would not have been able to complete this work without help of many people. First, I would like to thank my advisor, Henrik Wann Jensen, who is exactly the reason that I decided to move thousands of miles away from where I did my undergraduate study. His research work sparked my interest in light transport simulation, and I am very lucky to be able to work with him for building up this dissertation. I learned a lot from him about what it means to be a great researcher during my time at UCSD. I would also like to thank Matthias Zwicker for having been a wonderful supporter during my first years of study. Even though he is not my advisor, he was always there for discussion and help. Thanks to his support, the start of my journey to PhD was very smooth. I hope to pay my gratitude toward both Henrik and Matthias forward throughout my next career.

I thank and pay my sincere respect to the members of my dissertation committee, James Arvo, Serge Belongie, Samuel Buss, and Sanjoy Dasgupta, who kindly agreed to read and evaluate this dissertation. I would like to further thank my undergraduate advisor, Seiichi Koshizuka, who supported me to come to here for PhD study. My undergraduate research project under him formed my interest for simulating the real world.

My time at UCSD has been very enjoyable thanks to my colleagues, in particular: Oleg Bisker, Carlos Dominguez Caballero, Bin Chen, Yi Chen, Will Chang, Craig Donner, Wojciech Jarosz, Neel Joshi, Han Suk Kim, Arash Keshmirian, Wan-Yen Lo, Krystle de Mesa, Iman Mostafavi, Iman Sadeghi, and Marios Papas. I thank them for stimulating discussion, help for miscellaneous things, and many relaxing conversations. I am also fortunate to collaborate with wonderful people during my study: Kevin Dale, Greg Humphreys, Jason Lawrence, Shree Nayar, Shinji Ogak, Jacopo Pantaleoni, Ravi Ramamoorthi, Richard Weistroffer, and Sung Eui Yoon. I thank Youichi Kimura for kindly providing his model data for testing my work. I also owe a lot to all the members of Lab Dodgson (aka “Doji-ken”) who cultivated my interests in computer graphics.

Last but not least, I would like to thank my family. My parents, Tomoko and Shunji, have always encouraged me to do things I like such as studying computer graphics. Occasionally having a chat on random things with my sister, Saaya, has been very relaxing. Playing with our beloved dog, Taro, was one of the important events to have when I was visiting Japan. I dedicate this dissertation to them.

Parts of this dissertation are based on papers which I have co-authored with others. I was the primary investigator and author of all the following papers.

- Chapter 5 is a reproduction of the material published in the article:
  Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. "Progressive Photon Mapping", ACM Transactions on Graphics, 27(5), 130:1–130:8, 2008.

- Chapter 7 is a reproduction of the material published in the article:
  Toshiya Hachisuka and Henrik Wann Jensen. "Stochastic Progressive Photon Mapping", ACM Transactions on Graphics, 28(5), 141:1–141:8, 2009.

- Chapter 6 is a reproduction of the material published in the article:
  Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. "A Progressive Error Estimation Framework for Photon Density Estimation", ACM Transactions on Graphics, 29(6), 144:1–144:12, 2010
  and the talk:
  Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. "An Error-Estimation Framework for Photon Density Estimation", ACM SIGGRAPH 2010 Talks, 3:1–3:1, 2010.

- Chapter 10 is a reproduction of the material published in the talk:
  Toshiya Hachisuka and Henrik Wann Jensen. "Parallel Progressive Photon Mapping on GPUs", ACM SIGGRAPH ASIA 2010 Sketches, 54:1–54:1, 2010.

- Chapter 8 is a reproduction of the material in the article:
  Toshiya Hachisuka and Henrik Wann Jensen. "Robust Adaptive Photon Tracing using Photon Path Visibility", ACM Transactions on Graphics, in publication, 2011.

- Chapter 9 is a reproduction of the material in the article:
  Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. "Multisampled Progressive Photon Mapping", submitted, 2011.

VITA

2006	Bachelor of Engineering, *dean award (equivalent to summa cum laude)*, The University of Tokyo
2011	Doctor of Philosophy, University of California, San Diego

PUBLICATIONS

Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen, "Multisampled Progressive Photon Mapping", submitted, 2011.

Toshiya Hachisuka and Henrik Wann Jensen. "Robust Adaptive Photon Tracing using Photon Path Visibility", ACM Transactions on Graphics, in publication, 2011.

Toshiya Hachisuka and Henrik Wann Jensen. "Parallel Progressive Photon Mapping on GPUs", ACM SIGGRAPH ASIA 2010 Sketches, 54:1–54:1, 2010.

Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. "A Progressive Error Estimation Framework for Photon Density Estimation", ACM Transactions on Graphics, 29(6), 144:1–144:12, 2010.

Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. "An Error-Estimation Framework for Photon Density Estimation", ACM SIGGRAPH 2010 Talks, 3:1–3:1, 2010.

Toshiya Hachisuka and Henrik Wann Jensen. "Stochastic Progressive Photon Mapping", ACM Transactions on Graphics, 28(5), 141:1–141:8, 2009.

Craig Donner, Jason Lawrence, Ravi Ramamoorthi, Toshiya Hachisuka, Henrik Wann Jensen, and Shree Nayar. "An Empirical BSSRDF Model", ACM Transactions on Graphics, 28(3), 30:1–30:10, 2009.

Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. "Progressive Photon Mapping.", ACM Transactions on Graphics, 27(5), 130:1–130:8, 2008.

Toshiya Hachisuka, Wojciech Jarosz, Richard Peter Weistroffer, Kevin Dale, Greg Humphreys, Matthias Zwicker, and Henrik Wann Jensen. "Multidimensional Adaptive Sampling and Reconstruction for Ray Tracing", ACM Transactions on Graphics, 27(3), 33:1–33:10, 2008.

ABSTRACT OF THE DISSERTATION

Robust Light Transport Simulation using Progressive Density Estimation

by

Toshiya Hachisuka

Doctor of Philosophy in Computer Science

University of California, San Diego, 2011

Professor Henrik Wann Jensen, Chair

This dissertation introduces a new light transport simulation framework that significantly expands a class of scene configurations that we can handle. The main contribution is a novel density estimation method, called progressive density estimation, which addresses fundamental limitations of existing density estimation methods. The key feature of progressive density estimation is that the method does not need to store a full set of samples to guarantee convergence to the correct solution. Progressive density estimation led to a new light transport algorithm which can simulate many optical configurations that would be impractical to handle with any existing algorithm. In particular, the algorithm can efficiently simulate complex lighting fixtures from the filament/LED-level for the first time.

This dissertation also extends this basic framework of progressive density estimation. We first introduce a practical error estimator for progressive density estimation. This method can estimate how much expected error exists for a given computed solution without needing any knowledge of the correct solution. Since we often need to estimate average illumination over a region that is unknown before computation in computer graphics, we developed stochastic progressive density estimation which provides a simple solution to this problem. This estimator extends progressive density estimation for computing average density over unknown region with provable convergence.

In order to improve computational efficiency of the proposed framework, we applied an adaptive Markov chain Monte Carlo method to light transport simulation. With this adaptive algorithm, we can focus computation on only to the visible region. To our knowledge, this is the first application of adaptive Markov chain Monte Carlo methods in light transport simulation. We also propose a novel framework that achieves the adaptive combination of progressive density estimation and other approaches based on Monte Carlo integration. In order to develop this framework, we conducted theoretical analysis of a provably good combination of density estimation methods and Monte Carlo integration. For parallel computation of the proposed framework, we developed a new spatial hashing method. This new hashing algorithm is designed to work correctly regardless of the result of contentions in parallel processes as opposed to avoiding the contentions.

Chapter 1

Introduction

One important aspect of many applications of computer graphics is the requirement of accurate recreations of the physical world. For example, computer-generated images used for product design need to be as visually close as possible to the real world correspondences. Applications in entertainment industry such as movie production, advertisements, and video games increasingly demand photorealism of computer-generated images. Computer simulation of how light interacts with given virtual shapes and materials, or in other words, *light transport simulation*, is a natural approach to achieve photorealism if the images should recreate appearances of the real world.

Light transport simulation has been a core topic in computer graphics research over a few decades, and the effort led to many practical algorithms to the date. Initial effort toward light transport simulation dealt with computation of shadows from one object to another [7]. Ward introduced the idea of ray tracing, which samples a path of light by recursive computation of light interactions [130]. Cook et al. [17] later implemented the idea of random sampling of paths to simulate various optical effects. These work pioneered the light transport problem in computer graphics, however, they omitted interreflection of light between surfaces.

One earliest attempt to simulate general interreflection of light between surfaces is the radiosity method [37]. The radiosity method was initially developed as a method for simulating heat transfer in the engineering field [132]. Since heat transfer between distance surfaces is in fact due to radiation of photons, this particular problem is the same as the light transport problem. This work led to a bloom of work on light transport simulation in computer graphics [16, 39, 48, 106].

Although the radiosity method is the first successful method that simulates interreflection of light between surfaces, the method made a simplifying assumption that surfaces reflect light uniformly over all directions. This assumption limits types of materials that the radiosity method can simulate to essentially the one with a matte appearance. In order to remove this limitation, Immel et al. proposed a recursive integral equation of light transport that considers generalized surface reflectance profiles [57]. Concurrently, Kajiya proposed a similar formulation, called *the rendering equation* [66]. The main difference of these two work is that Immel et al. solved this equation in the same framework as the radiosity method, whereas Kajiya proposed another approach to solve light transport based on Monte Carlo integration methods.

As the Monte Carlo integration turned out to be more efficient than the linear equations approach employed in the radiosity method under general settings, this work by Kajiya led to another bloom of research in light transport simulation algorithms. At the same time, the original method proposed by Kajiya, the *path tracing* algorithm [66], posed a challenging problem of keeping computational efficiency for arbitrary scene input. For example, although path tracing works well for scenes with matte materials and large light sources, it becomes very inefficient in the presence of reflections of light from a small light source due a mirror or glass. In other words, path tracing is not *robust* for this type of scenes.

Improving the robustness has been a focus of research in light transport simulation. One interesting history of the development is that, in contrast to how real light behaves, the initial algorithm by Kajiya was using computation starting from the eye. This is a valid approach as the Stokes-Helmholtz reversion-reciprocity principle (commonly known as Helmholtz reciprocity) [52, 109] states that the direction of a light path can be reversed without affecting its throughput. Arvo [8] however proposed to solve the light transport problem starting from light sources, and demonstrated that this approach captures a certain type of light paths more efficiently than the inverse approach used in path tracing. This work later led to another class of algorithms that start computation from both light sources and the eye at the same time [71, 118]. Another direction of effort has been to improve efficiency of sampling in the Monte Carlo process. Metropolis light transport [119] is the application of Markov chain Monte Carlo sampling, which improves the efficiency under strongly localized illumination effects such as illumination coming through a slight opening of door. Several work [15, 67, 73] improved upon this basic algorithm. All of these algorithms are all classified as unbiased Monte Carlo methods, where the expected solution is exactly equal to the correct solution.

The above algorithms are all considering the rendering equation as an integral equation, but there is another important class of light transport algorithms based on photon density estimation [61, 104, 121]. The basic algorithm is a two pass algorithm where the first pass distributes packets of light energy, called *photons*, based on light transport simulation starting from light sources, and the second pass estimates illumination at different locations within the scene by estimating density of photons. This class of methods formulates light transport as a density estimation problem.

While photon density estimation is robust to many scenes, fundamental limitations of density estimation has been restricting its usage only to computing a rough solution of light transport. This is first because any existing density estimation algorithm is a biased algorithm, meaning that expected solutions of a biased algorithm are different from the correct solutions. Another important reason is that existing density estimation algorithms are, though convergent in theory, not convergent in practice. Although theoretical conditions for making density estimation convergent has been known for a while, these conditions are mainly of theoretical interests and tweaking parameters of density estimation based on the conditions is necessary to obtain converging results. It is thus considered that using unbiased algorithms is the only option for accurate light transport simulation. Recent advancement in biased light transport algorithms has focused on improving efficiency under restricted scene configurations [122, 123], and no research have taken the direction to develop a new biased algorithm that can compete with the unbiased algorithms under general scene configurations.

The goal of this dissertation work is to develop a robust general algorithm that simulates light transport under general scene configurations. The main idea builds upon a new density estimation framework, called *progressive density estimation*, which addresses fundamental limitations of existing density estimation methods. Progressive density estimation performs density estimation with provable convergence to the correct solution in a single run. The application of this framework is a new lighting simulation algorithm which can simulate many optical configurations that would be impractical with the use of any existing algorithm. This dissertation work blurs a border between the biased algorithms and the unbiased algorithms as progressive density estimation is a biased algorithm yet the result asymptotically converges to the correct solution similar to the unbiased algorithms. To the best of our knowledge, this is the first biased light transport framework that leverages the convergence as the key property.

1.1 Contributions

Progressive Density Estimation

We introduce a new density estimation method called progressive density estimation that converges to the correct density value as more samples are used. The key feature is that the convergence to the correct solution can be achieved simply by iteratively adding more samples. Unlike other density estimation methods, it possible to compute a solution with any desired accuracy without multiple runs. This is a general density estimation algorithm that is applicable beyond computer graphics.

Progressive Photon Mapping

Based on progressive density estimation, we developed a new global illumination algorithm called progressive photon mapping. Each newly added sample results in an increasingly accurate light transport solution that can be visualized for progressive feedback. Compared with existing light transport algorithms, progressive photon mapping provides an efficient and robust alternative in the presence of complex light transport. This is the first light transport algorithm that can robustly and accurately capture specular-diffuse-specular light paths without putting any restriction on scene configurations.

Stochastic Density Estimation

In computer graphics, we often need to estimate average illumination over a region that is unknown before computation. A prominent example is a lens simulation where we need to compute average illumination coming to the sensor through a potentially complex footprint of the aperture through the lens. We extended progressive density estimation for dealing with such problem settings. This extension, stochastic progressive density estimation, does not need an explicit representation of a region that we compute average density, yet provide provable convergence to the correct average density in the limit. This method is also a general density estimation method that is applicable for problems other than light transport simulation.

Error Estimation Framework for Progressive Density Estimation

Although provable convergence to the correct solution in progressive density estimation is theoretically appealing, in many applications of light transport simulation, it is important to know how much computation is necessary to achieve a given allowable computation error. We provide a practical error estimation framework for progressive density estimation to achieve this goal. The resulting algorithm can estimate how much error is expected for a given computed illumination, without needing any input of the correct solution. This work provides one of the first practical error estimation methods for a biased light transport simulation.

Adaptive Photon Tracing based on Visibility

Existing light transport simulation algorithms based on photon density estimation become inefficient when we focus on simulating light transport in a small part of a scene. We developed a new adaptive computation algorithm that automatically concentrates computational effort to the region of interest. The key idea is the use of visibility of samples, and we formulate photon trajectory simulation as a sampling problem using adaptive Markov chain Monte Carlo methods. To the best of our knowledge, this is the first application of adaptive Markov chain Monte Carlo sampling in light transport simulation.

Hybrid Estimator for Progressive Density Estimation

We provide a unified framework that adaptively combines progressive density estimation and unbiased Monte Carlo integration methods for light transport simulation. In order to take the best of both approaches without relying on a heuristic choice of the algorithms, we developed an application of multiple importance sampling framework [118] to the combination of progressive photon mapping and unbiased Monte Carlo ray tracing. We extended the theoretical analysis of a provably optimal combination by considering the presence of a biased method, namely progressive density estimation.

Stochastic Spatial Hashing

Range queries occupy a major part of computation in progressive density estimation. In particular, since progressive density estimation iteratively accumulate statistics of independent samples by means of range queries in multiple passes, not only that the number of range queries can be quite high, but also that the repeated constructions of an acceleration data structure for range queries can be quite costly. We developed a simple data-parallel range query algorithm based on a novel hashing algorithm called stochastic hashing. The algorithm is suitable for a massively parallel computation platform such as recent graphics processing units. The key feature is that the method incorporates the contention of parallel processes as a stochastic process of the algorithm, rather than trying to avoid the contention.

1.2 Organization

The dissertation consists of eleven chapters. The following first three chapters (Chapter 2 to Chapter 4) review background materials of this dissertation work, the next six chapters describe the original contributions (Chapter 5 to Chapter 10), and the last chapter concludes with the perspective for future work.

Chapter 2 and Chapter 3 introduce basic concepts of Monte Carlo integration and density estimation, which are both fundamental tools that the dissertation work builds upon. Chapter 4 describes how we can use these tools to solve the light transport problem based on the rendering equation. Readers familiar with these basic concepts can skip these three chapters.

Chapter 5 builds a novel density estimation algorithm, *progressive density estimation*, which is applied to the light transport problem as *progressive photon mapping*. This is a new density estimation algorithm that has built-in provable convergence to the correct solution. The application to light transport, progressive photon mapping, is the first light transport algorithm that demonstrates robust simulation of realistic lighting fixtures such as light bulbs. Chapter 6 provides a practical error estimation framework for progressive density estimation, which has been commonly considered possible only with unbiased Monte Carlo light transport simulation. Chapter 7 extends the original progressive density estimation algorithm to be able to estimate average density over an arbitrary region. This average density estimator improves efficiency for common light transport configurations in computer graphics such as computing average illumination (i.e., density of photons) over a region seeing through a complex lens system. The proposed algorithm is also the first to achieve provable convergence of average density estimation over an arbitrary region.

Chapter 8 introduces a simple adaptive computation method that improves the efficiency of progressive photon mapping under scene configurations that is very inefficient to handle with existing light transport algorithms. This is the first application of some recent development in Monte Carlo integration into light transport simulation in computer graphics. Although progressive photon mapping is a unified algorithm which can robustly handle many scene configurations, existing methods can sometimes be more efficient for some scenes. Chapter 9 describes a novel framework that automatically combines progressive photon mapping and the unbiased Monte Carlo methods based on scene configurations. This work provides a unified view of progressive photon mapping and other light transport algorithms. Chapter 10 looks into a computational consideration of progressive density estimation on a highly parallel processor such as a graphics processing unit. This chapter then introduces a novel parallel spatial hashing algorithm, *stochastic hashing*, that utilizes the statistical identity of the density estimation problem in order to parallelize its computation with a simple algorithm.

Chapter 11 summarizes the contributions and discusses possible future work.

Chapter 2

Monte Carlo Integration

This chapter reviews Monte Carlo integration. Monte Carlo integration is a general numerical integration method that uses random numbers. Since the rendering equation formulates the light transport problem as an integral equation, many light transport algorithms directly build upon Monte Carlo integration. This chapter introduces some basic concepts and techniques of Monte Carlo integration that we will be using in the following chapters.

2.1 Background

Consider a definite integral over a domain $\Omega = [0,1]^d$:

$$I = \int_{\Omega} f(\vec{x})\, d\vec{x}, \tag{2.1}$$

where $\vec{x} = (x_1, x_2, \ldots, x_d) \in \Omega$ is a point in the domain of integration, d is the number of dimensions of the domain Ω, and $f : \Omega \to \mathbb{R}$ is a scalar function. In general, there is no single analytical approach that can solve such an integral for all types of the function f. We therefore often need to use *numerical integration* in order to obtain an approximate solution of the integral.

One popular approach is to use a *quadrature rule*, which takes a weighted sum of point-wise evaluations (i.e., *samples*) of the function f at predetermined locations:

$$I \approx \sum_{i_1=1}^{n} \sum_{i_1=1}^{n} \cdots \sum_{i_d=1}^{n} w_{i_1} w_{i_1} \cdots w_{i_d} f(x_{i_1}, x_{i_2}, \ldots, x_{i_d}), \tag{2.2}$$

where n is the number of samples along each axis, w_{i_d} is the weight of i_dth coordinate of the sample. Common choice of weight includes Newton–Cotes quadrature for equally spaced samples, and Gaussian quadrature and Clenshaw–Curtis quadrature for unequally spaced samples. Although the quadrature rule approach is applicable to a general function, the total number of samples needed is exponential to the number of dimensions $O(n^d)$, making its application to high dimensional integrals impractical. This problem is often referred as the curse of dimensionality [89].

Monte Carlo integration [85] is another numerical integration method that takes a weighted sum of point-wise estimations of the function at *random* locations:

$$I \approx \sum_{i_1=1}^{N} w_i f(x_i), \tag{2.3}$$

where N is the total number of samples and w_i is the weight for the ith sample. The number of samples in Monte Carlo integration does not have direct dependence on the number of dimensions d. This independence is particular useful in the light transport problem, since it can be formulated as integration over a very high dimensional (or in general, infinite dimensional) domain. Note that it does not mean that Monte Carlo integration is completely independent from the number of dimensions as we discuss later. The true advantage of Monte Carlo integration is in its flexible placement of samples.

2.2 Methods

2.2.1 Basic Theory

Before describing techniques of Monte Carlo integration, we first review the basic properties of Monte Carlo integration. We use 1D examples over the domain $[0,1]$ in the following discussion for brevity.

Probability Density Functions and Cumulative Distribution Functions

Suppose that we generate $x \in [0,1]$ that satisfies $a < x < b$. We define the *probability* of generating such x as

$$\mathrm{Prob}[a < x < b] = \int_a^b p(x)\,dx = P(b) - P(a). \tag{2.4}$$

$p(x)$ is called a *probability density function*, and $P(x)$ is called a *cumulative distribution function*. Note that

$$\frac{dP(x)}{dx} = p(x). \tag{2.5}$$

Following the definition, integral of any probability density function is equal to 1;

$$\int_0^1 p(x)\,dx = \mathrm{Prob}[0 < x < 1] = 1, \tag{2.6}$$

and any cumulative distribution functions satisfies $P(1) = 1$ and $P(0) = 0$ in this domain. Note also that the probability of generating a point (i.e., $x = a = b$) is zero, but its probability density can be non-zero and more than one.

Expected Values and Variance

The expected value of a random variable $f(x)$ with the probability density function $p(x)$ is

$$E[f(X)] = \int_0^1 f(x)p(x)\,dx \tag{2.7}$$

and the variance is

$$\mathrm{V}[f(X)] = \mathrm{E}\left[(f(X) - \mathrm{E}[f(X)])^2\right]. \tag{2.8}$$

If samples are independent from each other, the expression for the variance can be simplified as

$$\begin{aligned} \mathrm{V}[f(X)] &= \mathrm{E}\left[f(X)^2 - 2f(X)\,\mathrm{E}[f(X)] + \mathrm{E}[f(X)]^2\right] \\ &= \mathrm{E}[f(X)^2] - 2\,\mathrm{E}[f(X)\,\mathrm{E}[f(X)]] + \mathrm{E}[\mathrm{E}[f(X)]^2] \\ &= \mathrm{E}[f(X)^2] - 2\,\mathrm{E}[f(X)]^2 + \mathrm{E}[f(X)]^2 \\ &= \mathrm{E}[f(X)^2] - \mathrm{E}[f(X)]^2. \end{aligned} \tag{2.9}$$

Convergence Rate of Monte Carlo Integration

Suppose that we would like to solve the definite integral

$$\int_0^1 f(x)\,dx. \tag{2.10}$$

Using the definition of expected values, we can formulate the definite integral as the expected value of a random variable $f(X)/p(X)$ using the probability density function $p(x)$ since;

$$\mathrm{E}\left[\frac{f(X)}{p(X)}\right] = \int_0^1 \frac{f(x)}{p(x)} p(x)\,dx = \int_0^1 f(x)\,dx. \tag{2.11}$$

This means the a sampled value that we consider is $\frac{f(x)}{p(x)}$ rather than $f(x)$ itself. Monte Carlo integration approximates this expected value with a finite number of samples $(x_1, x_2, \ldots, x_N)$.

$$\mathrm{E}\left[\frac{f(X)}{p(X)}\right] \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{2.12}$$

Since the estimate in Monte Carlo integration is a random variable, in order to quantify the error of the estimate, we consider the expected squared error as follows:

$$\mathrm{E}\left[\left(\mathrm{E}\left[\frac{f(X)}{p(X)}\right] - \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)}\right)^2\right] \tag{2.13}$$

We can expand this equation assuming that estimates $\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$ as random variables are independent from each other

$$\begin{aligned}
&\mathrm{E}\left[\left(\mathrm{E}\left[\frac{f(X)}{p(X)}\right]-\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2\right]\\
&=\mathrm{E}\left[\mathrm{E}\left[\frac{f(X)}{p(X)}\right]^2-2\,\mathrm{E}\left[\frac{f(X)}{p(X)}\right]\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}+\left(\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2\right]\\
&=\mathrm{E}\left[\frac{f(X)}{p(X)}\right]^2-2\,\mathrm{E}\left[\frac{f(X)}{p(X)}\right]^2+\mathrm{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2\right] \qquad (2.14)\\
&=\mathrm{E}\left[\left(\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2\right]-\mathrm{E}\left[\frac{f(X)}{p(X)}\right]^2\\
&=\frac{1}{N}\mathrm{V}\left[\frac{f(X)}{p(X)}\right].
\end{aligned}$$

This immediately shows that the expected squared error of Monte Carlo integration reduces as $O(N^{-1})$ (thus the error reduces as $O(N^{-\frac{1}{2}})$). Note that the assumption that $\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}$ is independent is not correct unless the samples are drawn from the normal distribution. However, the central limit theorem states that the distribution of these average values converges to the normal distribution with $N\to\infty$, so the assumption is asymptotically true. This derivation also assumes that $V\left[\frac{f(X)}{p(X)}\right]$ is finite and exists. However, even if the variance is infinite, the strong law of large numbers tells us that

$$\lim_{N\to\infty}\text{Prob}\left[E\left[\frac{f(X)}{p(X)}\right]-\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}=0\right]=1 \qquad (2.15)$$

This result often referred as the proof of independence of the convergence rate of Monte Carlo integration from the number of dimensions. However, such argument is not entirely accurate. We should note that the constant factor $V\left[\frac{f(X)}{p(X)}\right]$ could reintroduce the dependence on the number of dimensions. For example, suppose that $f(x)$ is a binary function that returns 1 when x is in a unit hypersphere within the domain centered at $(0.5,\ldots,0.5)$, and 0 otherwise. Assuming $p(x)=1$, we have $V\left[\frac{f(X)}{p(X)}\right]=O(0.5^d)$. Therefore, the dependence on the number of dimensions is not always avoided even in Monte Carlo integration. However, a more sophisticated sample placement in Monte

Carlo integration might achieve $V\left[\frac{f(X)}{p(X)}\right] = O(1)$ if we have enough knowledge about the function $f(x)$. The independence of Monte Carlo integration from the number of dimensions is not an unconditional property.

Consistency and Unbiasedness of Estimates

A Monte Carlo estimator $f_N(X)$ is consistent if the estimate converges almost surly to the correct solution with an infinite number of samples. In the context of Monte Carlo integration, a consistent estimator satisfies

$$\lim_{N\to\infty} \text{Prob}\left[\int_0^1 f(x)\,dx - f_N(X) = 0\right] = 1, \tag{2.16}$$

where N is the number of samples.

As a related, yet separate, concept is unbiasedness. A Monte Carlo estimator $f_N(X)$ is unbiased if the expected value of the estimate is equal to the correct solution. To be concrete, an unbiased estimator satisfy

$$\int_0^1 f(x)\,dx - E[f_N(X)] = 0 \tag{2.17}$$

A *biased* estimator does not satisfy the above condition. Note that an unbiased estimator is unbiased regardless of the number of samples.

One common misconception is that all unbiased estimators are consistent. This is not true and the definitions of consistent/inconsistent and unbiased/biased are independent. For example, think about estimating the expected numbers obtained by throwing a regular cubic dice. We can obtain example estimators as follows.

- **Consistent and unbiased estimator**:
  The sum of numbers we obtained so far divided by the number of throws.
- **Inconsistent and unbiased estimator**:
  The number obtained from the first throw regardless of the rest of throws.
- **Consistent and unbiased estimator**:
  The sum of numbers we obtained so far divided by the number of throws + 1.
- **Inconsistent and biased estimator**:
  4, regardless of the throws.

Notice that there exists *inconsistent and unbiased estimator* even in this simple example. This estimator is unbiased since the expected value of the number obtained from the first throw is equal to the correct value of 3.5. However, this estimator is inconsistent because the estimate never converges to the correct value of 3.5 even with an infinite number of throws. One example estimator of this kind is Metropolis light transport with the start-up bias elimination [117]. Although the initial weighting of a Markov chain makes the process unbiased, since the weight is sampled once and is not resampled over the process, the result is unbiased, but inconsistent. Veach briefly touched this issue as "Since all subsequent samples use the same weight $W_i = W_0$, this would lead to a completely black final image" in 11.3.2 [117] (in practice, however, we do not see the problem as the weight is often close to one even across many different runs). The existence of this class of estimators is often ignored, so one should be careful not to immediately conclude that a new unbiased estimator is also consistent.

Another misconception is that unbiased estimators are *more accurate* than biased estimators. This is incorrect in two ways. First, as we mentioned above, just because an estimator is unbiased does guarantee that it is also consistent. If a biased estimator in comparison is consistent and an unbiased estimator is actually inconsistent, solutions obtained by the biased estimator can be closer to the correct solution as we take more samples. Second, an actual solution we obtain from an unbiased estimator is not the expected value itself, but a single probabilistic estimate of a random variable of which the expected value is equal to the correct solution. It is possible that a biased estimator gives us a solution closer to the correct solution with the same number of samples.

Some care also should be taken on consistency of estimators. Ensuring consistency is preferable in theory in order to claim the correctness of Monte Carlo estimates. However, one should notice that the condition for consistency is based on an infinite number of samples, which never happens in practice. It is thus again incorrect to claim that any consistent method is *more accurate* than inconsistent methods, unless we take an infinite number of samples.

The fact that one estimator is unbiased or consistent should not be taken as any evidence on the accuracy of its estimate with a finite number of samples. We emphasize that it is incorrect to make such a claim in general settings.

2.2.2 Irregular Sample Placement

The key advantage of Monte Carlo integration is that the placement of samples is very flexible. Samples can be placed even based on partial knowledge of the function that we integrate. In the following subsections, we review some common techniques that can improve the efficiency of Monte Carlo integration. We first review techniques that place samples with an irregular distribution.

Importance sampling

Recall that we can choose arbitrary probability density function in Monte Carlo integration (Equation 2.12). Recall also that error in Monte Carlo integration is proportional to the variance $V[f(x)/p(x)]$ (Equation 2.14). The idea of importance sampling is to choose $p(x)$ that reduces the variance $V[f(x)/p(x)]$, which in turn reduces error of Monte Carlo integration.

In fact, having $cp(x) = f(x)$ with a constant c gives us exactly *zero* error regardless of the number of samples since

$$E\left[\left(E\left[\frac{f(X)}{p(X)}\right] - \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)}\right)^2\right] = \frac{1}{N}V\left[\frac{f(X)}{p(X)}\right] = \frac{1}{N}V\left[\frac{1}{c}\right] = 0. \tag{2.18}$$

However, since any probability density functions should integrate to one, we need to know the proportionality constant c as

$$c = \frac{1}{\int_{\Omega} f(x)\,dx} \tag{2.19}$$

which unfortunately is the exact integral that we are trying to solve. Although finding such $p(x)$ is not plausible in general, the above equation provides the rule of thumb that the shape of $p(x)$ should be close to the shape of $f(x)$. Note however that we can also make the variance $V[f(x)/p(x)]$ arbitrary large [92], thus we should carefully choose $p(x)$. When we have multiple choices of $p(x)$, we could use multiple importance sampling [118] to combine the result from each probability density function.

Adaptive sampling

Instead of trying to find a good probability density function that reduces error, adaptive sampling uses the statistical analysis of partial information of $f(x)$ obtained through previous samples in order to generate a new sample. There are many forms of adaptive sampling (for a comprehensive review in computer graphics, one may refer to the book by Glassner [34]), but one common form is a two stage algorithm, where the first stage blindly takes pilot samples over the integration domain and the second stage constructs a guess of the shape of $f(x)$ from the pilot samples in order to drive importance sampling based on the guess.

2.2.3 Uniform Sample Placement

Since we can only take a finite number of samples in practice, it is possible (though less likely) that all the samples land very close to each other. This is undesirable since we would like samples to cover the entire domain of integration. Samples will eventually cover the entire domain if we take an infinite number of samples, but we would like a finite number of samples to distribute as uniformly as possible. We describe two techniques that improves distribution of samples. We consider the case $p(x) = 1$ in this section (i.e., uniform sampling), but the same concept is applicable to any $p(x)$.

Stratified sampling

Stratified sampling breaks the domain into strata, and place an equal number of samples to each stratum. As long as strata are well-shaped, all the samples cannot land within a small region. Suppose that we divide the integration domain into $\frac{N}{n}$ uniformly-sized strata D_i with n samples in each stratum, Monte Carlo integration using stratified sampling is given as

$$E\left[\frac{f(X)}{p(X)}\right] \approx \frac{1}{N}\sum_{i=1}^{\frac{N}{n}}\sum_{i=1}^{n}\frac{f(x_i)}{p(x_i)}. \tag{2.20}$$

If we define V_{D_i} as the variance of $\frac{f(x_i)}{p(x_i)}$ within each stratum, the variance of the above estimate is

$$\frac{1}{N}\sum_{i=1}^{\frac{N}{n}} V_{D_i}\left[\frac{f(X)}{p(X)}\right] \leq \frac{1}{N} V\left[\frac{f(X)}{p(X)}\right]. \tag{2.21}$$

The variance of the estimate using stratified sampling can be smaller if the sum of the variance of each stratum V_{D_i} is smaller than the variance of the entire integration domain.

In particular, if $\frac{f(X)}{p(X)}$ is constant within each stratum, the estimate using stratified sampling has zero error. It is also possible to combine adaptive sampling and stratified sampling such that the shapes of strata depend on samples taken so far. VEGAS [79] is one such algorithm that is particularly efficient for 1D-separable functions.

Quasi-Monte Carlo methods

Stratified sampling can be quite costly in high dimensional space because of the book keeping of strata. Quasi-Monte Carlo methods modifies random number generators such that generated samples are less clumped by giving up the randomness of samples. One common approach to generate such samples is based on a low-discrepancy sequence such as Halton sequence, Sobol sequence, and Hammersley sequence. A comprehensive survey in the context of light transport simulation is given by Owen [91]. One such application is pioneered by Keller [68].

2.2.4 Correlated Sampling

Samples in Monte Carlo integration methods are usually not correlated. There are however a few techniques that intentionally introduce correlation between samples, such that additional *covariance* of samples works to reduce error of Monte Carlo integration. Similar to importance sampling, this technique can increase error of Monte Carlo integration if not carefully used.

Control variates

Suppose that we have function $g(x)$ with known integral $G_p = \int_\Omega g(x)\,dx$, control variates use the following estimation to perform Monte Carlo integration of $f(x)$:

$$\begin{aligned}\int_0^1 f(x)\,dx &= \int_0^1 f(x) - g(x) + g(x)\,dx \\ &= \int_0^1 f(x) - g(x)\,dx + G \\ &\approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i) - g(x_i)}{p(x_i)} + G.\end{aligned} \tag{2.22}$$

The variance of the estimate is given as

$$V\left[\frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)-g(x_i)}{p(x_i)}+G\right]=\frac{1}{N}\left(V\left[\frac{f(X)}{p(X)}\right]+V\left[\frac{g(X)}{p(X)}\right]-2\mathrm{Cov}\left[\frac{f(X)}{p(X)},\frac{g(X)}{p(X)}\right]\right). \tag{2.23}$$

where Cov $\left[\frac{f(X)}{p(X)},\frac{g(X)}{p(X)}\right]$ is covariance of $\frac{f(X)}{p(X)}$ and $\frac{g(X)}{p(X)}$. We can think that samples for $f(x)$ and $g(x)$ are correlated. In order that this estimation reduces the variance, $f(x)$ and $g(x)$ should be at least positively correlated.

Antithetic variates

Another technique that uses correlated samples is antithetic variates. We consider only the original function $f(x)$ but take multiple correlated samples such that correlation reduces variance of combined estimates. For example, suppose that we generate a pair of correlated samples x_i and y_i, Monte Carlo integration of $f(x)$ using antithetic variates is

$$\int_0^1 f(x)\approx\frac{1}{N}\sum_{i=1}^{N}\frac{\frac{f(x_i)}{p(x_i)}+\frac{f(y_i)}{p(y_i)}}{2}. \tag{2.24}$$

The variance of the estimate is given as

$$V\left[\frac{1}{N}\sum_{i=1}^{\frac{N}{2}}\left(\frac{f(x_i)}{p(x_i)}+\frac{f(y_i)}{p(y_i)}\right)\right]=\frac{1}{4N}\left(V\left[\frac{f(X)}{p(X)}\right]+V\left[\frac{f(Y)}{p(Y)}\right]+2\mathrm{Cov}\left[\frac{f(X)}{p(X)},\frac{f(Y)}{p(Y)}\right]\right). \tag{2.25}$$

Note that if x_i and y_i are independent, then the estimate becomes exactly equal to the estimate on using regular Monte Carlo integration. In order that this estimation reduces the variance, $\frac{f(x_i)}{p(x_i)}$ and $\frac{f(y_i)}{p(y_i)}$ should be at least negatively correlated. If $\frac{f(x)}{p(x)}$ is a monotonic function over the domain, one of the easiest choice of such y_i is $y_i = 1 - x_i$.

2.3 Sample Generation

The previous section does not discuss how to generate samples from a given probability density function $p(x)$. Generating samples is in fact not a trivial problem, especially if we do not have much knowledge on $p(x)$. This section review some sample generation methods.

2.3.1 Inversion Method

One easiest method for generating samples according $p(x)$ is to use the inverse function of its cumulative distribution function $P^-1(x)$. This method first generates a random number μ that is uniformly distributed within the interval $\mu \in [0,1]$, and takes $P^-1(\mu) = x$ as the random sample from the probability density distribution $p(x)$. It is easy to see this method is valid since

$$\text{Prob}(P^-1(\mu) \leq x) = \text{Prob}(\mu \leq P(x)) = P(x). \tag{2.26}$$

This method is useful if the inverse function of the cumulative distribution function $P^-1(x)$ is available either symbolically or numerically. One example of this method is the Box-Muller transform for generating samples from the normal distribution [10].

2.3.2 Rejection Sampling

Suppose that we can generate samples according to the probability density function $q(x)$, but not based on $p(x)$. If we can find a constant M where $p(x) < Mq(x)$ for all x, then we can generate samples according to $p(x)$ by rejection sampling. The method first generates a sample x according to $q(x)$ and additionally generates another sample μ from the uniform distribution, and repeats the process until $\frac{p(x)}{Mq(x)} > \mu$.

The validation of the method can be shown using Bayes' theorem. We define $\text{Prob}(x|a)$ as the probability of generating a sample x given that x is not rejected, $\text{Prob}(a|x)$ as the probability of accepting x given the generated sample x. We would like to know if $\text{Prob}(x|a) = p(x)$. From the algorithm, we can observe that

$$\text{Prob}(a|x) = \frac{p(x)}{Mq(x)} 2\text{EM}\,\text{Prob}(x) = q(x), \tag{2.27}$$

$$\text{Prob}(a) = \int_x \text{Prob}(a|x)\,\text{Prob}(x)\,dx = \int_x \frac{p(x)}{M}\,dx = \frac{1}{M}. \tag{2.28}$$

Applying the Bayes' theorem, we obtain

$$\text{Prob}(x|a) = \frac{\text{Prob}(a|x)\,\text{Prob}(x)}{\text{Prob}(a)} = \frac{\frac{p(x)}{Mq(x)}q(x)}{\frac{1}{M}} = p(x) \tag{2.29}$$

as we wanted.

Another useful view of rejection sampling is that the method generates samples under the area below $Mq(x)$ and accept only samples that are below $p(x)$. If the envelope $Mq(x)$ is closer to $p(x)$, less samples will be rejected. Note that no rejection will occur only if $Mq(x) = p(x)$, which defeats the purpose of using rejection sampling to begin with.

2.3.3 Markov Chain Monte Carlo Sampling

The idea of generating samples within the envelope and rejecting samples outside is related to Markov chain Monte Carlo sampling. Since Markov chain Monte Carlo sampling in general covers many different algorithms, this section provides only the overall idea. We can think of the key idea as another form of rejection sampling. The idea is that, instead of generating samples from the envelope $Mq(x)$, we use a random walk that is crawling under the area defined by $p(x)$ and reject a random walk to obtain the sample distribution $p(x)$.

One popular approach is the Metropolis-Hastings algorithm. Suppose that the current sample (or the state of the random walk) is x_t. The algorithm uses the proposal density function $Q(a|b)$ which describes the probability density function of new samples a given the current sample b. Using this proposal density function, the algorithm accept a newly generated sample only if another random number $\mu \in [0,1]$ from the uniform distribution satisfies

$$\mu < \frac{p(x_p)Q(x_t|x_p)}{p(x_t)Q(x_p|x_t)} \tag{2.30}$$

where x_p is the newly generated sample. If this new sample is accepted, the next sample is $x_{t+1} = x_p$, and otherwise $x_{t+1} = x_t$. Samples in the history of this random walk are distributed according to $p(x)$ for a large enough t (or in general $t \to \infty$).

Markov chain Monte Carlo sampling is a very strong approach as it only needs the ability to evaluate $p(x)$ up to a constant scaling factor. There are many variations of this basic algorithm. For the direction of improving proposal distributions, the Multiple-try algorithm [80] proposed an extension to include multiple proposals at the same time, instead of a single proposal. Gibbs sampling uses (described in the book by Geman and Geman [33]) importance sampling along an axis instead of the random proposal. Slice sampling [87] uses rejection sampling along the axis of the function value, which can be seen as a mixture of rejection sampling and Markov Chain Monte Carlo sampling. Hybrid Monte Carlo [21] uses gradients of the function in order to incorporate a symplectic integrator to generate a proposal along the same magnitude of the function value, and another proposal is used for generating a Markov chain along a different function value.

Another direction is to modify the target function $p(x)$ in some ways. Ensemble Monte Carlo methods such as parallel tempering or replica exchange [111] extend the function space by an auxiliary variable in order to reduce correlation of samples. Umbrella sampling [116] weight the function according to the histogram such that the tail of the distribution gets better sample distribution. If we look at approaches that uses multiple Markov chains, population Monte Carlo methods (also known as particle filters and sequential Monte Carlo) [38] uses multiple chains that interact with each other to propagate information from one chain to another, and perfect sampling [81] proposes an algorithm that can completely get rid of correction of samples at the cost of running multiple chains until they match the state to obtain a single sample.

The method has been successfully applied to many problems including the light transport problem [15, 103, 119]. One general issue however is that samples are now correlated due to the random walk process. Another issue is that in general there is no way to know if samples are really distributed according $p(x)$ with a finite number of steps. Both of the issues are largely affected by the proposal density function, which is usually problem dependent. Nevertheless, Markov Chain Monte Carlo sampling is often the only reasonable approach to generate samples from an unknown arbitrary function $p(x)$ since both the inverse method and rejection sampling uses some knowledge on $p(x)$. In this dissertation, Chapter 8 goes into more details of the our new application.

Chapter 3

Density Estimation

The basis of this dissertation work is *density estimation*, which is a problem of constructing an unknown probability distribution of finite observed samples. Density estimation has many applications over very different topics. For example, a teacher might be interested in the distribution of scores given a set of scores of an exam, which is equal to estimate density of scores. A survey of a geographical distribution of a particular animal species often involves density estimation over a surface. In this dissertation, we use density estimation to determine the intensity of illumination given the distribution of photons. This chapter introduces some basic concepts of density estimation and also reviews some existing methods.

3.1 Formulation

Density estimation is a problem of constructing an unknown probability density function $p(x)$ from observed samples that are generated according to $p(x)$. One approach for density estimation is a *parametric* approach, where we assume that samples are drawn from a known family of probability density functions. For example, we could assume that samples are drawn from the normal distributions. In this case, the parametric approach estimates the parameters of the normal distributions (i.e., mean and variance).

In this dissertation, we instead focus on a *nonparametric* approach which does not restrict the estimated probability density function to be a given family of probability density functions. The application of density estimation to the light transport problem has been explored in computer graphics [62, 104, 121, 124], and we will discuss more details on how this dissertation work differentiates from the existing work in the following chapters.

We can formulate many density estimation methods as a sum of weight functions placed at each sample position [131]. Consider a function $w(x,y)$ that satisfies the following two conditions;

$$\int_{-\infty}^{\infty} w(x,y)\, dy = 1 \tag{3.1}$$

and

$$w(x,y) \geq 0 \quad \forall\, x,y. \tag{3.2}$$

Given N samples $x_1, x_2, \ldots, x_N$, an estimate of the probability density may be obtained by

$$\hat{p}(x) = \frac{1}{N} \sum_{i=1}^{N} w_i(x_i, x). \tag{3.3}$$

Note that the weight function can vary across samples. This general formulation, commonly called a *δ-sequence density estimator*, can express all the methods we describe in the next section.

It is easy to confirm that the estimate is a probability density function regardless of samples since

$$\int_{-\infty}^{\infty} \hat{p}(x)\, dx = \int_{-\infty}^{\infty} \frac{1}{N} \sum_{i=1}^{N} w_i(x_i, x)\, dx = \frac{1}{N} \sum_{i=1}^{N} \int_{-\infty}^{\infty} w_i(x_i, x)\, dx = 1. \tag{3.4}$$

We can also think of density estimation as a result of spatially varying filtering over the correct function $p(x)$ since x_i are distributed according to $p(x)$:

$$E[\hat{p}(x)] = \frac{1}{N}\sum_{i=1}^{N} E[w_i(x_i,x)] = \int w_t(t,x)p(t)\,dt. \tag{3.5}$$

This alternative view is particularly useful when for analyzing asymptotic properties. If we consider the function $w'_x(t) = w_t(t,x)p(t)$, the above expression also tells us that density estimation is essentially equivalent to Monte Carlo integration of this combined function $w'_x(t)$ at x.

3.2 Methods

This section reviews some common methods for nonparametric density estimation. This section by no means an extensive survey of all the available density estimation methods. Although it is classical, the book by Silverman [105] covers major density estimation methods that are still in use today. A more recent extensive survey was done by Izenman [58].

3.2.1 Histograms

The histogram method is the oldest and still widely used density estimation method. Given the position X that we would like to estimate density, we define an interval $X \in [X_a, X_b]$, then the estimate is given as

$$\hat{p}_{\text{histo}}(X) = \frac{|x_i \in [X_a, X_b]|}{N(X_b - X_a)}, \tag{3.6}$$

where $|x_i \in [X_a, X_b]|$ is the number of samples that lands within $[X_a, X_b]$. In its simplest form, we can break the entire domain of interest into uniform sized intervals, and simply count the number of samples within each interval to construct the estimate. Note that the above definition does not require intervals to be non-overlapping.

By definition, the estimate using histograms is not a continuous function. Instead, $\hat{p}_{\text{histo}}(X)$ as discontinuities at each end of the intervals, and has zero derivatives everywhere. This may or may not be an issue depending on the application, however, the following methods will remove this limitation.

The histogram method also has the asymptotic mean-squared-error convergence rate of $O(N^{-\frac{2}{3}})$ at most [102], which is worse than $O(N^{-\frac{4}{5}})$ of the best possible kernel estimator in the next section [31]. This give another, somewhat theoretical, reason why one wants to use another density estimator.

3.2.2 Kernel Estimator

The idea of the kernel estimator (or kernel density estimation) is that, instead of simply counting the number of samples within each interval as in the histogram method, we weight the contribution of each sample by a *kernel* function around each sample point [11, 93]. Kernel functions $K(t)$ satisfy

$$\int_{-\infty}^{\infty} K(t)\, dt = 1 \tag{3.7}$$

and the estimate is

$$\hat{p}_{\text{kernel}}(X) = \frac{1}{N}\sum_{i=1}^{N} K_i(X - x_i), \tag{3.8}$$

where

$$K_i(t) = \frac{K\left(\frac{t}{h_i}\right)}{h_i}. \tag{3.9}$$

The parameter h_i is often called the *window width*, *bandwidth*, or the *smoothing parameter*, which decides 'smoothness' of the estimate. If a kernel function is k times differentiable, the resulting estimate is also k times differentiable. Therefore, we can obtain a continuous probability density function using the kernel estimator.

The kernel density estimator with nonadaptive, nonnegative kernels has the asymptotic mean-squared-error convergence rate of $O(N^{-\frac{4}{5}})$ [31]. Although there are many choices of the kernel, error of the estimate is insensitive to the choice of the kernel [83].

Recall that the MSE in density estimation depends on both on the weighting functions (in the kernel estimator, the kernel function) and the number of samples. Therefore, different kernel functions can give us different MSE. It is however considered difficult to automatically select the optimal kernel and its parameters in general cases since we will need to quantify error only using a finite set of samples [65]. At the moment, there is no such method that works robustly for a small number of samples.

One methods that tries to adapt the amount smoothing based on the local density of samples is the *kNN estimator* [28]. In the kNN estimator, instead of specifying the smoothing parameter, we specify the number of nearest neighbors around each estimation point k. The kernel function in the kNN estimator is defined as

$$K_{\text{kNN}}(t) = \frac{K\left(\frac{t}{d(t,x_k)}\right)}{d(t,x_k)}. \tag{3.10}$$

where $d(t,x_k)$ is the distance between t and its k-th nearest neighbor sample x_k. As we will discuss later, the kNN estimator has successfully been used in the applications for the light transport problem [61, 62, 63].

3.2.3 Orthogonal Series Estimator

The orthogonal series estimator uses projections of samples on a set of orthogonal basis functions [13]. The estimate is a weighted sum of the orthogonal basis where we compute each weight using the projection. In order to use the orthogonal series estimator, we expand a probability density function using a orthogonal series $\phi_{b,j}(x)$;

$$p(X) = \sum_{j=1}^{\infty} w_{b,j}\phi_{b,j}(X) \tag{3.11}$$

where $w_{b,j}$ is defined as

$$w_{b,j} = \int_{-\infty}^{\infty} p(x)\phi_{b,j}(x)\, dx \tag{3.12}$$

which can be estimated using Monte Carlo integration using samples x_i.

Note that we often need to truncate the orthogonal series since the expansion of general $p(x)$ will result in an infinite series (e.g., Fourier series), thus the estimate is

$$\hat{p}_{\text{ortho}}(X) = \sum_{j=1}^{N} w_{b,j}\phi_{b,j}(X). \tag{3.13}$$

One issue of this approach is that the resulting estimate cannot be guaranteed to be non-negative in general. Since all probability density functions are non-negative by definition, this issue may be undesirable in some applications including the light transport problem. Note also that the method is not necessarily free from the smoothing parameters. Choosing the number of terms in the estimate poses essentially the same challenges as choosing the smoothing parameters.

3.2.4 Maximum Likelihood Estimator

A general view of density estimation is to find a function $\hat{p}(x)$ that approximates $p(x)$ from given samples. In order to find such a function, we need some measure of how close the approximation $\hat{p}(x)$ is to the correct solution $p(x)$ using samples. One common measure is the log likelihood function [27];

$$L(\hat{p}(x)) = \sum_{j=1}^{N} \log \hat{p}(x_i). \tag{3.14}$$

The problem is then to find a function $\hat{p}(x)$ (subject to the condition that the function being in a specific class of functions) that maximizes the likelihood function, which is called the maximum likelihood problem.

The idea of the maximum likelihood estimator is to use a solution (not necessarily unique) to this problem. Note however that we need to put some restrictions (indeed "smoothing" conditions) in order that the solution to make sense. This is because one can attain the maximum likelihood by placing the delta functions at sample locations, which is unlikely to be equal to the correct solution $p(x)$. There are many variations of such restrictions, but one popular way is to "penalize roughness" of the function by using the Φ-penalized log likelihood function [36];

$$\hat{L}(\hat{p}(x)) = \sum_{j=1}^{N} \log \hat{p}(x_i) - \Phi(\hat{p}(x_i)) \tag{3.15}$$

where Φ is the penalty function.

One should be cautious that the log likelihood function (either penalized or non-penalized) is another heuristic measure of the accuracy of $\hat{p}(x)$ except for some simple cases, especially for a finite number of samples and general functions. The histogram estimator indeed gives us a unique solution to this problem where the class of functions is a sum of rectangular functions [19], but even the uniqueness of the solution is not the case in general. For example, a common model of Gaussian mixtures indeed poses those issues in theory. However, many applications found that the maximum likelihood estimator work well in practice. A recent application of this class of estimators to light transport simulation demonstrates some benefits of using this approach [59].

3.3 Properties of the Error

Since we only observe a finite number of samples in practice, an estimated probability density function usually differs from the correct probability density function. In this section, we review some important properties of error in density estimation.

3.3.1 Bias and Variance

There are various ways to quantify error in density estimation. One popular approach is to consider the mean square error (MSE) at a single point. Given the estimated probability density function $\hat{p}(x)$ and the correct probability density function $p(x)$, the MSE is given as

$$\mathrm{MSE}(\hat{p}(x)) = E[p(x) - \hat{p}(x)]^2 = (E[\hat{p}(x)] - p(x))^2 + V[\hat{p}(x)]. \tag{3.16}$$

Since $E[\hat{p}(x)] - p(x)$ is bias, the MSE is simply described as a sum of squared bias and variance. Note that the MSE is again given as expected error over possible sets of samples (i.e., average error over different random number sequences). This is because the estimated probability density function $\hat{p}(x)$ is stochastic similar estimates in Monte Carlo integration.

3.3.2 Expected Mean Square Error

Although it is usually not possible to exactly quantify error in density estimation, we can analyze the expected behavior of the error using the generalized formulation above. To be concrete, the bias term in the MSE can be expanded as

$$(E[\hat{p}(x)] - p(x))^2 = \left(\int w_t(t,x) p(t)\, dt - p(x) \right)^2 \tag{3.17}$$

and the variance term in the MSE can be expanded as

$$V[\hat{p}(x)] = \frac{1}{N^2} \sum_{i=1}^{N} \left(\int w_i(t,x)^2 p(t)\, dt - \left(\int w_i(t,x) p(t)\, dt \right)^2 \right). \tag{3.18}$$

The result for the bias term tells us that just increasing the number of samples N does not decrease bias, and the result for the variance term tells us that variance depends on both the number of samples and the weight functions $w_i(x,y)$. Similar to Monte Carlo integration, density estimation is said to be unbiased if the bias term is zero, and consistent if the MSE almost surely converges to zero as we increase the number of samples.

3.3.3 Asymptotic Properties of Error

In order to further analyze properties of error in density estimation, we restrict ourself for the kernel estimator on 1D space using a single kernel. Under this setting and $y = x - ht$, we can expand the bias term as [105]

$$\begin{aligned}
E[\hat{p}(x)] - p(x) &= \int w_y(y,x)p(y)\,dt - p(x) \\
&= \int K(t)p(x-ht)\,dt - p(x) \\
&= \int K(t)(p(x-ht) - p(x))\,dt \\
&\approx \int K(t)\left(p(x) - ht\frac{dp}{dx}(x) + \frac{1}{2}h^2t^2\frac{d^2p}{dx^2}(x) - p(x)\right)\,dt \\
&= \frac{1}{2}k_2h^2\frac{d^2p}{dx^2}(x).
\end{aligned} \tag{3.19}$$

Similarly, we can expand the variance term for large N and small h as [105]

$$\begin{aligned}
V[\hat{p}(x)] &= \frac{1}{N}\left(\int\left(\frac{K\left(\frac{x-y}{h}\right)}{h}\right)^2 p(y)\,dy - \left(\int\frac{K\left(\frac{x-y}{h}\right)}{h}p(y)\,dy\right)^2\right) \\
&= \frac{1}{N}\left(\frac{1}{h}\int K(t)^2p(x-ht)\,dt - (p(x) + (E[\hat{p}(x)] - p(x)))^2\right) \\
&\approx \frac{1}{N}\left(\frac{1}{h}\int K(t)^2p(x-ht)\,dt - \left(p(x) + \frac{1}{2}k_2h^2\frac{d^2p}{dx^2}(x)\right)^2\right) \\
&\approx \frac{1}{N}\left(\frac{1}{h}\int K(t)^2\left(p(x) - ht\frac{dp}{dx}(x) + \frac{1}{2}h^2t^2\frac{d^2p}{dx^2}(x)\right)\,dt\right) + O\left(\frac{1}{N}\right) \\
&\approx \frac{1}{Nh}\int K(t)^2p(x)\,dt + O(h) + O\left(\frac{1}{N}\right) \\
&\approx \frac{k_3}{Nh}p(x).
\end{aligned} \tag{3.20}$$

These derivations assume that the kernel function satisfies

$$\begin{aligned} \int K(t)\,dt = 1 \qquad & \int tK(t)\,dt = 0 \\ \int t^2K(t)\,dt = k_2 \neq 0 \qquad & \int K(t)^2\,dt = k_3 \neq 0 \end{aligned} \tag{3.21}$$

which is true for a radially symmetric kernel function. Combining these results together, the MSE in density estimation is

$$\mathrm{MSE}(\hat{p}(x)) \approx \frac{k_3}{Nh}p(x) + \frac{1}{4}k_2^2h^4\left(\frac{d^2p}{dx^2}(x)\right)^2. \tag{3.22}$$

Notice again that the error due to the bias term does not decrease by simply increasing the number of samples N. Furthermore, this derivation reveals that the variance term asymptotically reduces according to the reciprocal of the product of the number of samples N and the smoothing parameter h. This derivation also suggests that there is a trade-off between bias and variance for the smoothing parameter h. Reducing h results in reducing bias, but increasing variance in the MSE. Conversely, increasing h results in reducing variance, but increasing bias in the MSE. This is effect is commonly called *bias-variance* trade-off.

3.3.4 Conditions of Consistency

The asymptotic behavior of the MSE leads to the following important condition that ensures the convergence of density estimation:

$$\lim_{N\to\infty} \mathrm{MSE}(\hat{p}(x)) = 0 \tag{3.23}$$

if

$$\lim_{N\to\infty} h = 0 \qquad \lim_{N\to\infty} Nh = \infty \tag{3.24}$$

are satisfied provided that $p(x)$ is continuous at x [93]. In other words, under these conditions, the estimate $\hat{p}(x)$ exhibits a *strong point-wise convergence* for $p(x)$. This result has been mainly used purely for theoretical analysis [30] or guiding the development of automatic smoothing parameter selection algorithms [2]. One contribution of this dissertation work is a new desity estimator that incorporates these conditions into the algorithm itself.

Chapter 4

Light Transport Theory

This chapter reviews the foundations of light transport theory which are basis for many light transport simulation algorithms. Although physics of light is well understood throughout history, there are some specific assumptions made in computer graphics in order to make simulation of light transport computationally tractable. In this chapter, the first few sections define the terminology that are necessarily to describe the properties of light, and develop the equations that govern light transport. The succeeding section briefly summarizes how we can perform light transport simulation based on the equations in different ways.

4.1 Assumptions in Simulation

In modern physics, light is understood as a packet of energy that exhibits both wave and particle properties (refer to the book by Saleh and Teich [100] for example). This view is based on quantum physics, and the study of light using this view is called *quantum optics*. Quantum optics can model all the light transport phenomena that we can observe in nature. Although it is tempting to use quantum optics to light transport simulation, this level of physical accuracy is usually not necessary in computer graphics. Instead, computer graphics typically uses *geometric optics*.

Geometric optics describes light as a particle which travels along a straight line. Using geometric optics, we can describe emission, absorption, reflection, and refraction of light. In computer graphics, we ignore temporal behavior of light, which leads to a common abstraction of light as a ray. Geometric optics provides an excellent approximation of light transport in the real world when the wavelength of light is significantly smaller than the scale of effects that we would like to simulate. This assumption is true in many interesting cases as wavelength of visible light is in the order of 10^{-7} m and objects that we see are typically in 10^{-1} m to 10^{1} m.

Although geometric optics ignores wave properties of light, it can capture many phenomena of light that are visually important for generating photorealistic images using light transport simulation. In addition, it is indeed possible to simulate some phenomena such as polarization and interference that are described by wave optics using an light transport simulation algorithm based on geometric optics [35, 113]. It is wrong to claim that light transport simulation based on geometric optics cannot capture such phenomena.

4.2 Radiometry

This section provides a brief overview of the radiometric units that describe physical quantities of light. Note that, there are also relevant units called *photometric* units that deals with the perception of light energy to a human observer. This dissertation focuses only on radiometric quantities since photometric quantities can be derived from corresponding radiometric quantities.

4.2.1 Background

Before going into the definitions of radiometric units, we will first review some mathematical concepts used throughout their definitions. We first define a space of normalized vectors for given another vector n as

$$\Omega_+ = \{\vec{\omega} : |\vec{\omega}| = 1, \vec{\omega} \cdot \vec{n} \geq 0\}, \tag{4.1}$$

which is basically a set of directions over the upper hemisphere around n. Similarly, we can define another space for the lower hemisphere

$$\Omega_- = \{\vec{\omega} : |\vec{\omega}| = 1, \vec{\omega} \cdot \vec{n} \leq 0\}. \tag{4.2}$$

In computer graphics, we conventionally use $\Omega = \Omega_+$ and this dissertation also follows this convention. We also use A to denote a set of all the points on the input geometry.

Solid Angle

Since light travels in a 3D space, it is useful to have a notation of angle in a 3D space. The solid angle $\Delta\Omega$ serves exactly this purpose, which is an extension of conventional angle on a 2D Euclidean space to the surface of a unit sphere. Suppose that we use $\vec{\omega} = (\theta, \phi)$ to denote longitudinal and azimuthal angles of the vector originating from the center of the sphere to any point, the solid angle of a surface S at the center of the sphere is defined as a finite integral

$$\Delta\Omega = \iint_S |\sin\theta| \, d\theta \, d\phi = \iint_S d\Omega \tag{4.3}$$

where $d\Omega = |\sin\theta| \, d\theta \, d\phi$ is a differential solid angle. The solid angle is basically the projected surface area of S onto the unit sphere. The solid angle subtended by S that covers all the directions is 4π. Similarly, the solid angle subtended by Ω_+ and Ω_- are 2π.

The projected solid angle, $\Delta\Omega_{\vec{n}}$, is the solid angle around a small surface oriented toward $\vec{n}$ ($|\vec{n}| = 1$);

$$\Delta\Omega_{\vec{n}} = \iint_S |\vec{\omega} \cdot \vec{n}| \, d\Omega \tag{4.4}$$

4.2.2 Radiometric Quantities

Radiometric quantities describe energy flow due to light. Depending on the measurement unit of energy, there are four quantities that are commonly used in computer graphics.

Flux

Flux (or radiant power) is the fundamental quantity that express total energy flow per unit time. It is usually denoted as Φ and this dissertation follows the same convention. The unit is watts (W). Note that flux does not specify any configurations of the source of energy of the receiver. Flux simply states how much energy flows over all. An example of usages is to specify brightness of a light source in flux. For example, if we have a light bulb with a power rating of 100 W, if there is no loss of energy, its flux is 100 W.

Radiant Intensity

Radiant intensity, I, is flux per unit solid angle defined as

$$I = \frac{d\Phi}{d\Omega}. \tag{4.5}$$

This notation may be confusing at a glance since it looks like flux Φ is a function of solid angle, which does not sound immediately consistent with the definition of flux (i.e., flux just denotes total energy flow). The key is to consider radiant intensity as a limit of the ratio of flux flowing within a small solid angle and the solid angle itself. Using this view, it is easy to see that

$$\Phi = \int_\Omega I \, d\Omega = \int_\Omega \frac{d\Phi}{d\Omega} \, d\Omega. \tag{4.6}$$

In computer graphics, radiant intensity is used to specify the intensity of light source that is infinitely far away from the receiver. Such a light source gives us a moderately good approximation of sunlight. Note that radiant intensity can be used for describing both energy arriving to a receiver or energy emanating from a light source. For example, if we have a light source with 100 W that emits its energy uniformly over all directions, its radiant intensity is $\frac{100}{4\pi}$ W/s

Irradiance

Irradiance, E, is similar to radiant intensity, but is defined as flux per unit area;

$$E = \frac{d\Phi}{dA}. \tag{4.7}$$

Irradiance only describes energy *arriving* to the receiver. Energy emitted from a light source is instead called radiant exitance (or radiosity). For example, if a surface with an area of 5 m^2 uniformly receives 100 W of flux, irradiance is 20 W/m^2. It is again important to realize that, while irradiance can be a function over a receiver, flux usually is just a single value that describes total energy flow. Flux is an integral of irradiance (or radiant intensity) over a surface;

$$\Phi = \int_A E\, dA = \int_A \frac{d\Phi}{dA}\, dA. \tag{4.8}$$

In computer graphics, it is common to use a point light source to approximate a light source with small geometric size. Note that, although we can specify brightness of a point light source either by radiant intensity or flux, we cannot use irradiance for this purpose. This is because irradiance is infinity for a point light source.

Radiance

Radiance, L, is flux per unit projected area per unit solid angle;

$$L = \frac{d^2\Phi}{d\Omega\, dA_{\vec{n}}}, \tag{4.9}$$

where $dA_{\vec{n}}$ is projected differential area. Radiance is the important quantity that basically captures what we "see" as light. For example, the response of sensors (either cameras or the human eye) is proportional to radiance.

Radiance is invariant along straight paths regardless of distance, and this is the reason why apparent brightness of objects does not change by distance. For example, suppose that we see a monitor that emits 100 W of flux in total. If we get far away from the monitor, flux (total energy), from the monitor to our eye decreases. However, at the same time, the solid angle subtended by the monitor reduces as we getting far away, which exactly cancels out reduction of flux and keeping radiance constant.

Some relationships to other radiometric quantities are as follows

$$\begin{aligned} L &= \frac{dE}{d\Omega_{\vec{n}}} = \frac{dI}{dA_{\vec{n}}} \\ \Phi &= \int_A \int_\Omega L \, d\Omega_{\vec{n}} \, dA \\ E &= \int_\Omega L \, d\Omega_{\vec{n}}. \end{aligned} \tag{4.10}$$

Again, notation like $\frac{dE}{d\Omega_{\vec{n}}}$ is confusing at a glance, but it is just a limit of the ratio of a fraction of irradiance that comes through a small solid angle and the (projected) solid angle, not meaning irradiance itself is a function over directions.

The reason that we use the projected area instead the area in the definition of radiance is that we basically would like to measure a fraction of flux arriving at the receiver [22]. Expressing differential flux using radiance makes this point clear;

$$d\Phi = L \, d\Omega \, dA_{\vec{n}}. \tag{4.11}$$

For a large inclination angle (therefore, smaller projected area), apparent geometric area of the receiver shrinks and the receiver gets less flux. The cosine term accounts for this reduction in apparent geometric area due to the orientation of the receiver.

Since radiance is a function of direction and position, we use $L(\vec{x} \to \vec{\omega})$ to express radiance leaving from $\vec{x}$ to the direction $\vec{\omega}$ and $L(\vec{x} \leftarrow \vec{\omega})$ to express radiance coming toward $\vec{x}$ from the direction $\vec{\omega}$.

4.3 Interaction of Light with Surfaces

Bidirectional Scattering Distribution Functions

Since light transport involves interactions of light and surfaces, we need to describe how radiometric quantities of light change when the light interacts with a surface. This interaction depends on optical properties of the surface. In computer graphics, this interaction is described by a *bidirectional scattering distribution function* (commonly abbreviated as BSDF) defined as

$$f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) = \frac{dL(\vec{x} \to \vec{\omega}_o)}{dE(\vec{x} \leftarrow \vec{\omega}_i)} = \frac{dL(\vec{x} \to \vec{\omega}_o)}{L(\vec{x} \leftarrow \vec{\omega}_i) \cos(\vec{n}, \vec{\omega}_i)} \, d\omega_i. \tag{4.12}$$

Using the BRDF, the reflected radiance can be expressed as:

$$\begin{aligned} dL(\vec{x} \to \vec{\omega}_o) &= f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)\, dE(\vec{x} \leftarrow \vec{\omega}_i) \\ L(\vec{x} \to \vec{\omega}_o) &= \int_\Omega dL(\vec{x} \to \vec{\omega}_o) \\ &= \int_\Omega f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)\, dE(\vec{x} \leftarrow \vec{\omega}_i) \\ &= \int_\Omega f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) L(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}}. \end{aligned} \tag{4.13}$$

Conditions for BRDFs

It is important to note that the BRDF is not the reflectance which expresses the percentage of energy reflected by the material. As opposed to the reflectance, which value is in $[0,1]$, the BRDF can take any positive value. In fact, the reflectance R is defined as

$$R = \frac{1}{\pi} \iint_\Omega f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)\, d\omega_{o\vec{n}}\, d\omega_{i\vec{n}}. \tag{4.14}$$

Since the reflectance should be $R \in [0,1]$ by definition, we can see that

$$\int_\Omega f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o)\, d\omega_{o\vec{n}} \le 1. \tag{4.15}$$

This is the necessary and sufficient condition that the BRDF satisfies the law of conservation of energy.

Helmholtz reciprocity states that the direction of a light path can be reversed without affecting its throughput [52, 109]. For the BRDF, this translates into the following condition:

$$f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}_o) = f_r(\vec{x}, \vec{\omega}_o, \vec{\omega}_i). \tag{4.16}$$

BRDFs, BTDFs, BSDFs, and BSSRDFs

It is possible to define similar functions that describe the amount of transmitting light. In general, we have the following four possible functions [108]:

$$\begin{aligned} f_r^+ &: \Omega_+ \times \Omega_+ \to \mathbb{R} \\ f_r^- &: \Omega_- \times \Omega_- \to \mathbb{R} \\ f_t^+ &: \Omega_+ \times \Omega_- \to \mathbb{R} \\ f_t^- &: \Omega_- \times \Omega_+ \to \mathbb{R}, \end{aligned} \tag{4.17}$$

where f_r^+ and f_r^- define BRDFs for outer and inner surfaces, and f_t^+ and f_t^- are called *Bidirectional Transmittance Distribution Function* (BTDF) that define the transmittance for incoming and outgoing light. The union of the functions is called *Bidirectional Scattering Distribution Function* (BSDF).

The BRDF does not describe all the phenomena that we can see in the real world. For example, the BRDF assumes that the wavelength of light does not change after its reflection/transmission. This is not the case for fluorescence where incoming light is converted into another light with different wavelength. The BRDF also makes the assumption that the outgoing light exits at the point where the incoming light arrives. This approximately is largely incorrect for surfaces that exhibits subsurface light transport such as human skin, wax, milk, and many organic materials. The generalized function called Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF) [88] formulates such materials, but this thesis mainly focuses on surfaces that can be described by the BRDF.

4.4 Rendering Equation

Given all the notations, we can now formulate the light transport problem as the equilibrium distribution of light energy as the rendering equation [66, 57]:

$$L(\vec{x} \to \vec{\omega}) = L_e(\vec{x} \to \vec{\omega}) + \int_{\Omega} f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}) L(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}}, \tag{4.18}$$

where $L_e(\vec{x} \to \vec{\omega})$ is the emitted radiance from the point $\vec{x}$ toward the direction $\vec{\omega}$. The equation is a recursive integral equation since the unknown radiance appear on the both side of the equation (formally, this is a Fredholm equation of the second kind). Note that this equation assumes the absence of participating media that alter paths of light between surfaces. Examples of participating media are smoke, fog, clouds, and fire. In other words, we assume that objects are surrounded by vacuum.

Except for some restricted cases [107], analytic solutions to this equation are difficult to obtain. The goal of a light transport algorithm is to numerically approximate the solution to this equation at an arbitrary point. The following subsections describe different formulations and methods to solve the rendering equation.

4.4.1 Operator Formulation

Suppose that we define the operator $\mathbf{T}$ as

$$\mathbf{T}g(\vec{x} \to \vec{\omega}) = \int_{\Omega} f_r(\vec{x}, \vec{\omega}_i, \vec{\omega}) g(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}}. \tag{4.19}$$

Using this operator $\mathbf{T}$, we can reformulate the rendering equation as

$$L(\vec{x} \to \vec{\omega}) = L_e(\vec{x} \to \vec{\omega}) + \mathbf{T}L(\vec{x} \to \vec{\omega}) \tag{4.20}$$

Using $\mathbf{I}$ as the identity operator, we can solve the above equation for $L(\vec{x} \to \vec{\omega})$ as

$$\begin{aligned} \mathbf{I}L(\vec{x} \to \vec{\omega}) &= L_e(\vec{x} \to \vec{\omega}) + \mathbf{T}L(\vec{x} \to \vec{\omega}) \\ \mathbf{I}L(\vec{x} \to \vec{\omega}) - \mathbf{T}L(\vec{x} \to \vec{\omega}) &= L_e(\vec{x} \to \vec{\omega}) \\ (\mathbf{I} - \mathbf{T})L(\vec{x} \to \vec{\omega}) &= L_e(\vec{x} \to \vec{\omega}) \\ L(\vec{x} \to \vec{\omega}) &= (\mathbf{I} - \mathbf{T})^{-1} L_e(\vec{x} \to \vec{\omega}). \end{aligned} \tag{4.21}$$

The result tells us that the solution to the rendering equation is the result of applying the operator $(\mathbf{I} - \mathbf{T})^{-1}$ to the input emittance distribution $L_e(\vec{x} \to \vec{\omega})$.

This formulation is the basis of the algorithms using finite elements methods such as the radiosity method [37] and its extension [57]. The method first discretizes the solution space $L(\vec{x} \to \vec{\omega})$ such that the integral can be written as a weighted sum of discretized elements. The operator $\mathbf{T}$ then becomes a matrix and L becomes the solution to the linear equation $L = L_e + \sum_i T_i L$. Although finite elements methods are not actively used anymore because of the need of discretization and computational cost of solving linear equations, some recent work revisited this formulation and demonstrated promising results [18, 78].

4.4.2 Path Formulation

We can expand the rendering equation as follows

$$\begin{aligned} L &= L_e + \mathbf{T}L \\ &= L_e + \mathbf{T}(L_e + \mathbf{T}L) \\ &= L_e + \mathbf{T}L_e + \mathbf{T}^2 L \\ &= L_e + \mathbf{T}L_e + \mathbf{T}^2(L_e + \mathbf{T}L) \\ &= L_e + \mathbf{T}L_e + \mathbf{T}^2 L_e + \mathbf{T}^3 L_e \cdots . \end{aligned} \tag{4.22}$$

Since the operator $\mathbf{T}L_e$ corresponds to integral of the known input function L_e, this expression tell us that the solution rendering equation is the sum of integrals. One can abstract the sum of integrals as a single integral over the union of space $\mathbf{T}^i$ as *p*ath space Ω^*;

$$L(\vec{x} \to \vec{\omega}) = \int_{\Omega^*} f(\vec{X})\, d\mu(\vec{X}), \tag{4.23}$$

where $d\mu(\vec{X})$ is a measure in path space, $\vec{X}$ is any path that evaluates the function of $\mathbf{T}^i$ with some i. In a concrete example, a path corresponds to a light path that is coming from the light source and reaching to the eye. This formulation was formally introduced into light transport simulation by Veach [117], but it has been the basis of many stochastic light transport algorithms before its formal introduction.

The path formulation is useful to formulate the algorithms based on Monte Carlo integration such as path tracing [66], bidirectional path tracing [71, 118], and the more advanced methods based on Markov Chain Monte Carlo methods [15, 119]. Common to all the methods is that they all solve the integral of the path formulation using Monte Carlo integration. The difference lies in how to perform Monte Carlo integration, and different methods use different techniques such as importance sampling that we reviewed in Chapter 2.

4.4.3 Density Formulation

Another way to look at the rendering equation is based on a photon trajectory simulation. Recall that radiance is flux per projected area per solid angle. Since flux is called by photons, flux is proportional to the number of photons, and we can evaluate radiance if we know the distribution of photons.

If we write the expected number of photons we can observe from the position $\vec{x}$ toward the direction $\vec{\omega}$ as $N(\vec{x} \to \vec{\omega})$, a photon trajectory simulation can be formulated as

$$N(\vec{x} \to \vec{\omega}) = Q(\vec{x} \to \vec{\omega}) + \int_{\Omega} p(\vec{x}, \vec{\omega}_i, \vec{\omega}) N(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}}, \tag{4.24}$$

where $p(\vec{x}, \vec{\omega}_i, \vec{\omega})$ is a transition probability density function of sampling a photon that comes to $\vec{x}$ toward the direction $\vec{\omega}$ from another photon that is coming from the direction $\vec{\omega}_i$, and $Q(\vec{x} \to \vec{\omega})$ is the initial photon distribution (i.e., initial emittance distribution). This equation is equivalent to the rendering equation if $p(\vec{x}, \vec{\omega}_i, \vec{\omega}) = f_r(\vec{x}, \vec{\omega}_i, \vec{\omega})$.

The standard approach for estimating $N(\vec{x} \to \vec{\omega})$ is to recording particles based on random walk simulations. To be precise, we first start from the initial distribution

$$N(\vec{x} \to \vec{\omega}) \approx \hat{N}(\vec{x} \to \vec{\omega}) = Q(\vec{x} \to \vec{\omega}). \tag{4.25}$$

We then generate a new particle distribution by random walk simulation of the current distribution $\hat{N}(\vec{x} \to \vec{\omega})$ using the transition probability density function given by the scene configuration as

$$\int_{\Omega} p(\vec{x}, \vec{\omega}_i, \vec{\omega}) \hat{N}(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}}. \tag{4.26}$$

This new distribution is added to the current distribution

$$N(\vec{x} \to \vec{\omega}) \approx \hat{N}(\vec{x} \to \vec{\omega}) + \int_{\Omega} p(\vec{x}, \vec{\omega}_i, \vec{\omega}) \hat{N}(\vec{x} \leftarrow \vec{\omega}_i)\, d\omega_{i\vec{n}} \tag{4.27}$$

and we iterate the whole process by using the right hand side of the above equation as the current distribution. Since radiance is obtained as density of particles, we need to use density estimation techniques that are reviewed in Chapter 3.

To the best of our knowledge, this formal description of photon density estimation methods has been largely ignored except for the book by Dutré [22]. One common alternative description is that photon density estimation solves for the 'blurred solution' of the rendering equation. However, this description does not precisely differentiate the Monte Carlo integration methods and photon density estimation. In this alternative description, applying blur seems a redundant intentional operation, given the solution of the rendering equation. The above formulation more cleanly describes why we need to introduce "blur" by density estimation.

This formulation is the basis of multiple algorithms [61, 104, 121] as well as this dissertation work. The main difference lies in density estimation techniques they employed. In this dissertation work, we use a novel density estimation framework that is suitable for solving the light transport problem using this formulation.

The methods based on Markov Chain Monte Carlo to generate paths [15, 73, 119] can actually be categorized into this formulation as well. These methods directly generate samples from the unknown distribution $N(\vec{x} \to \vec{\omega})$ by means of Markov Chain Monte Carlo methods, and compute the constant factor using Monte Carlo integration to convert $N(\vec{x} \to \vec{\omega})$ into $L(\vec{x} \to \vec{\omega})$. These methods use a histogram density estimator over an image to estimate the distribution from samples.

Chapter 5

Progressive Density Estimation

The basis of this dissertation is a novel density estimation method called progressive density estimation. The key is that, unlike regular density estimation methods, progressive density estimation can produce a solution with any desired accuracy without multiple runs with different parameters. In other words, convergence of progressive density estimation is built into the algorithm itself. We applied progressive density estimation to build a new light transport simulation algorithm which we named as progressive photon mapping. Compared with existing light transport algorithms based on Monte Carlo integration, progressive photon mapping is more efficient and robust in the presence of complex light paths such as ones from a lighting fixture.

5.1 Related Work

Photon mapping is a two-pass light transport simulation algorithm developed by Jensen [61]. The first pass is building a photon map using photon tracing, and the second pass uses ray tracing to render the image. In the ray tracing pass the photon map is used to estimate the radiance at different locations within the scene. This is done by locating the nearest photons and performing a nearest neighbor density estimation. The density estimation process can be considered as a way of loosely connecting paths from the eye to the light.

The density estimation process effectively blurs the lighting in the scene, and to represent sharp illumination details it is necessary to use a large number of photons. The main challenge is scenes with lighting dominated by caustics. This type of lighting is rendered using a direct visualization of the photon map, which can require a large number of photons to resolve the details. Since the photon map is stored in memory, the final quality is often limited by the maximum number of photons that can be stored in the photon map. We overcome this problem in photon mapping by using a sequence of smaller photon maps without the need of storing all the photons.

To improve the performance of photon mapping and final gathering Havran et al. [51] introduced the concept of reverse photon mapping. Reverse photon mapping uses ray tracing in the first pass and photon tracing in the second pass. The ray tracing step builds a kd-tree over the hit points and in the photon tracing step this kd-tree is used to find the nearest hit points that a photon contributes to. The motivation for this approach is to reduce the complexity and improve the performance of photon mapping when a large number of rays are used in the ray tracing pass (e.g., as part of final gathering). We use a concept similar to reverse photon mapping, but our goal is to compute a highly accurate solution of light transport simulation in scenes with complex lighting and to overcome the inherent limitation of the final image quality in standard photon mapping.

Suykens and Willems [110] proposed an adaptive image filtering algorithm that asymptotically converges to the correct solutions. They described several heuristics that reduces the width of filtering kernel based on the number of samples. Our method also uses the number of photons to reduce the search radius of density estimation, which asymptotically converges to the correct solutions similar to their method. The key difference is that our method utilizes the fact that photon mapping is a consistent method. Therefore we can keep the robustness of photon mapping while obtaining asymptotically correct images.

A number of papers have addressed the issues in the standard photon mapping algorithm. Ray splatting [53] removes boundary bias and topology bias [101] by using rays rather than photons to perform density estimation. Proximity bias is not avoided as these methods still rely on regular density estimation. Fradin et al. [29] presented an out of core photon mapping approach optimized for large buildings where only a small part of the photon map is used. Christensen et al. [14] introduced brick maps as a compact approximate representation of the illumination represented by the photon map.

None of these existing work essentially resolved the issue of storage requirement for many photons which comes from a fundamental limitation existing density estimation methods. With progressive density estimation, for the first time, we can use an unlimited number of samples with a finite amount of memory.

5.2 Issues with Regular Photon Mapping

We briefly review the photon mapping [62] algorithm in this section. The first pass of photon mapping is photon tracing, which traces photons from the light sources into the scene and stores them in a photon map as they interact with the surfaces. The second pass is rendering in which the photon map is used to estimate the illumination in the scene. Given a photon map, exitant radiance at any surface location x can be estimated as:

$$L(x, \vec{\omega}) \approx \sum_{p=1}^{n} \frac{f_r(x, \vec{\omega}, \vec{\omega}_p)\phi_p(x_p, \vec{\omega}_p)}{\pi r^2}, \tag{5.1}$$

where n is the number of nearest photons used to estimate the incoming radiance. ϕ_p is the flux of the pth photon, f_r is the BRDF, $\vec{\omega}$ and $\vec{\omega}_p$ are the outgoing and incoming directions. r is the radius of the sphere containing the n nearest photons. This estimate assumes that the local set of photons represents incoming radiance at x, and that the surface is locally flat around x.

The photon density estimation in Equation 5.1 is the source of bias in photon mapping. To ensure convergence to the correct solution, it is necessary to use an infinite number of photons in the photon map and in the photon density estimation. Furthermore, the radius should converge to zero. Recall that these are the conditions of consistency for density estimation in general (Chapter 3).

We can satisfy these requirements, for example, by using N photons in the photon map, but only N^{β} with $\beta \in]0:1[$ photons in the photon density estimation. As N becomes infinite both N and N^{β} will become infinite, but N^{β} will be infinitely smaller than N, which ensures that r will converge to zero. This can be written as [62]:

$$L(x,\vec{\omega}) = \lim_{N\to\infty} \sum_{p=1}^{\lfloor N^{\beta} \rfloor} \frac{f_r(x,\vec{\omega},\vec{\omega}_p)\phi_p(x_p,\vec{\omega}_p)}{\pi r^2}, \tag{5.2}$$

Similar to other density estimation methods, in standard photon mapping, this result is only of theoretical interest since all the photons have to be stored in memory in order to use this result. This makes it impossible to obtain a solution with arbitrary precision with a single run of the algorithm. In the following sections, we describe a new density estimation method that fulfills the requirements of Equation 5.2 without having to store all the photons in memory.

5.3 Progressive Density Estimation

The traditional density estimation as given in Equation 5.1 relies on an estimate of the local density of photons. The estimate of the local density $d(x)$ is

$$d(x) = \frac{n}{\pi r^2}. \tag{5.3}$$

This estimate is based on locating the n nearest photons within a sphere of radius r, assuming that the surface is locally flat such that the photons are located within a disc. If we generate another photon map and use it to compute the density at x, we might find n' photons within the *same disc*, which may results in a different density estimate $d'(x)$:

$$d'(x) = \frac{n'}{\pi r^2}. \tag{5.4}$$

Note that we are using the same radius as Equation 5.3. By averaging $d(x)$ and $d'(x)$ we can obtain a more accurate estimate of the density within the disc of radius r.

This approach was proposed by Christensen [64], and it will lead to a less noisy density estimate, but the final result does not have more detail than each individual photon map. Furthermore, the averaging procedure is not consistent and the method will not converge to the exact value at x. Instead, it computes the average value within a constant radius r. As a result, it cannot resolve small details within the radius r, and the accuracy is effectively limited by the total number of photons in each individual photon map.

Progressive density estimation combines the result from several photon maps in a way that the final estimate will converge to the correct solution. It is able to resolve details in the illumination that is not captured by the individual photon maps. The key insight that makes this possible is a new technique for reducing the radius in the density estimate at each measurement point, while increasing the number of accumulated photons. This effectively ensures that the number of samples in an infinitely small region diverges in the limit in accordance with Equation 5.2. The following sections describe how the photon density is progressively computed.

5.3.1 Radius Reduction

Each point has a radius, $R(x)$. Initially, the radius, $R(x)$, at x is set to a non-zero value such as the footprint of the pixel. It is also possible to estimate the radius after the first photon tracing pass by using regular photon mapping to obtain the radius around each point. Our goal is to reduce this radius while increasing the number of photons accumulated within this radius, $N(x)$. The density $d(x)$ at a point x is computed using Equation 5.3. Assume that a number of photon tracing steps have been performed and that $N(x)$ photons have been accumulated at x. If we perform one additional photon tracing step and find $M(x)$ photons within the radius $R(x)$ then we can add these $M(x)$ photons to x, which results in a new photon density $\hat{d}(x)$:

$$\hat{d}(x) = \frac{N(x) + M(x)}{\pi R(x)^2}\,. \tag{5.5}$$

The next step of the algorithm is reducing the radius $R(x)$ by $dR(x)$. If we assume that the photon density is constant within $R(x)$, we can compute the new total number of photons $\hat{N}(x)$ within a disc of radius $\hat{R}(x) = R(x) - dR(x)$ as:

$$\hat{N}(x) = \pi \hat{R}(x)^2 \hat{d}(x) = \pi (R(x) - dR(x))^2 \hat{d}(x)\,. \tag{5.6}$$

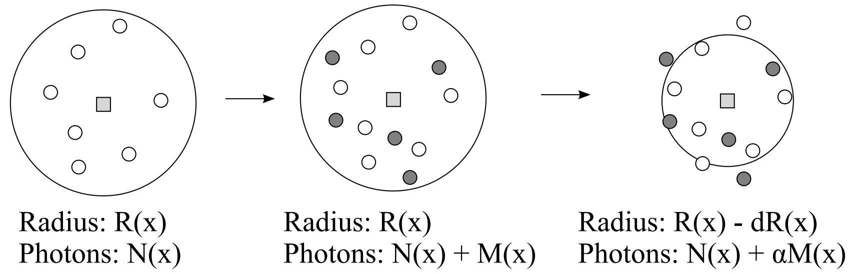


Figure 5.1: Each point in the ray tracing pass is stored in a global data structure with an associated radius and accumulated photon power. After each photon tracing pass we find the new photons within the radius of each point, and we reduce the radius based on the newly added photons. Progressive density estimation ensures that the final value at each point will converge to the correct radiance value.

To satisfy the consistency condition in Equation 5.2, there has to be a gain in the total number of photons at every iteration (i.e. $\hat{N}(x) > N(x)$). A single parameter $\alpha = (0,1)$ controls the fraction of photons to keep after every iteration. Therefore, $\hat{N}(x)$ can be computed as:

$$\hat{N}(x) = N(x) + \alpha M(x)\,, \tag{5.7}$$

which states that we would like to add $\alpha M(x)$ new photons at each iteration. Note that the value of $\alpha M(x)$ can be a real number, rather than an integer. We can compute the actual reduction of the radius $dR(x)$ by combining Equations 5.5, 5.6 and 5.7:

$$\begin{aligned} & \pi(R(x) - dR(x))^2 \hat{d}(x) = \hat{N}(x) \\ \Leftrightarrow \quad & \pi(R(x) - dR(x))^2 \frac{N(x)+M(x)}{\pi R(x)^2} = N(x) + \alpha M(x) \\ \Leftrightarrow \quad & dR(x) = R(x) - R(x)\sqrt{\frac{N(x)+\alpha M(x)}{N(x)+M(x)}}\,. \end{aligned} \tag{5.8}$$

Finally, the reduced radius $\hat{R}(x)$ is computed as:

$$\hat{R}(x) = R(x) - dR(x) = R(x)\sqrt{\frac{N(x)+\alpha M(x)}{N(x)+M(x)}}\,. \tag{5.9}$$

Note that this equation is solved independently for each point.

5.3.2 Flux Correction

When a point receives $M(x)$ photons, we need to accumulate the flux carried by those photons. In addition, we need to adjust this flux to take into account the radius reduction described in the previous section. Each point stores the unnormalized total flux received pre-multiplied by the BRDF. This quantity $\tau(x,\vec{\omega})$ for the $N(x)$ photons is computed as:

$$\tau_N(x,\vec{\omega}) = \sum_{p=1}^{N(x)} f_r(x,\vec{\omega},\vec{\omega}_p)\phi'_p(x_p,\vec{\omega}_p)\,, \tag{5.10}$$

where $\vec{\omega}$ is the direction of the incident ray at the point, $\vec{\omega}_p$ is the direction of the incident photon, and $\phi'_p(x_p,\vec{\omega}_p)$ is unnormalized flux carried by the photon p. Note, that the flux at this stage is not divided by the number of emitted photons as in standard photon mapping. Similarly, the $M(x)$ new photons give:

$$\tau_M(x,\vec{\omega}) = \sum_{p=1}^{M(x)} f_r(x,\vec{\omega},\vec{\omega}_p)\phi'_p(x_p,\vec{\omega}_p)\,. \tag{5.11}$$

If the radius was constant, one could simply add $\tau_M(x,\vec{\omega})$ to $\tau_N(x,\vec{\omega})$, but since the radius is reduced, it is necessary to account for the photons that fall outside the reduced radius (see Figure 5.1). One method for finding those photons would be to keep a list of all photons within the disc and remove those that are not within the reduced radius disc. However, this method is not practical as it would require too much memory for the photon lists. Instead, progressive density estimation assumes that the illumination and the photon density within the disc is constant, which results in the following adjustment:

$$\begin{aligned}
\tau_{\hat{N}}(x,\vec{\omega}) &= (\tau_N(x,\vec{\omega}) + \tau_M(x,\vec{\omega}))\frac{\pi\hat{R}(x)^2}{\pi R(x)^2} \\
&= \tau_{N+M}(x,\vec{\omega})\frac{\pi\left(R(x)\sqrt{\frac{N(x)+\alpha M(x)}{N(x)+M(x)}}\right)^2}{\pi R(x)^2} \\
&= \tau_{N+M}(x,\vec{\omega})\frac{N(x)+\alpha M(x)}{N(x)+M(x)}\,,
\end{aligned} \tag{5.12}$$

where $\tau_{N+M}(x,\vec{\omega}) = \tau_N(x,\vec{\omega}) + \tau_M(x,\vec{\omega})$, and $\tau_{\hat{N}}(x,\vec{\omega})$ is the reduced value for the reduced radius disc corresponding to $\hat{N}(x)$ photons. The assumption that the photon density and thereby the illumination is constant within the disc may not be correct initially, but it becomes increasingly true as the radius becomes smaller except for points exactly at illumination discontinuities. This is not an issue, since the illumination at discontinuities is undefined and the probability of having a point exactly at a discontinuity is zero.

5.3.3 Radiance Evaluation

After each photon tracing pass, we can evaluate the radiance at the points. Recall that the quantities stored include the current radius and the current intercepted flux multiplied by the BRDF. The evaluated radiance is multiplied by the pixel weight and added to the pixel associated with the point. To evaluate the radiance, we further need to know the total number of emitted photons $N_{emitted}$ in order to normalize $\tau(x,\vec{\omega})$. The radiance is evaluated as follows:

$$\begin{aligned} L(x,\vec{\omega}) &= \int_{2\pi} f_r(x,\vec{\omega},\vec{\omega}')L(x,\vec{\omega}')(\vec{n}\cdot\vec{\omega}')\,d\omega' \\ &\approx \frac{1}{\Delta A}\sum_{p=1}^{n} f_r(x,\vec{\omega},\vec{\omega}_p)\Delta\phi_p(x_p,\vec{\omega}_p) \\ &= \frac{1}{\pi R(x)^2}\frac{\tau(x,\vec{\omega})}{N_{emitted}}. \end{aligned} \tag{5.13}$$

Similar to the regular photon mapping, this formulation is not restricted to Lambertian materials as we pre-multiply the flux with the BRDF and store it as $\tau(x,\vec{\omega})$. Note that the radius $R(x)$ will not be reduced if the disc defined by $R(x)$ is within an unlit region (i.e., $M(x) = 0$). Although this situation seemingly breaks the conditions of consistency, it still converges to the correct solution $L(x,\vec{\omega}) = 0$, since $\tau(x,\vec{\omega})$ will not increase and $L(x,\vec{\omega}) \to 0$ as $N_{emitted} \to \infty$. Figure 5.3 numerically demonstrates that the progressive density estimate converges to the correct radiance value $L(x)$, while the radius $R(x)$ is reduced to zero, and the number of photons $N(x)$ grows to infinity. The progressive density estimate ensures that the photon density at each point increases at each iteration and it is therefore consistent in accordance with Equation 5.2.

5.3.4 Progressive Photon Mapping

This section describes the overall steps of the proposed light transport algorithm. Progressive photon mapping is a multi-pass algorithm in which the first pass is eye ray

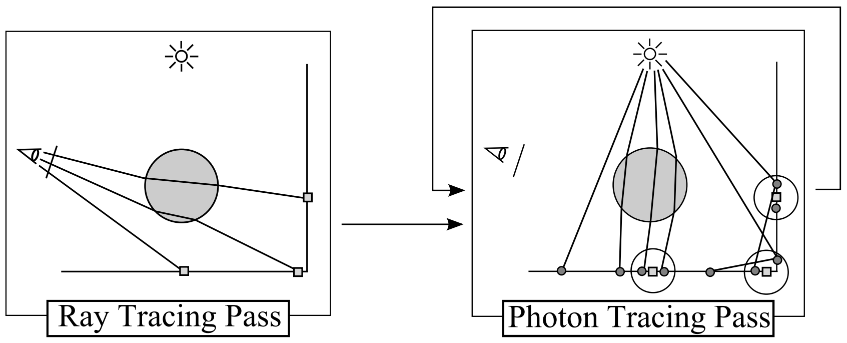


Figure 5.2: Progressive photon mapping uses eye ray tracing in the first pass followed by multiple photon tracing passes.

tracing and all subsequent passes use photon tracing. Each photon tracing pass improves the accuracy of the solution and the algorithm is progressive in nature. Figure 5.2 summarizes our algorithm.

Eye Ray Tracing Pass: The eye ray tracing pass is similar to reverse photon shooting [51] without the use of gathering. It uses standard ray tracing to find all the non-specular surfaces in the scene visible through each pixel in the image. Each eye path includes all specular bounces until the first non-specular surface seen. In scenes with a large number of specular surfaces, the length of the ray paths can be limited by using Russian Roulette.

For each eye path, we store all hit points along the path where the surface has a non-specular component in the BRDF. With each hit point we store the hit location x, the ray direction $\vec{\omega}$, scaling factors including BRDF and pixel filtering value, and the associated pixel location. In addition we store extra data necessary for the progressive density estimate including a radius, the intercepted flux, and the number of photons within the radius. Since these hit points are essentially the points that we would like to "measure" radiance during the computation, we often call those points as *measurement points* in this dissertation.

Note that nothing prevents us not to use eye ray tracing to generate these points. Measuring radiance at arbitrary location is possible as long as we can generate such a point. The argument that progressive photon mapping loses the view-independence of regular photon mapping is incorrect. One can generate measurement points independent from the view and display the result based on a novel view by means of any reasonable interpolation method (e.g., lightmaps).

We represent these values in the following structure:

```
struct mpoint {
  position $x$      Hit location
  normal $\vec{n}$      Normal at x
  vector $\vec{\omega}$      Ray direction
  integer B      BRDF index
  float x,y      Pixel location
  color wgt      Pixel weight
  float R        Current photon radius
  float N        Accumulated photon count
  color $\tau$        Accumulated reflected flux
}
```

Photon Tracing Passes: The photon tracing step is used to accumulate photon power at the measurement points found in the ray tracing pass. Each pass consists of tracing a given number of photons into the scene in order to build a photon map. After each photon tracing pass, we loop through all the measurement points and find the photons within the radius of each measurement point. We use the newly added photons to refine the estimate of the illumination within the measurement point as described in the following section. Once the contribution of the photons have been recorded, they are no longer needed, and we can discard all photons and proceed with a new photon tracing pass. This continues until enough photons have been accumulated and the final image quality is sufficient. Note that we can render an image after each photon tracing pass, which can provide feedback of the progress of rendering to a user. As more photons are accumulated the quality of the image will progressively improve toward the final result.

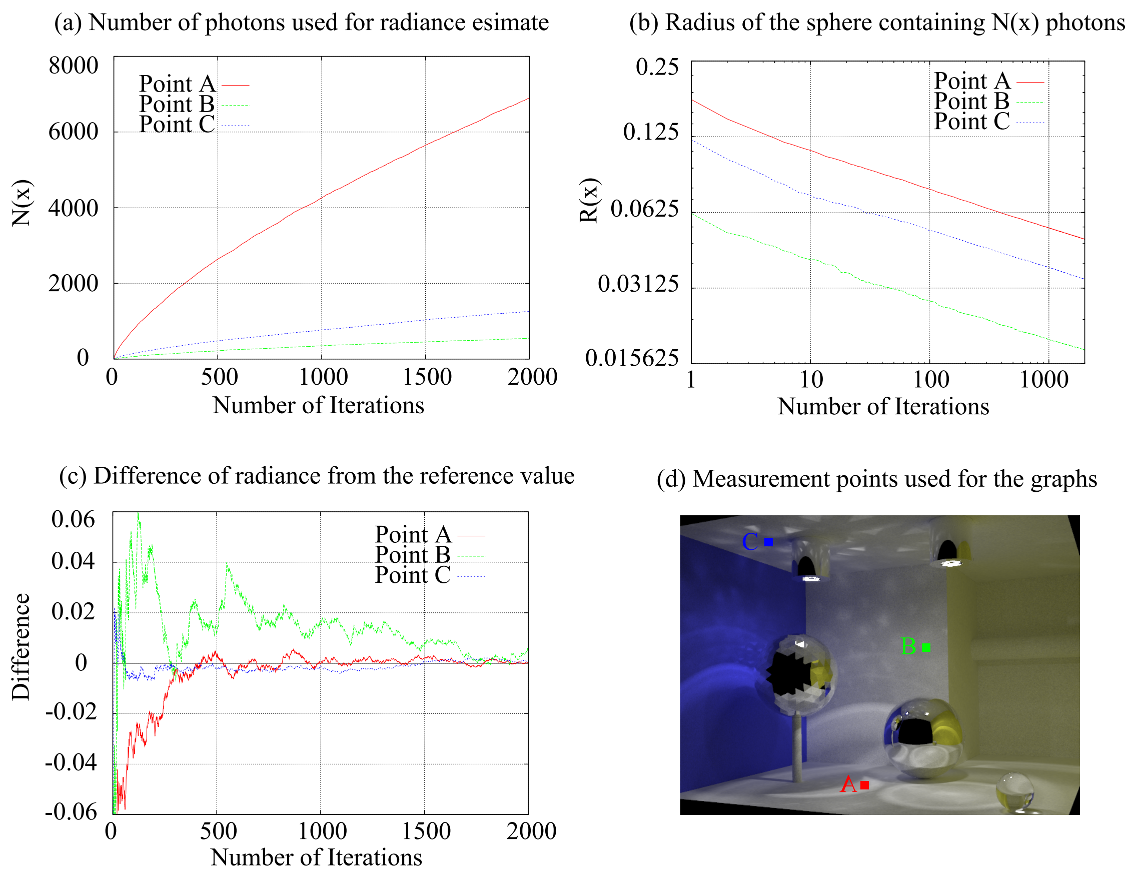


Figure 5.3: Statistics at the measurement points as a function of the number of iterations. The three measurement pointss A, B, and C are indicated in (d). Each iteration is using 100000 photons. Note that the graph of $R(x)$ uses log scale for both axes.

5.4 Results and Discussion

This section present results based on our implementation of the progressive photon mapping algorithm (PPM). Progressive photon mapping is compared against path tracing (PT) with explicit direct light source sampling, bidirectional path tracing (BDPT) with multiple importance sampling [118], Metropolis light transport [119] based on the primary sample space [67] (MLT), and photon mapping (PM) [62]. All the examples have been rendered on a PC with 1GB of memory and a 2.4GHz Intel Core 2 Q6600 using one core. The rendered images have a width of 640 pixels. Progressive photon mapping uses 100000 photons per photon shooting pass and $\alpha = 0.7$. The results of regular photon mapping is limited to 20 million photons due to memory constraints on the

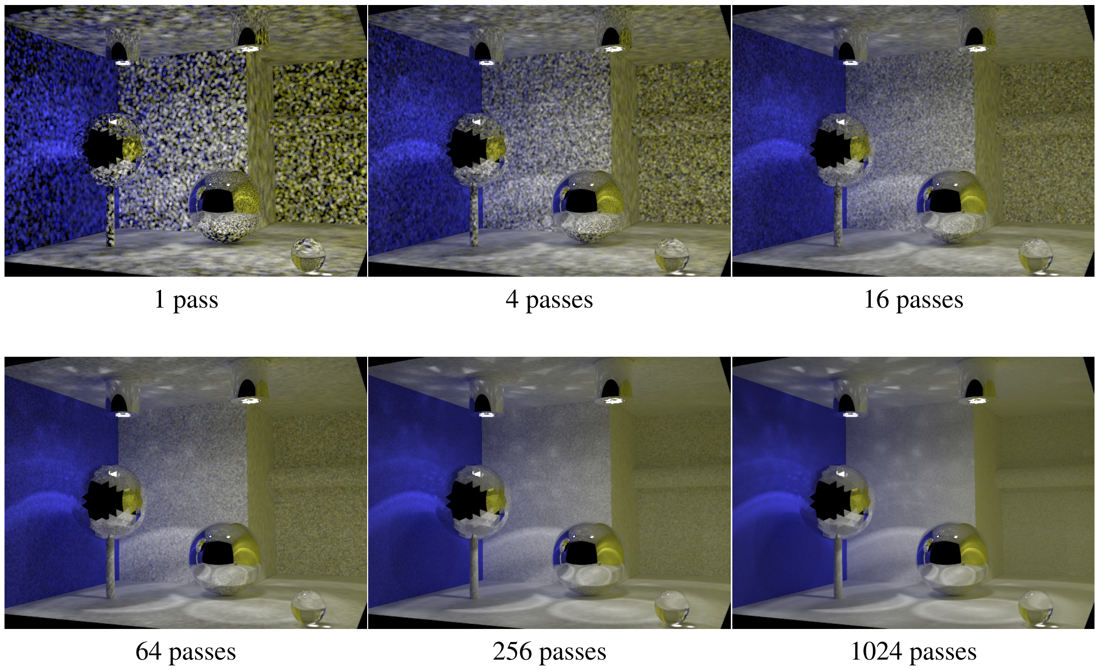


Figure 5.4: Sequence of images with increasing number of photons. The images show a direct visualization the progressive density estimate after 1, 4, 16, 64, 256, and 1024 photon tracing passes. Note how well the image with only after 16 passes captures rough illumination. Adding more photons makes the image sharper and reduces low frequency noise.

machine that was used for rendering. Regular photon mapping used 500-1500 photons in the density estimate for obtaining visually satisfactory results without too much bias or noise in the images. All the results are equal time comparisons except for photon mapping, where the rendering time is shorter since the total number of photons is limited by the available memory. Table 5.1 summarizes the computational effort for the different scenes.

Figure 5.8 shows a glass lamp illuminating a wall. The images have been rendered using BDPT, MLT, PM, and PPM. The image was rendered in 22 hours. Note how the illumination seen in the glass lamp is very noisy in all the Monte Carlo ray tracing methods, while the photon mapping result exhibits low frequency noise even with 20 million photons. Progressive photon mapping was able to use 165 million photons, which captures the sharp features in the illumination without the high frequency noise seen in the Monte Carlo ray tracing images.

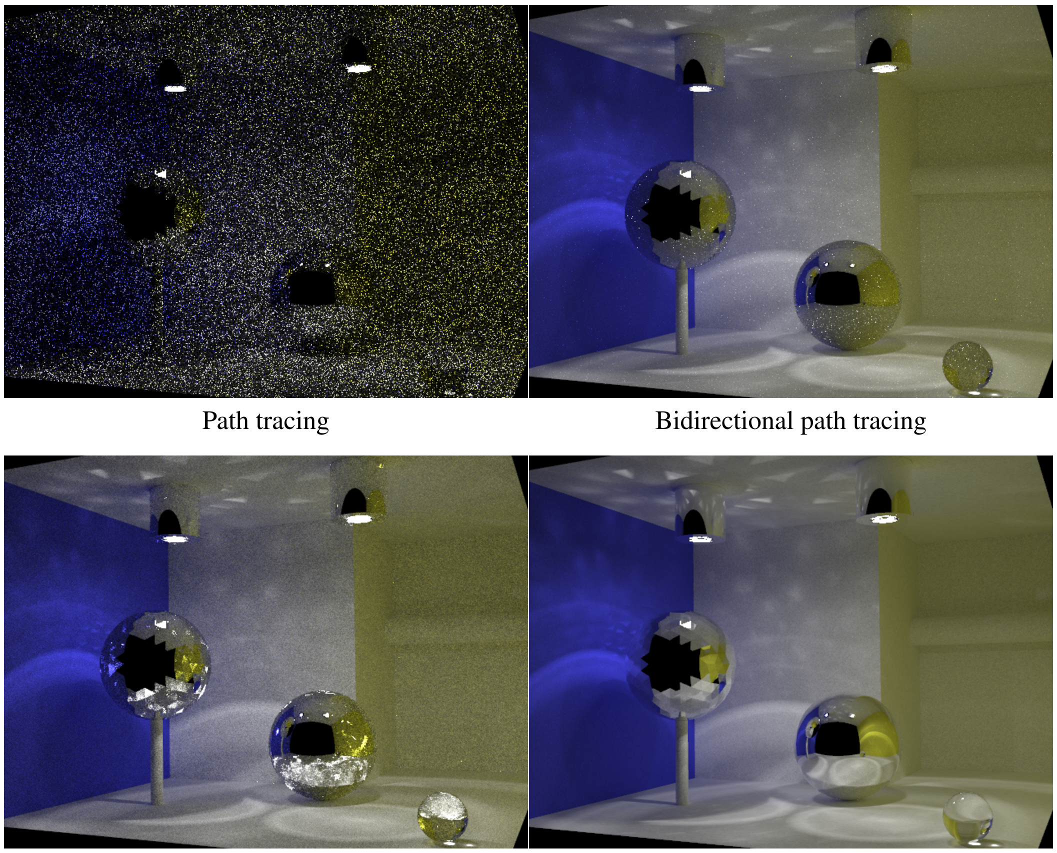
The test scene shown in Figure 5.5 is a box with two mirror balls and a smaller glass ball. One of the mirror balls is faceted to cause interesting caustics pattern. The scene is illuminated by two lighting fixtures, where each lighting fixture is a small spherical light source behind a spherical lens inside the metal cylinders at the ceiling. This type of caustics illumination results in numerous SDS paths within the model. Note how the reflections in the mirror ball are very noisy in both BDPT and MLT. Normal photon mapping generated in a slightly blurry result. Progressive photon mapping can increase the number of photons to 213 million, which makes it possible to capture the detailed illumination within the scene with considerable less noise than the unbiased Monte Carlo ray tracing methods. The unbiased Monte Carlo ray tracing images and the progressive photon mapping image were rendered in 4 hours, while the normal photon mapping image was rendered in 1 hour. The progressive nature of our method is demonstrated on the box scene in Figure 5.4 where the intermediate results have been visualized.

Figure 5.6 demonstrates the illumination on a torus embedded in a glass cube (inspired by a similar scene by Cline et al. [15]). All the Monte Carlo ray tracing methods have trouble rendering the lighting on the torus, while progressive photon mapping renders a smooth noise free result in the same time (2 hours). The reference image rendered using path tracing still exhibit noise even after using 51500 samples per pixel and 91 hours of rendering time.

The bathroom scene in Figure 5.7 is an example of complex geometry and illumination. The illumination in this scene is due to two small spherical area light sources enclosed within bumpy glass tubes. This type of illumination is common in real world lighting. Both BDPT and MLT fail to properly sample the reflection in the mirror and on the chrome pipes as well as the lighting behind the glass door. Our method robustly handles all of these illumination paths robustly. Normal photon mapping is also fairly robust in this scene, but it is not possible to render a noise free image using 20 million photons.

Table 5.1: Rendering statistics for the different scenes. The samples and mutations numbers are the average number per pixel. PPM uses 100000 photons per iteration.

	PT Samples	BDPT Samples	MLT Mutations	PPM Iterations	Time Hours
Lamp	840	80	82	1651	22
Box	1428	155	132	2134	4
Torus	1050	550	359	520	2
Bathroom	675	66	66	6126	16



Path tracing — Bidirectional path tracing

Metropolis light transport — Progressive photon mapping

Figure 5.5: Box scene illuminated by a lighting fixture. The lighting fixture is behind glass and the illumination in the scene is dominated by caustics. The specular reflections and refractions have significant noise even with Metropolis light transport. Standard photon mapping cannot resolve the sharp illumination details in the scene with the maximum 20 million photons in the photon map. Progressive photon mapping could use 213 million photons, which resolves all the details in the scene and provides a noise free image in the same rendering time as the Monte Carlo ray tracing methods.

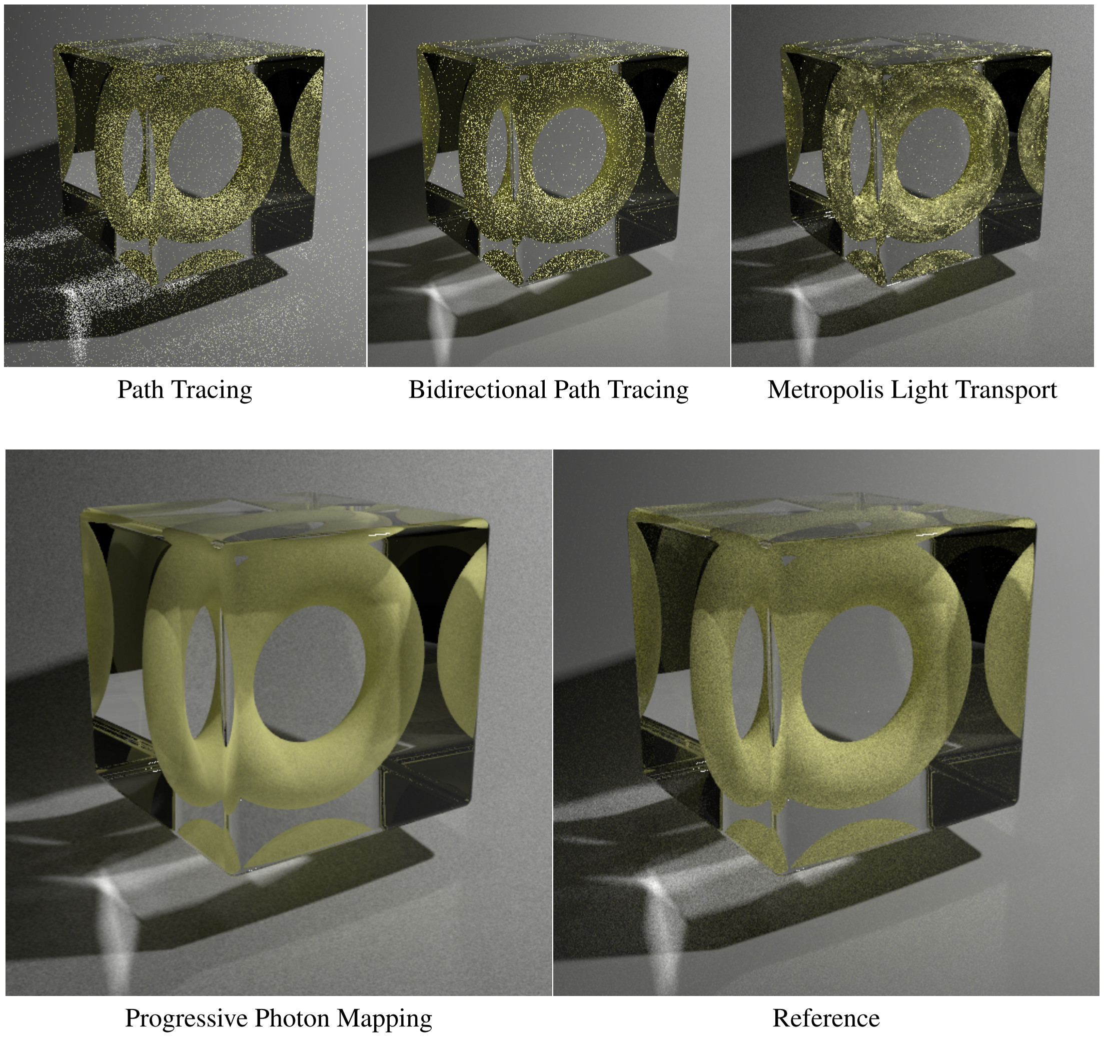

Path Tracing
Bidirectional Path Tracing
Metropolis Light Transport
Progressive Photon Mapping
Reference

Figure 5.6: Torus embedded in a glass cube. The reference image has been rendered using path tracing with 51500 samples per pixel. The Monte Carlo ray tracing methods fail to capture the lighting within the glass cube, while progressive photon mapping provides a smooth result using the same rendering time.

5.5 Theoretical Analysis

This section provides some theoretical analysis on progressive density estimation. Following the original work presented in this chapter, Knaus and Zwicker [70] recently provided a thorough theoretical analysis of progressive photon mapping. This section contains the original results obtained ahead of their work as well as observations on the convergence rate based on their results.

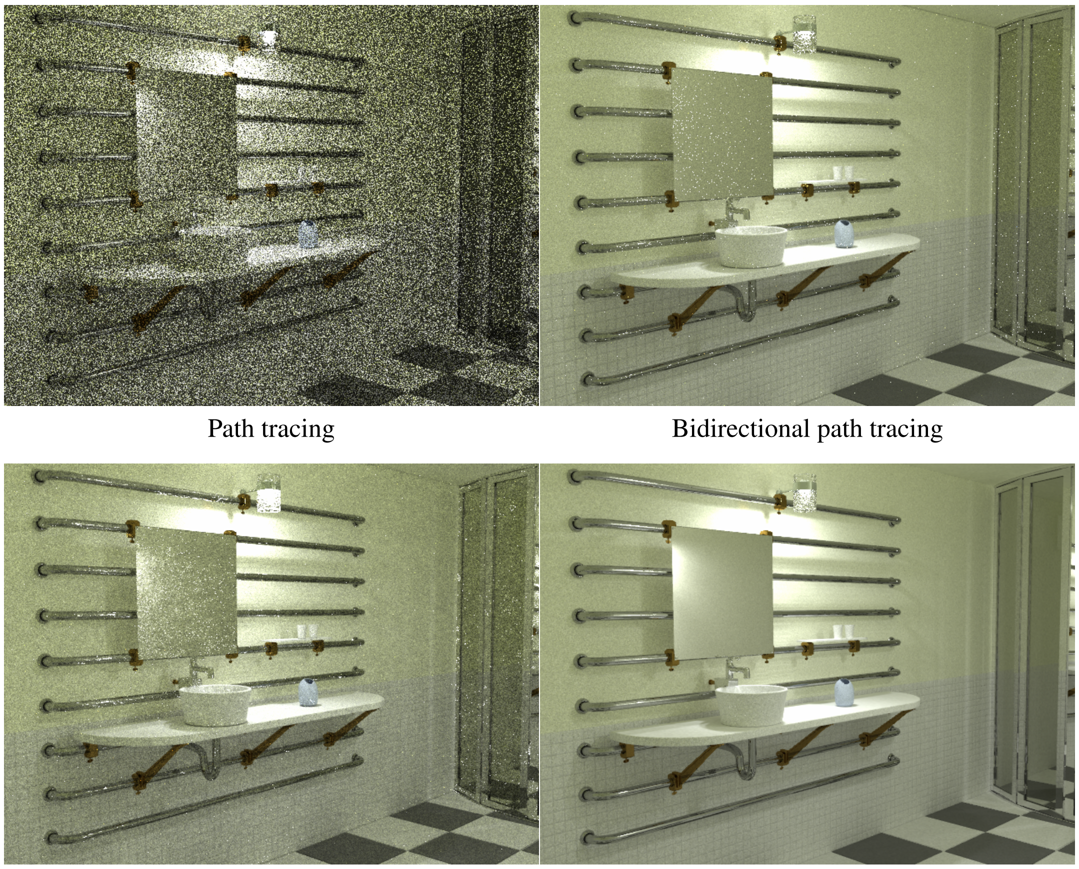

Path tracing
Bidirectional path tracing
Metropolis light transport
Progressive photon mapping

Figure 5.7: Lighting simulation in a bathroom. The scene is illuminated by a small lighting fixture consisting of a light source embedded in glass. The illumination in the mirror cannot be resolved using any Monte Carlo ray tracing method. Photon mapping with 20 million photons results in a noisy and blurry image, while progressive photon mapping is able to resolve the details in the mirror and in the illumination without noise.

5.5.1 Convergence of Radius

The following derivation shows that the radius converges to 0 by using an infinite number of samples. Recall that the equation to update the radius is as follows:

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}}, \tag{5.14}$$

where R_i is the radius, N_i is the number of local photons, and M_i is the number of photons within the radius at ith iteration. The radius at the next iteration, R_{i+1}, is computed based on the equation above with a user defined parameter $\alpha \in (0,1)$. For simplicity, the following proof considers the squared radius ${R_{i+1}}^2$. The squared radius is updated with

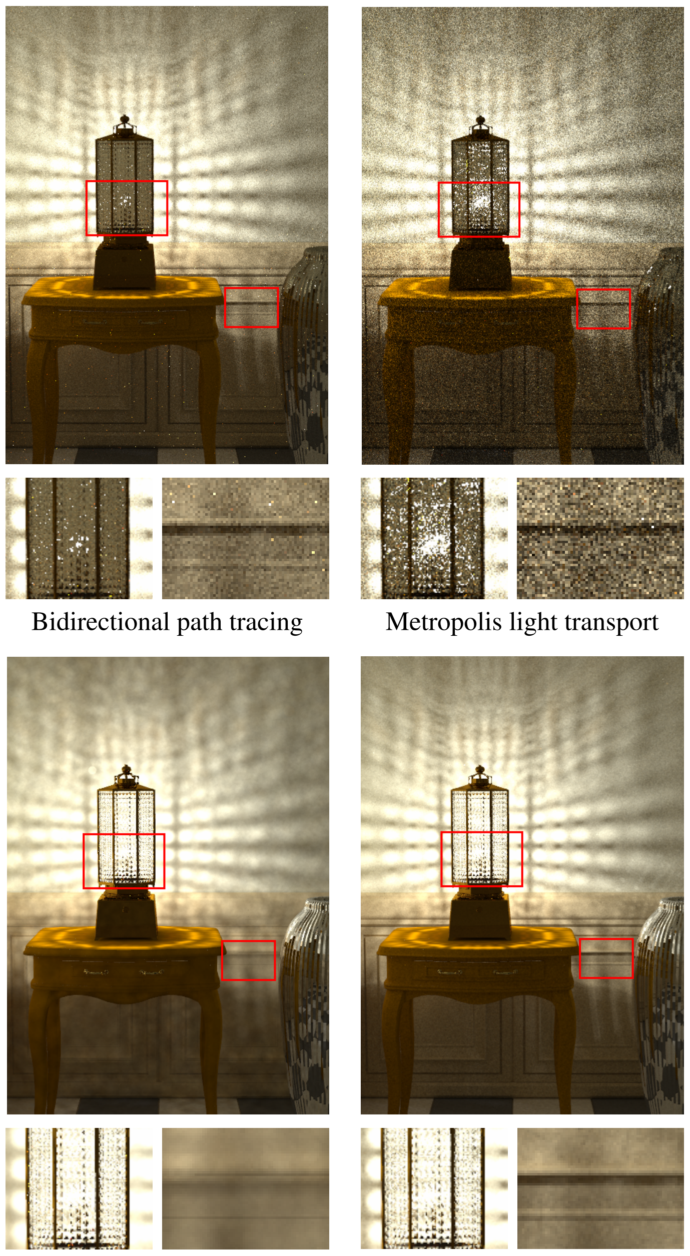

Bidirectional path tracing
Metropolis light transport
Photon mapping
Progressive photon mapping

Figure 5.8: A glass lamp illuminates a wall and generates a complex caustics lighting pattern on the wall. This type of illumination is difficult to simulate with Monte Carlo ray tracing methods. The lighting seen through the lamp is particularly difficult for these methods. Photon mapping is significantly better at capturing the caustics lighting seen through the lamp, but the final quality is limited by the memory available for the photon map and it lacks the fine detail in the illumination. Progressive photon mapping provides an image with substantially less noise in the same render time as the Monte Carlo ray tracing methods and the final quality is not limited by the available memory.

the following equation:

$$R_{i+1}{}^2 = \left(R_i\sqrt{\frac{N_i+\alpha M_i}{N_i+M_i}}\right)^2 = R_i{}^2\frac{N_i+\alpha M_i}{N_i+M_i}. \tag{5.15}$$

Note that the radius converges to 0 if and only if the squared radius converges to 0

$$\lim_{i\to\infty} R_i{}^2 = 0 \Leftrightarrow \lim_{i\to\infty} R_i = 0. \tag{5.16}$$

First, Equation 5.15 expands as follows:

$$\begin{aligned}
R_{i+1}{}^2 &= R_i{}^2\frac{N_i+\alpha M_i}{N_i+M_i}\\
&= \left(R_{i-1}{}^2\frac{N_{i-1}+\alpha M_{i-1}}{N_{i-1}+M_{i-1}}\right)\frac{N_i+\alpha M_i}{N_i+M_i}\\
&= R_{i-1}{}^2\frac{N_{i-1}+\alpha M_{i-1}}{N_{i-1}+M_{i-1}}\frac{N_i+\alpha M_i}{N_i+M_i}\\
&\cdots\\
&= R_0{}^2\frac{N_0+\alpha M_0}{N_0+M_0}\cdots\frac{N_i+\alpha M_i}{N_i+M_i}.
\end{aligned} \tag{5.17}$$

Since the initial squared radius $R_0{}^2$ is a non-zero value, showing $\lim_{i\to\infty} R_i{}^2 = 0$ is equivalent as showing that the following infinite product converges to 0:

$$\prod_{i=0}^{\infty}\frac{N_i+\alpha M_i}{N_i+M_i}. \tag{5.18}$$

Substituting the update equation for the number of local photons, $N_{i+1} = N_i + \alpha M_i$, we obtain:

$$\begin{aligned}
\prod_{i=0}^{\infty}\frac{N_i+\alpha M_i}{N_i+M_i} &= \prod_{i=0}^{\infty}\frac{N_{i+1}}{N_i+M_i}\\
&= \prod_{i=0}^{\infty}\frac{N_{i+1}}{N_{i+1}+(1-\alpha)M_i}\\
&= \prod_{i=0}^{\infty}\frac{1}{1+\frac{(1-\alpha)M_i}{N_{i+1}}}.
\end{aligned} \tag{5.19}$$

Since the numerator is always 1 and the product is positive, we can show that the infinite product converges to 0 if its reciprocal diverges to ∞:

$$\prod_{i=0}^{\infty}\frac{N_i+\alpha M_i}{N_i+M_i} = 0 \Leftrightarrow \prod_{i=0}^{\infty}\left(1+\frac{(1-\alpha)M_i}{N_{i+1}}\right) = \infty. \tag{5.20}$$

By defining a sequence $b_i = \frac{(1-\alpha)M_i}{N_{i+1}}$ we can also write down the infinite product as follows:

$$\prod_{i=0}^{\infty}\left(1+\frac{(1-\alpha)M_i}{N_{i+1}}\right)=\prod_{i=0}^{\infty}(1+b_i), \tag{5.21}$$

Since $b_i \geq 0$, we can bound the infinite product by

$$1+\sum_{i=0}^{\infty}b_i \leq \prod_{i=0}^{\infty}(1+b_i). \tag{5.22}$$

Therefore, the infinite product $\prod_{i=0}^{\infty}(1+b_i)$ diverges if the infinite sum $\sum_{i=0}^{\infty}b_i$ diverges. To show the the infinite sum diverges, first we substitute back b_i:

$$\sum_{i=0}^{\infty}b_i=\sum_{i=0}^{\infty}\frac{(1-\alpha)M_i}{N_{i+1}}. \tag{5.23}$$

If we look at the definition of N_i, it grows at most linearly to the number of iterations no matter how the photon shooting part is effective because we shoot the fixed number of photons per iteration. In addition, since $M_i \geq 1$, we have the following lower bound for sufficiently large K

$$\sum_{i=0}^{K}\frac{(1-\alpha)}{M_{max}(i+1)} \leq \sum_{i=0}^{K}\frac{(1-\alpha)}{N_{i+1}} \leq \sum_{i=0}^{K}\frac{(1-\alpha)M_i}{N_{i+1}}, \tag{5.24}$$

where M_{max} is the number of photons per iteration (i.e., the maximum number of photons in the radius at each iteration). Although it is possible to have $M_i = 0$ if the radius becomes small, we can safely ignore any iteration with $M_i = 0$ and reassigning indices i for the purpose of our proof. Note that any iteration with $M_i = 0$ merely increases computation time, and it does not change any variable such as R_i and N_i. Based on the definition of Riemann sum, we can further bound the summation by the integral for sufficiently large K:

$$\int_{1}^{K+1}\frac{(1-\alpha)}{M_{max}x}dx \leq \sum_{i=0}^{K}\frac{(1-\alpha)}{M_{max}(i+1)}. \tag{5.25}$$

The above integral can be easily solved as follows

$$\int_{1}^{K+1}\frac{(1-\alpha)}{M_{max}x}dx = \frac{(1-\alpha)}{M_{max}}\log(K+1), \tag{5.26}$$

so this integral diverges as $K \to \infty$. Therefore, the infinite sum also diverges. As a result

$$\begin{aligned}\sum_{i=0}^{\infty} \frac{(1-\alpha)M_i}{N_{i+1}} = \infty \Leftrightarrow \\ \prod_{i=0}^{\infty}\left(1 + \frac{(1-\alpha)M_i}{N_{i+1}}\right) = \infty \Leftrightarrow \\ \prod_{i=0}^{\infty} \frac{N_i + \alpha M_i}{N_i + M_i} = 0,\end{aligned} \tag{5.27}$$

which shows that R_i converges to 0 as i goes to infinity.

5.5.2 Divergence of Local Photon Count

Given the proof of convergence of radius, it is trivial to show N_i diverges since

$$N_{i+1} = N_i \frac{N_{i+1}}{N_i} = N_i \frac{N_i + \alpha M_i}{N_i} = N_i \left(1 + \frac{\alpha M_i}{N_i}\right) = N_0 \prod_{i=0}^{\infty}\left(1 + \frac{\alpha M_i}{N_i}\right), \tag{5.28}$$

which gives us a reciprocal of the same form of infinite product as in Equation 5.19 but with α instead of $1-\alpha$. Therefore, R_i converges to 0 and N_i diverges to ∞ as i goes to ∞. Along with convergence of radius, note that this result satisfies the consistency conditions of density estimate.

5.5.3 Convergence of Progressive Radiance Estimate

First, the estimated radiance $L_{j+1}(x)$ at the $j+1^{\text{th}}$ iteration using the refinement procedure expands as

$$\begin{aligned} L_{j+1}(x) &= \frac{\tau_{j+1}(x,\vec{\omega})}{\pi R_{j+1}(x)^2} = \frac{R_{j+1}(x)^2}{R_j(x)^2}\frac{\tau'_{j+1}(x,\vec{\omega})}{\pi R_{j+1}(x)^2} \\ &= \frac{j\tau_j(x,\vec{\omega}) + \tilde{\tau}_{j+1}(x,\vec{\omega})}{j+1}\frac{1}{\pi R_j(x)^2} \\ &= \frac{1}{j+1}\left(j\frac{\tau_j(x,\vec{\omega})}{\pi R_j(x)^2} + \frac{\tilde{\tau}_{j+1}(x,\vec{\omega})}{\pi R_j(x)^2}\right) \\ &= \frac{1}{j+1}\left(jL_j(x) + L_{p,j}(x)\right), \end{aligned} \tag{5.29}$$

where $L_{p,j}(x)$ is radiance estimate using only the photons from the j^{th} pass. Note that $\frac{\tilde{\tau}_{j+1}(x,\vec{\omega})}{\pi R_j(x)^2} = L_{p,j}(x)$ because $\tilde{\tau}_{j+1}(x,\vec{\omega})$ includes the intercepted photons only at the

$j+1^{\text{th}}$ iteration *before* the radius reduction and the radius reduction happens after flux accumulation (i.e., the radius used in the photon query is $R_j(x)$). The solution of this recurrence equation is

$$\begin{aligned} L_{j+1}(x) &= \frac{1}{j+1}\left(jL_j(x)+L_{p,j}(x)\right) \\ &= \frac{1}{j+1}\left((j-1)L_{j-1}(x)+L_{p,j-1}(x)+L_{p,j}(x)\right) \\ &= \frac{1}{j+1}\sum_{k=0}^{j} L_{p,k}(x). \end{aligned} \tag{5.30}$$

Progressive density estimate thus can be seen as regular radiance estimate where the kernel is the arithmetic average of kernels with sizes. The averaged kernel $K_{j+1}(x)$ is normalized as follows.

$$\int K_{j+1}(x)\,dx = \frac{1}{j+1}\sum_{k=0}^{j}\int \frac{1}{\pi R_k^2}\,dx = \frac{1}{j+1}\sum_{k=0}^{j} 1 = 1. \tag{5.31}$$

Note also that $\lim_{j\to\infty} K_{j+1}(\vec{o}) = \infty$ (where $\vec{o}$ is the center of the kernels) since

$$\lim_{j\to\infty} K_{j+1}(0) = \lim_{j\to\infty}\frac{1}{j+1}\sum_{k=0}^{j}\frac{1}{\pi R_k^2} > \lim_{j\to\infty}\frac{1}{(j+1)\pi R_{j+1}^2} \propto \lim_{j\to\infty}\frac{1}{N_{j+1}} = \infty. \tag{5.32}$$

Standard derivations can show that the sequence of $K_{j+1}(\vec{x})$ converges to the delta function [1] by noting that

$$K_{j+1}(x) = \frac{1}{j+1}\sum_{k=0}^{j} G_{\frac{1}{R_j}}(x) \tag{5.33}$$

and

$$\lim_{j\to\infty}\int K_{j+1}(x)\,dx = H(|x|) \tag{5.34}$$

where $H(x)$ is the Heaviside unit step function, $G_r(x) = rG(rx)$ with $G(x) = H(x) - H(x-1)$. Any function that satisfies $G_r(x) = rG(rx)$ with a parameter r can construct this converging kernel in the limit of $r\to\infty$ [1]. Since limits and derivatives commute for generalized functions, by taking the derivative of the above function, we obtain $\lim_{j\to\infty} K_{j+1}(\vec{x}) = \delta(\vec{x})$.

Therefore, the averaged kernel converges to a delta function in the limit. Since the number of local photon counts diverges, progressive radiance estimate is equivalent to regular radiance estimate with a delta function with an infinite number of local photons in the limit. Since there are many different sequences of functions that converge to the delta function [1], our result indicates that there is indeed a large class of dynamic kernels that has provable convergence to the correct solution under progressive density estimation framework.

This view also reveals a relationship of progressive density estimation to recursive density estimation [133, 134]. Recursive density estimation considers the average of kernel density estimation with a converging sequence of the bandwidth parameter. The important difference is that progressive density estimation uses sample statistics to construct this sequence of bandwidth, rather than using a predetermined sequence by the user. This allows a simple trade-off between bias and variance by the single parameter α, which is difficult to achieve with an arbitrary sequence. The formulation by progressive photon mapping is also more flexible in the sense that no assumptions are made for the sample generation stage such as non-adaptivity. Although there are certain differences, it would be interesting to investigate if some theories developed around recursive density estimation are also applicable to progressive density estimation [20, 129].

5.5.4 Convergence Rate

Based on the follow up work by Knaus and Zwicker [70], we can observe that progressive density estimate asymptotically converges at the rate of $O(N^{\frac{1}{3}})$ with $\alpha = \frac{2}{3}$. According to their results, their alternative formulation as well as the original formulation both have the following convergence rates for bias and variance:

$$\text{Variance}[L_N] = O\left(\frac{1}{N^{\alpha}}\right) \qquad \text{Bias}[L_N] = O\left(\frac{1}{N^{1-\alpha}}\right), \tag{5.35}$$

where $\alpha \in (0,1)$ is the parameter of progressive photon mapping that decides which fraction of the newly arrived photons will be kept for the next iteration. Notice that setting $\alpha = 1$ makes variance converge at the same rate as unbiased Monte Carlo methods while keeping bias constant (i.e., Bias $= O(1)$). This is because $\alpha = 1$ result in no radius reduction and progressive radiance estimation turns into histogram density estimation, which has the same convergence rate as unbiased Monte Carlo methods. Conversely, setting $\alpha = 0$ makes bias converge at $O(1/N)$, but variance will be constant.

Noting that expected squared error of *any* biased method is a sum of variance and squared bias, expected squared error in progressive photon mapping has the following convergence behavior for given α;

$$\mathrm{E}\left[(L_N - L)^2\right] = O\left(\frac{1}{N^{\alpha}}\right) + O\left(\frac{1}{N^{2(1-\alpha)}}\right). \tag{5.36}$$

The fastest convergence rate is achieved by an α that satisfies $O\left(\frac{1}{N^{\alpha}}\right) = O\left(\frac{1}{N^{2(1-\alpha)}}\right)$, that is, $\alpha = \frac{2}{3}$. Therefore, error in progressive density estimate asymptotically converges at $O(N^{\frac{1}{3}})$. Note that this convergence rate is slower than vanilla Monte Carlo integration methods which is $O(N^{\frac{1}{2}})$ for N samples. This result however should not be taken as an indication that progressive density estimate is not useful in practice. The difference in convergence rates does not describe actual efficiency in light transport simulation as the results in this chapter demonstrated.

5.6 Extension to Volumetric Density Estimation

The rendering equation assumes that scenes are surrounded by vacuum - in other words, photons travel along a straight line without changing flux until hit any surfaces. This assumption breaks when scenes contain participating media that affects paths of light. This section describes a simple extension of progressive density estimation for handling participating media. The key is to realize that progressive density estimation holds for density estimation using different query shapes, and to apply beam radiance estimation by Jarosz et al [60].

In the combination with beam radiance estimation, the assumption that photon density is constant in progressive density estimation becomes that *radial* distribution of density is constant. Note that this assumption does not require constant density *along the direction of ray*, which is usually not constant even if the radius is infinitely small.

Suppose that we have radially constant but axially varying density $D(t)$, where t is distance along the ray. The number of photons $N+M$ within the cylinder around the ray with the kernel width R is computed by integration:

$$N+M = \int_0^{\infty} \pi R^2 D(t)dt = \pi R^2 \int_0^{\infty} D(t)dt. \tag{5.37}$$

Note that we can take out R outside of the integral because R is constant along the ray in our application. Based on this setting, we would like to find the new radius $0 < (R - dR) < R$ which keeps $N + \alpha M$ photons. Using the same integration as above, we obtain

$$N + \alpha M = \pi (R - dR)^2 \int_0^\infty D(t)dt. \tag{5.38}$$

By substituting Equation 5.37, we obtain exactly the same equation as the standard progressive density estimation for dR

$$N + \alpha M = \pi (R - dR)^2 \frac{N + M}{\pi R^2}. \tag{5.39}$$

Note that we do not need to know axially varying density $D(t)$ explicitly in our computation of dR. The flux correction part is done exactly the same. The only change we need to make is to count the number of photons within the cylinder of ray to get M_i.

5.7 Conclusion

This chapter introduced a new light transport simulation algorithm, progressive photon mapping, that makes it possible to compute solutions with arbitrary accuracy in scenes with complex illumination. The algorithm is based on a new progressive density estimation framework which achieves convergence to the correct solution in a single run for the first time. Our results show that progressive photon mapping is particularly robust in scenes with complex caustics illumination, and it is more efficient than methods based on unbiased Monte Carlo integration.

5.8 Acknowledgements

The material in this chapter is, in part, a reproduction of the material published in Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen, “Progressive Photon Mapping”, ACM Transactions on Graphics (SIGGRAPH Asia 2008), December 2008 [45]. The dissertation author was the primary investigator and author of this paper. We would like to thank Youichi Kimura (Studio Azurite) for providing us the glass lamp model and the bathroom model. The dissertation author was supported in part by the NSF grant CPA 0701992.

Chapter 6

Progressive Error Estimation Framework

This chapter introduces an error estimation framework for progressive density estimation in the context of light transport simulation. The proposed framework characterizes the error by the sum of a bias estimate and a stochastic noise bound, which is motivated by stochastic error bounds formulation in biased methods. As a part of the error estimate, this chapter extends progressive density estimation to operate with smooth kernels. This extension enables the calculation of illumination gradients with arbitrary accuracy, which the framework uses to progressively compute the local bias in the density estimate. This chapter also describes how variance can be computed in progressive photon mapping, which is used to estimate the error due to noise. As an example application, numerical tests demonstrate how our error estimation can be used to compute images with a given error threshold. For this example application, the proposed framework only requires the error threshold and a confidence level to automatically terminate rendering. The results demonstrate how our error estimation framework works well in realistic scene configurations.

6.1 Related Work

It is well known that error estimation in unbiased Monte Carlo ray tracing algorithms, such as path tracing [66] and bidirectional path tracing [71, 118], can be done by computing the variance of the solution. This fact that error in unbiased Monte Carlo methods appears as variance has been used in many applications. For example, variance is often used for adaptive sampling in order to find regions with large error [77, 96, 112]. Assuming errors are distributed according the normal distribution, it is also possible to approximate a stochastic error bound based on variance, which provides an estimated range of error based on a confidence probability provided by the user. Unfortunately, the same technique cannot be used as an error estimate for biased Monte Carlo methods, such as photon mapping [62], since these methods suffer from *bias*.

Although error estimation in biased methods in general has not been well-studied compared to unbiased methods, there has been some effort to derive error estimators for biased methods. An example includes the error bound used to guide the hierarchical sampling of many point lights in Lightcuts [123] and the first-order Taylor expansion of error in a hypothetical Lambertian "split-sphere" environment used by irradiance caching [127].

If we focus on photon density estimation, Myszkowski [86] proposed to use multiple radii during density estimation to estimate bias, and then selected the estimate that minimized an error heuristic. Walter [121] improved this heuristic error estimation using a perceptual metric. Bias compensation [101] also uses multiple radii around the same point, but uses a binary search to find the radius with the minimum heuristic error. This method estimates bias by assuming the density estimate with the smallest radius is an unbiased estimate. These prior techniques all use similar error analyses as ours, however, none of them provide an actual estimate of error as is possible with unbiased methods. Our method provides an algorithm to estimate error, especially focusing on progressive photon mapping. Another difference is that our bias estimation is not based on multiple radii, but based on a mathematical approximation of bias used in the density estimation literature [105]. This frees a user from specifying the number of density estimates with different radii or the minimum radius.

In order to estimate bias, our method estimates derivatives of the illumination using kernel derivatives. The same idea is commonly used in standard density estimation methods [105]; however, the key difference is that our derivation is based on progressive density estimation. As with progressive density estimation, our derivative estimates converge to the correct solution with a bounded memory consumption. This is not the case for standard density estimation methods, which require storing an infinite number of samples to obtain the correct solution. Gradient computation of light transport itself has been studied in previous work [126, 97]. This chapter introduces an alternative method which integrates easily in the proposed error estimation framework.

There are several techniques that reduce the bias of photon density estimation [54, 50, 53]. Although it is conceivable to use our bias estimate for bias reduction, this application is not the target of the work in this chapter; instead, the goal is to estimate the bias with reasonable accuracy for error estimation in rendering.

Note that confidence interval estimation is often obtained through bootstrapping [47, 26] outside of graphics. Unfortunately, bootstrapping cannot be directly applied to progressive photon mapping because it is unclear how to handle the increasing number of samples in bootstrapping without consuming increasing amount of memory.

6.2 Reformulation of PPM using Normalized Flux

This section reformulates progressive photon mapping in order to derive our error estimation framework. This alternative formulation uses *normalized* flux (i.e., accumulated flux divided by the number of emitted photons) to formulate progressive photon mapping, whereas previous formulation used *un*-normalized accumulated flux. Table 6.1 summarizes the notation.

In progressive photon mapping, each measurement point tracks the photons which fall within its search radius R and accumulates their power into τ. After pass i, the radiance at a measurement point is approximated as:

$$L_i(x,\vec{\omega}) \approx \frac{\tau_i(x,\vec{\omega})}{\pi R_i(x)^2}. \tag{6.1}$$

After the first photon tracing pass, $\tau_1(x,\vec{\omega})$ stores the flux premultiplied by the

Table 6.1: Definitions of quantities used throughout this chapter.

Symbol	Description
x	Position of a measurement point
$\vec{\omega}$	Outgoing direction of a measurement point
N^e	Number of emitted photons in each pass
N_i^e	Total number of emitted photons after i passes
N_i	Accumulated intercepted photon count at x after pass i
M_{i+1}	Number of intercepted photons at x during pass $i+1$
R_i	Radius at photon pass i
α	Fraction of photons kept after each pass
$\tau_i(x,\vec{\omega})$	Flux within $R_i(x)$ after pass i
$\tilde{\tau}_{i+1}(x,\vec{\omega})$	Flux within $R_i(x)$ using only photons from pass i
$\tau'_{i+1}(x,\vec{\omega})$	Flux within $R_i(x)$ in pass $i+1$ (*before* radius reduction)
$\tau_{i+1}(x,\vec{\omega})$	Flux within $R_{i+1}(x)$ after pass $i+1$ (*after* radius reduction)

BRDF from all N_1 photons landed within radius R_1:

$$\tau_1(x,\vec{\omega}) = \frac{1}{N_1^e}\sum_{p=1}^{N_1} f_r(x,\vec{\omega},\vec{\omega}_p)\Phi_p(x_p,\vec{\omega}_p), \tag{6.2}$$

where $\vec{\omega}$ is the outgoing direction at the measurement point, $\vec{\omega}_p$ is the incoming direction of the photon, f_r is the BRDF, and $\Phi(x_p,\vec{\omega}_p)$ is the flux carried by photon p. The number of photons *emitted* in each pass is N^e, and $N_i^e = iN^e$ is the *total* number of emitted photons up to and including pass i. Note that the radius R_1 and number of *intercepted* photons N_1 both depend on the measurement point x, but the notation here omits these dependencies for simplicity.

After the first pass, these statistics are refined by performing additional photon tracing passes, each emitting N^e more photons. During pass $i+1$, M_{i+1} additional photons are intercepted at x. The flux from just these additional photons would be:

$$\tilde{\tau}_{i+1}(x,\vec{\omega}) = \frac{1}{N^e}\sum_{p=1}^{M_{i+1}} f_r(x,\vec{\omega},\vec{\omega}_p)\Phi_p(x_p,\vec{\omega}_p). \tag{6.3}$$

If we used all M_{i+1} photons and kept the radius fixed, we obtain an improved estimate for the flux using all $N_i + M_{i+1}$ photons by just accumulating this additional flux as:

$$\begin{aligned}\tau'_{i+1}(x,\vec{\omega}) &= \frac{1}{N^e_{i+1}} \sum_{p=1}^{N_i+M_{i+1}} f_r(x,\vec{\omega},\vec{\omega}_p)\Phi_p(x_p,\vec{\omega}_p), \\ &= \frac{N^e_i \tau_i + N^e \tilde{\tau}_{i+1}}{N^e_{i+1}} = \frac{i\tau_i + \tilde{\tau}_{i+1}}{i+1}.\end{aligned} \tag{6.4}$$

However, to obtain a consistent estimator, progressive photon mapping must ensure that the bias and the variance are reduced at each iteration. This is accomplished by reducing the radius and statistically keeping only a fraction of the intercepted photons at each iteration. A single parameter $\alpha \in (0,1)$ is used to control the fraction of photons to keep from each pass. If M_{i+1} photons are found within search radius R_i during photon pass $i+1$, the new accumulated photon count is computed as:

$$N_{i+1} = N_i + \alpha M_{i+1}, \tag{6.5}$$

and the reduced radius is computed as:

$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_{i+1}}{N_i + M_{i+1}}}. \tag{6.6}$$

Due to this radius reduction, we must account for the flux of the photons that now fall outside the reduced radius. To account for this, the accumulated flux at the next iteration is computed as:

$$\tau_{i+1}(x,\vec{\omega}) = \tau'_{i+1}(x,\vec{\omega}) \frac{{R_{i+1}}^2}{R_i^2} = \tau'_{i+1}(x,\vec{\omega}) \frac{N_i + \alpha M_{i+1}}{N_i + M_{i+1}}. \tag{6.7}$$

Traditional photon mapping can be considered as a special case of progressive photon mapping where the radius $R_i(x)$ stays the same ($\alpha = 1.0$).

6.3 Error Estimation Framework

Stochastic error bounds in unbiased Monte Carlo rendering methods can be estimated using the variance of the estimate. The reason for this is that the error is caused purely by *noise* due to stochastic sampling. However, biased Monte Carlo rendering methods have *bias* in addition to noise. Bias is the difference between the estimated density without noise and the correct density. Since photon density estimation, including progressive photon mapping, is a biased rendering method, we need to take into account both bias and noise in the error estimate.

Note that there is a critical difference between a *stochastic* bound and a *deterministic* bound. A deterministic bound always correctly bounds the value, but a stochastic bound only correctly bounds with some probability. This probability is usually provided by the user as a desirable *confidence* in the bound. Standard variance-based methods in fact compute a stochastic bound and, unfortunately, there is little hope of obtaining a deterministic bound for Monte Carlo estimation. More information on this concept can be found in statistics textbook such as Hastie et al [49].

In biased methods, the error can be expressed as the sum of bias B_i of the estimate and noise N_i due to sampling (the exact bias is denoted as B_i since bias is changing at each iteration in progressive photon mapping):

$$L_i - L = B_i + N_i, \tag{6.8}$$

where L_i is the estimated radiance (i.e., density of photons) and L is the true radiance. This *bias-noise* decomposition has been used in prior work (such as [101]) as well and it is applicable to any biased Monte Carlo methods, including progressive photon mapping. Now, assuming we know bias B_i and sample variance V_i, the central limit theorem states that for an infinite number of samples:

$$\frac{L_i - L - B_i}{\sqrt{V_i}} \xrightarrow{d} N(0,1), \tag{6.9}$$

where $\xrightarrow{d}$ expresses that the *distribution* of the left-hand-side converges to the standard normal distribution $N(0,1)$. Note that this does not directly estimate error, but rather estimates the error distribution. Although this relationship is valid only for an infinite number of samples, it has been shown that assuming this is valid for a finite number of samples is reasonable in practice [112, 26]. This also indicates that the variance-based error estimation in unbiased methods is fundamentally an approximation unless we use an infinite number of samples. Our error estimation framework is also based on a few approximations.

Based on the error distribution, we can derive a stochastic bound (confidence interval) of radiance estimation using the t-distribution:

$$\begin{aligned} &P\left(-E_i < L_i - L - B_i < E_i\right) = 1 - \beta \\ &E_i(x) = t\left(i, 1 - \frac{\beta}{2}\right)\sqrt{\frac{V_i(x)}{i}}, \end{aligned} \tag{6.10}$$

where $t\left(i, 1 - \frac{\beta}{2}\right)$ is the $1 - \frac{\beta}{2}$ percentile of the t-distribution with degree i. The t-distribution describes the distribution of the quantity $\frac{L_i - L - B_i}{\sqrt{iV_i}}$. This gives the interval $[-E_i, E_i]$ that contains $L_i - L - B_i$ with probability $1 - \beta$. Note that this is a well-known derivation and interested readers should refer to, for example, Geertsema [32]. It is also important to note that the interval $[-E_i, E_i]$ is not for "error", but rather for "error - bias" at this point.

Shifting both the value $L_i - L - B_i$ and the interval $[-E_i, E_i]$ by adding B_i yeilds:

$$\begin{aligned} &P\left(-E_i < L_i - L - B_i < E_i\right) \\ &= P\left(-E_i + B_i < L_i - L < E_i + B_i\right). \end{aligned} \tag{6.11}$$

Note that the inequalities inside are equivalent. This operation is valid because bias B_i, by definition, is a deterministic variable since bias is a difference between an expected value and the correct value which are both deterministic. In other words, bias just affects error as a uniform, constant shift of the error distribution.

The above signed confidence-interval can be used as-is, however, it is sometimes desirable to obtain a single value as the stochastic bound of absolute error. Taking the absolute value of the bias yields such an expression as

$$\begin{aligned} &P\left(-E_i + B_i < L_i - L < E_i + B_i\right) \\ &\leq P\left(-E_i - |B_i| < L_i - L < E_i + |B_i|\right) \\ &= P\left(|L_i - L| < E_i + |B_i|\right), \end{aligned} \tag{6.12}$$

which provides $E_i + |B_i|$ as a single-valued stochastic error bound of the absolute error. This theoretically gives us a confidence interval with larger confidence than the user-specified confidence $1 - \beta$. Note that the above equations do not use the bound of bias, but bias itself.

Dependence of error on the absolute intensity of radiance is often not desirable since the human visual system is more sensitive to contrast than absolute differences in intensity [34]. Dividing the interval by the estimated radiance obtains the relative stochastic error bound:

$$P\left(|L_i - L| < E_i + |B_i|\right) = P\left(\left|\frac{L_i - L}{L_i}\right| < \frac{E_i + |B_i|}{L_i}\right). \tag{6.13}$$

The interval is $\left[0, \frac{E_i+|B_i|}{L_i}\right]$ with confidence larger than $1-\beta$. Note that the above Equation 6.8 to Equation 6.13 are theoretically valid only if we know the exact bias and the exact sampled variance.

The challenge here is that both the exact bias B_i and the exact sample variance V_i are unknown. We therefore need to rely on estimations of bias and variance. Estimation of bias B_i has been investigated by previous work, but none of them can be extended to progressive photon mapping. Moreover, we do not have an estimator of variance V_i in progressive photon mapping. To be concrete, progressive density estimation has a different formulation than standard density estimation and samples are correlated due to radius reduction. We therefore have to consider the correlation between samples when estimating variance. The following sections describe the key contributions of our work: estimators of B_i and V_i. Note that, in practice, we cannot obtain the exact stochastic bounds as defined above due to approximations in bias and variance estimators.

6.3.1 Bias Estimation

This section first shows that the bias in progressive density estimation is actually the average of bias induced by the kernel radii at each iteration

$$B_i = \frac{1}{i}\sum_{j=1}^{i} B_{p,j}, \tag{6.14}$$

where $B_{p,j}$ is bias of the radiance estimate using only the photons from the j^{th} pass. This section then introduces a bias estimator used in standard density estimation, which is applied for progressive photon mapping.

To obtain Equation 6.14, one can start by expanding the estimated radiance

$L_{j+1}(x)$ at the $j+1^{\text{th}}$ iteration using the refinement procedure described in Section 6.2:

$$\begin{aligned} L_{j+1}(x) &= \frac{\tau_{j+1}(x,\vec{\omega})}{\pi R_{j+1}(x)^2} = \frac{R_{j+1}(x)^2}{R_j(x)^2}\frac{\tau'_{j+1}(x,\vec{\omega})}{\pi R_{j+1}(x)^2} \\ &= \frac{j\tau_j(x,\vec{\omega}) + \tilde{\tau}_{j+1}(x,\vec{\omega})}{j+1}\frac{1}{\pi R_j(x)^2} \\ &= \frac{1}{j+1}\left(j\frac{\tau_j(x,\vec{\omega})}{\pi R_j(x)^2} + \frac{\tilde{\tau}_{j+1}(x,\vec{\omega})}{\pi R_j(x)^2}\right) \\ &= \frac{1}{j+1}\left(jL_j(x) + L_{p,j}(x)\right), \end{aligned} \tag{6.15}$$

where $L_{p,j}(x)$ is radiance estimate using only the photons from the j^{th} pass. Note that $\frac{\tilde{\tau}_{j+1}(x,\vec{\omega})}{\pi R_j(x)^2} = L_{p,j}(x)$ because $\tilde{\tau}_{j+1}(x,\vec{\omega})$ includes the intercepted photons only at the $j+1^{\text{th}}$ iteration *before* the radius reduction and the radius reduction happens after flux accumulation (i.e., the radius used in the photon query is $R_j(x)$).

The key observation is that $L_{p,j}$ is equivalent to the radiance estimate using a standard density estimation with radius R_j because it uses photons only from the j^{th} pass. Therefore, we can write down $\mathrm{E}[L_{p,j}] = L(x) + B_{p,j}$ based on the definition of bias in standard density estimation. Using Equation 6.15 recursively, the progressive density estimate $L_i(x)$ is expressed as:

$$\begin{aligned} \mathrm{E}\left[L_i(x)\right] &= \mathrm{E}\left[\frac{1}{i}\sum_{j=1}^{i} L_{p,j}(x)\right] = L(x) + B_i \\ &= \frac{1}{i}\sum_{j=1}^{i}\left(L(x) + B_{p,j}\right) = L(x) + \frac{1}{i}\sum_{j=1}^{i} B_{p,j}. \end{aligned} \tag{6.16}$$

Finally, subtracting $L(x)$ from both sides gives us Equation 6.14.

Silverman [105] showed that for standard density estimation bias can be approximated using the Laplacian of the density. We can apply this to each bias value $B_{p,j}$ as

$$\begin{aligned} B_{p,j} &= \frac{1}{2}R_j(x)^2 k_2 \nabla^2 L_{p,j}(x) + O\left(\sum_k \frac{\partial^k L_{p,j}(x_k)}{\partial x_k^k} R_j(x)^4\right) \\ &\approx \frac{1}{2}R_j(x)^2 k_2 \nabla^2 L_{p,j}(x), \end{aligned} \tag{6.17}$$

where $k_2 = \int t^2 K(t)\vec{d}t$ is a constant derived from the kernel $K(t)$ (see Appendix 6.8.1 for details). The last step drops the residual term proportional to $R_j(x)^4$, thus the bias estimator always estimates an approximation of the true bias.

Unfortunately, we cannot directly use this approximation for progressive photon mapping. This approximation needs to estimate $\nabla^2 L_{p,j}(x)$ at each iteration and combine Equation 6.17 and Equation 6.14 to compute the Laplacian of the progressive estimate $\nabla^2 L_i(x)$. However, estimating and storing $\nabla^2 L_{p,j}(x)$ at each iteration in progressive density estimation is not tractable since this would require unbounded memory and the method uses a relatively small number of photons per iteration. The following approximation of the Laplacian of the progressive estimate $\nabla^2 L_i(x)$ eliminates the need to store the Laplacian of each iteration.

First, substituting Equation 6.17 into Equation 6.14, we obtain

$$B_i = \frac{1}{i}\sum_{j=1}^{i} B_{p,j} \approx \frac{1}{i}\sum_{j=1}^{i} \frac{1}{2} R_j(x)^2 k_2 \nabla^2 L_{p,j}(x). \tag{6.18}$$

Approximating the summation as (note that we use $R_i(x)$ not $R_j(x)$) yields

$$\approx \frac{1}{2} R_i(x)^2 k_2 \left(\frac{1}{i} \sum_{j=1}^{i} \nabla^2 L_{p,j}(x) \right). \tag{6.19}$$

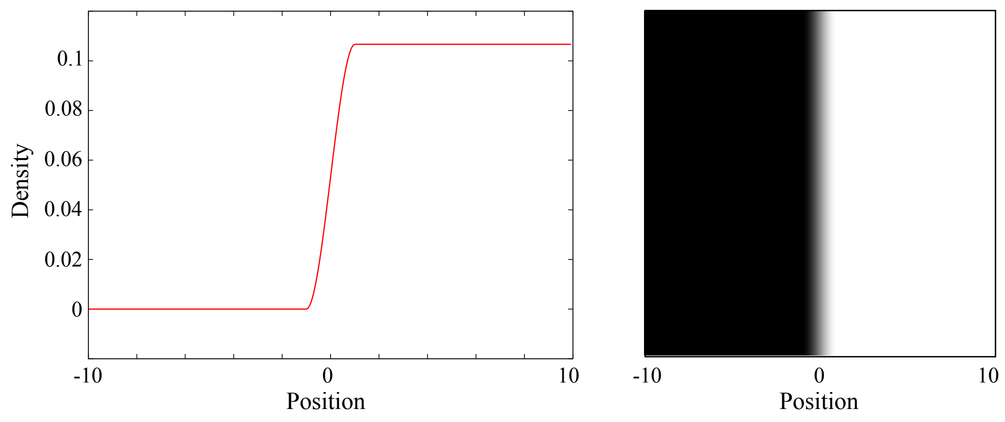
Finally, as with radiance estimation in Equation 6.16, we can expand the Laplacian of the progressive estimate as $\nabla^2 L_i(x) = \frac{1}{i}\sum_{j=1}^{i} \nabla^2 L_{p,j}(x)$ using the recursion relation in Equation 6.15. This provides the following identity

$$\begin{aligned} &= \frac{1}{2} R_i(x)^2 k_2 \left(\frac{1}{i} \sum_{j=1}^{i} \nabla^2 L_{p,j}(x) \right) \\ &= \frac{1}{2} R_i(x)^2 k_2 \nabla^2 L_i(x) \\ &= B_i' \approx B_i, \end{aligned} \tag{6.20}$$

where the approximation of bias is given as B_i'. As a result, we can approximate the exact bias B_i just by computing a progressive estimate of $\nabla^2 L_i(x)$. Section 6.4 will describe more details for how to estimate $\nabla^2 L_i(x)$.

We tested our bias estimation for progressive photon mapping on a scene where an analytic cubic smoothstep function mimics a soft shadow boundary. Figure 6.1(a) shows a visualization of the resulting radiance function while Figure 6.1(b) compares the actual bias to our estimated bias. In this scene, samples are generated using the cubic smoothstep function with importance sampling. Although the estimated bias does not exactly predict the actual bias, it captures two peaks of bias around edges of the smoothstep function as well as zero bias in the smooth region.

(a) Cubic ramp function



(b) Actual bias and estimated bias

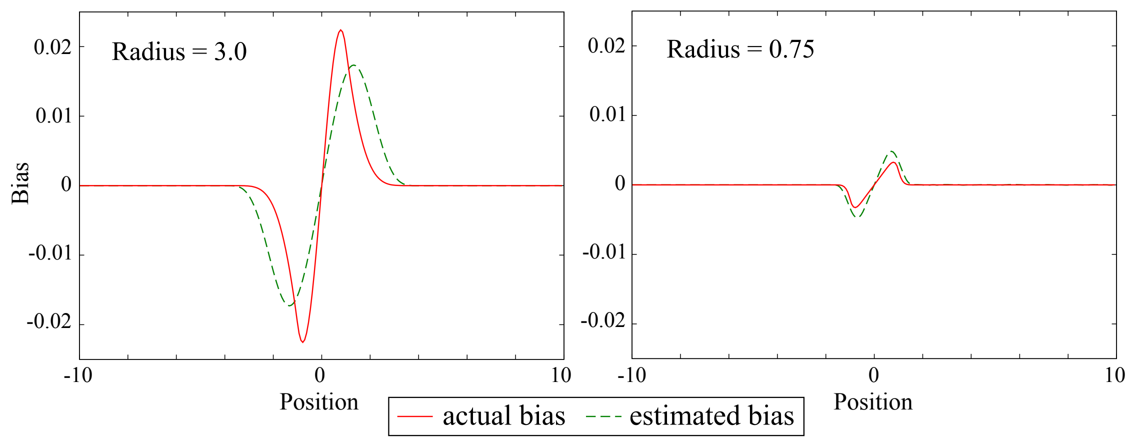


Figure 6.1: The graph and image in (a) show the cubic ramp function ($3t^2 - 2t^3$ with $t = \text{clamp}(\frac{x-1}{2})$ where $y = \text{clamp(x)}$ clamps x into $[0,1]$). The graphs in (b) show comparisons of the estimated bias and the actual bias in the cubic ramp scene. The bias estimate uses radius 3.0 and 0.75. The RMS errors of the bias estimation are 3.751×10^{-3} and 4.746×10^{-4}.

6.3.2 Variance Estimation

In standard kernel density estimation, variance is approximated as [105]:

$$V_i \approx \frac{1}{N_e^i R_i^2} k_3 L_i(x), \tag{6.21}$$

where $k_3 = \int K(t)^2 \vec{d}t$. This approximation assumes that samples are independent and identically distributed (i.i.d.). Since this is true in the case of traditional photon mapping, this approximation can be used directly for photon mapping.

Unfortunately, this is not directly usable in progressive density estimation. Note that the samples in progressive density estimation are each of the density estimations using samples (i.e., photons in light transport simulation) from *only* the j^{th} iteration, $L_{p,j}$. Since the radius reduction however depends on all previous photons, $L_{p,j}$ are *not statistically independent*. Furthermore, each $L_{p,j}$ gives us a biased estimation with a different radius R_j, thus $L_{p,j}$ are *not identically distributed.*

The key point is that photon tracing itself is statistically independent between each iteration in progressive photon mapping. The only dependence between samples is caused by the radius reduction. Therefore, instead of using samples ($L_{p,j}$) directly, our estimator uses bias-corrected samples $L_{p,j} - B_{p,j} \approx L_{p,j} - B'_j$. The bias-corrected samples are expected to be closer to statistically independent since dependence on radius is removed by $B_{p,j}$, and identically distributed because $L_{p,j} - B_{p,j}$ is distributed according to the true radiance L.

Using $L_{p,j} - B_j$, one can use the following standard procedure to estimate sample variance V_i at each iteration:

$$V_i \approx \frac{\sum_{j=1}^{i} {x_j}^2 - \left(\sum_{j=1}^{i} x_j\right)^2 / i}{i-1} \tag{6.22}$$

where $x_j = L_{p,j} - B'_j$ is a sample.

Note that $L_{p,j} - B'_j$ is not strictly unbiased or independent because B'_j is just an approximation of true bias. However, numerical experiments indicates that the effect of this approximation is not statistically significant in practice. Figure 6.2 shows that the autocorrelation of bias-corrected samples is almost 0 compared to the uncorrected samples. This indicates that the samples can be considered independent. Figure 6.2 furthermore shows that the bias correction makes the distribution of noise closer to the Gaussian distribution with mean 0, which is the distribution of ideal i.i.d. samples.

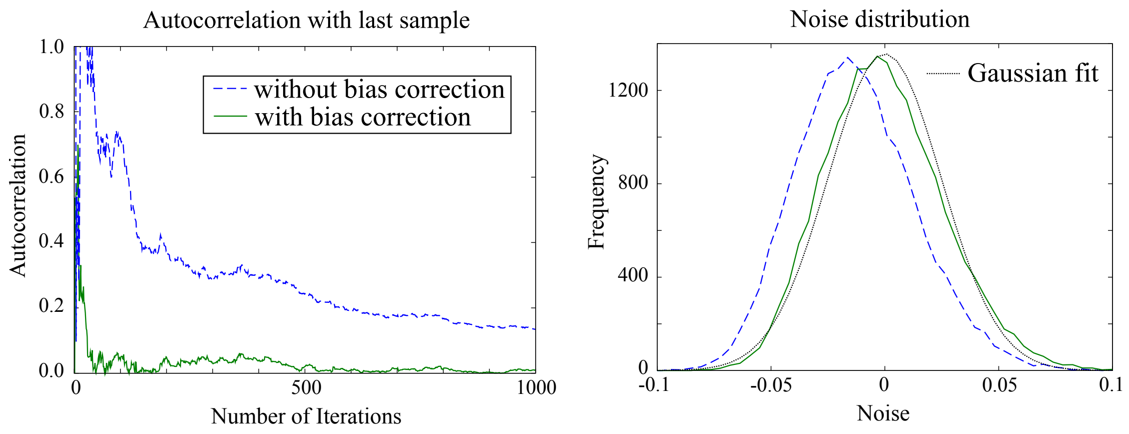


Figure 6.2: Autocorrelation with one previous sample and noise distribution in the cubic ramp test scene. This test measured autocorrelation and noise distribution on the point with largest bias. The graph on the left shows autocorrelation between one previous sample and current sample. The graph on the right shows the histogram of noise with 500000 independent runs. The Gaussian fit is using the sample variance of computed noise and mean 0.

6.4 Kernel-based Progressive Density Estimation

The previously described bias estimation method relies on the derivatives of radiance, $\nabla^2 L_i(x)$. To use this method, we must therefore estimate derivatives of radiance as well as radiance itself during rendering. In traditional photon mapping, derivatives can be easily estimated by just using derivatives of the kernel. However, estimation based on this method does not converge into the correct derivatives of radiance without using an infinite amount of memory (e.g. storing an infinite number of photons or using an infinite mesh tessellation).

This section provides a new, generalized formulation of progressive density estimation which allows for arbitrary smooth kernels. This generalized formulation shows that we can relax the "locally uniform density" assumption used in the original progressive photon mapping algorithm, and that the kernel should be radially symmetric and C^n continuous at the boundary to be able to estimate nth order derivatives. This generalization makes estimation of derivatives straightforward and convergent, which in turn allows us to estimate bias more accurately.

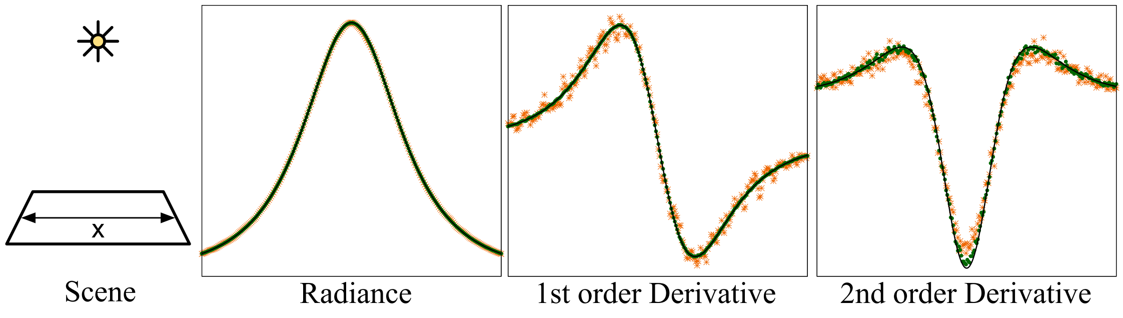


Figure 6.3: Visualizations of radiance including 1st and 2nd order derivatives for a simple scene containing a Lambertian plane and a point light source. The estimated values using progressive photon mapping with 10K (orange) and 1M (green) stored photons are compared to the analytic solutions (black). As more photons are used, the results converge to the analytic solutions.

6.4.1 Kernel-based Flux Correction

Kernel-based density estimation is a standard extension to traditional density estimation where the contribution of each sample in the density estimate is weighted by a, typically smooth, kernel K around the query location. In contrast to standard density estimation methods, simply replacing the constant kernel in the original progressive density estimate may not work. Specifically, since the radius in progressive density estimation changes, the accumulated flux needs to be corrected after radius reduction. To adapt kernel density estimation to progressive photon mapping, we need to derive an equation similar to Equation 6.7 (relating the accumulated flux before reducing the radius, τ'_{i+1}, with the accumulated flux after reducing the radius, τ_{i+1}) but in the presence of a weighting kernel K.

The accumulated flux before reducing the radius can be expressed in the continuous setting as

$$\tau'_{i+1} = \int_0^{2\pi}\int_0^{R_i} K\left(\frac{r}{R_i}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\, dr\, d\theta. \tag{6.23}$$

The above equation parametrizes radiance over a circle using polar coordinates. Note that τ'_{i+1} is a function of $(x, \vec{\omega})$, and R_i is a function of x, but the notation drops this dependence for simplicity.

The accumulated flux with a reduced radius R_{i+1} is:

$$\tau_{i+1} = \int_0^{2\pi}\int_0^{R_{i+1}} K\left(\frac{r}{R_{i+1}}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\, dr\, d\theta. \tag{6.24}$$

These two equations leads to a different expression of τ_{i+1} in terms of τ'_{i+1} as:

$$\tau_{i+1} = \tau'_{i+1} \frac{\int_0^{2\pi}\int_0^{R_{i+1}} K\left(\frac{r}{R_{i+1}}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\, dr\, d\theta}{\int_0^{2\pi}\int_0^{R_i} K\left(\frac{r}{R_i}, \theta\right) L_r(r, \theta, \vec{\omega})\, r\, dr\, d\theta}. \tag{6.25}$$

A change of variable with $t = \frac{r}{R_i}$, $dr = R_i\, dt$, and analogously for R_{i+1}, can simplify this expression:

$$\begin{aligned} \tau_{i+1} &= \tau'_{i+1} \frac{\int_0^{2\pi}\int_0^1 K(t, \theta)\, L_r(tR_{i+1}, \theta, \vec{\omega})\, t\, {R_{i+1}}^2\, dt\, d\theta}{\int_0^{2\pi}\int_0^1 K(t, \theta)\, L_r(tR_i, \theta, \vec{\omega})\, t\, {R_i}^2\, dt\, d\theta}, \\ &= \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\int_0^{2\pi}\int_0^1 K(t, \theta)\, L_r(tR_{i+1}, \theta, \vec{\omega})\, t\, dt\, d\theta}{\int_0^{2\pi}\int_0^1 K(t, \theta)\, L_r(tR_i, \theta, \vec{\omega})\, t\, dt\, d\theta}. \end{aligned} \tag{6.26}$$

Assuming that a kernel is radially symmetric $K(t, \theta) = K(t)$, the above expression can be further simplified by swapping the order of integration:

$$\tau_{i+1} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\int_0^1 K(t)\, t \int_0^{2\pi} L_r(tR_{i+1}, \theta, \vec{\omega})\, d\theta\, dt}{\int_0^1 K(t)\, t \int_0^{2\pi} L_r(tR_i, \theta, \vec{\omega})\, d\theta\, dt}. \tag{6.27}$$

Finally, the following assumption that the integrated radial distribution of radiance is locally constant

$$C = \int_0^{2\pi} L_r(tR_i, \theta, \vec{\omega})\, d\theta = \int_0^{2\pi} L_r(tR_{i+1}, \theta, \vec{\omega})\, d\theta, \tag{6.28}$$

for some constant C will further simplify Equation 6.27 to:

$$\tau_{i+1} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2} \frac{\cancel{\int_0^1 K(t)\, t\, C\, dt}}{\cancel{\int_0^1 K(t)\, t\, C\, dt}} = \tau'_{i+1} \frac{{R_{i+1}}^2}{{R_i}^2}. \tag{6.29}$$

This ends up being the same flux correction as in regular progressive photon mapping in Equation 6.7; however, it uses a weaker assumption and can be used with arbitrary radially symmetric kernels. Fortunately, it also means that the only change in terms of implementation is to weight each photon by a kernel function, thus it is straightforward to implement.

Note that this assumption is similar, but not as strict as the “locally uniform density” assumption used in the original progressive photon mapping algorithm. Although the end result looks similar to the original progressive photon mapping, this distinction is crucial to use a smooth kernel function as well as its derivatives. The original assumption implies zero derivatives, which makes it impossible to use for the derivative estimation in this framework.

6.4.2 Progressive Density Derivative Estimation

The kernel-based estimation gives us an elegant way to estimate derivatives. Computing the derivatives simply involves replacing the kernel with a corresponding derivative kernel. The following derivation describes how to compute only the first-order derivative for simplicity.

The first-order derivative of the radiance estimate can be computed directly as:

$$\nabla L(x,\vec{\omega}) \approx \nabla\left(\frac{\tau_i}{k_1 R_i^2}\right) = \frac{\nabla\tau_i}{k_1 R_i^2}, \tag{6.30}$$

where the derivative of the accumulated flux is:

$$\begin{aligned}\nabla\tau_i &= \frac{1}{N_i^{\mathrm{e}}}\nabla\left(\sum_{p=1}^{N_i} K(t)\, f_r(x,\vec{\omega},\vec{\omega}_p)\,\Phi_p(x_p,\vec{\omega}_p)\right)\\ &= \frac{1}{N_i^{\mathrm{e}}}\sum_{p=1}^{N_i}(\nabla K)\, f_r\,\Phi_p + K\,(\nabla f_r)\,\Phi_p \\ &= \frac{1}{N_i^{\mathrm{e}}}\sum_{p=1}^{N_i}(\nabla K)\, f_r\,\Phi_p.\end{aligned} \tag{6.31}$$

The gradient operator ∇ denotes a gradient with respect to the position parameter x. The $\nabla\Phi_p$ term from the product rule is omitted above because $\nabla\Phi_p = 0$. The ∇f_r term is also omitted since the BRDF within the radius is locally constant (i.e., BRDF is defined by the measurement point) thus $\nabla f_r = 0$.

Higher-order derivatives can be derived in exactly the same way by simply taking higher-order derivatives of the kernel function. The bias estimator in Equation 6.17 uses the second order derivatives $\nabla^2 L_i(x)$. Appendix 6.8.2 describes the details.

In order to estimate derivatives in a progressive manner, we need to express the accumulated flux derivatives after radius reduction, $\nabla\tau_{i+1}$, in terms of the accumulated value before radius reduction, $\nabla\tau'_{i+1}$. This relation can be established by differentiating Equation 6.29:

$$\nabla\tau_{i+1} = \nabla\left(\tau'_{i+1}\frac{R_i^2}{R_{i+1}^2}\right) = (\nabla\tau'_{i+1})\frac{R_i^2}{R_{i+1}^2}. \tag{6.32}$$

Hence, any order flux derivative is accumulated using the ratio of the squared radii, just like accumulated flux. Again, the only change in terms of implementation is replacing the kernel with derivatives of the kernel.

One condition on the kernel is that derivatives have to be finite in order for this computation to be meaningful. For example, if we use a Gaussian by cutting off the kernel to a finite radius, this gives us infinite derivatives along the edge of the kernel, which is not useful. The same thing happens with a Epanechnikov kernel, which is a popular choice in standard density estimation. The kernel has to be C^n continuous with 0 at the boundary for computing nth order derivatives (i.e., the nth derivative of the kernel must go to 0 at the edge). Appendix 6.8.1 provides an example of a kernel function that satisfies this condition.

In order to investigate the accuracy of the progressive kernel density estimate and the density derivative estimate, we used a simple scene with a point light source, where all the values can be computed analytically in Figure 6.3. The estimated radiance values match the analytical solution very closely with only 10K stored photons. Although the radiance derivative estimates are noisy with 10K stored photons, they are converging to the analytical solution with 1M stored photons.

6.5 Results

We implemented the error estimation on top of an implementation of the progressive photon mapping algorithm. All results were computed on a 2.4 GHz Intel Core 2 Q6600 using one core. All the examples used the parameter $\alpha = 0.8$ in the progressive photon mapping algorithm.

Figure 6.4 is an analytic investigation of our error estimation. This test used the cubic ramp scene described in Figure 6.1 with 640000 samples and an initial radius of 1.5. The user-specified confidence is $1-\beta = 0.95$. The left graph shows that the estimated

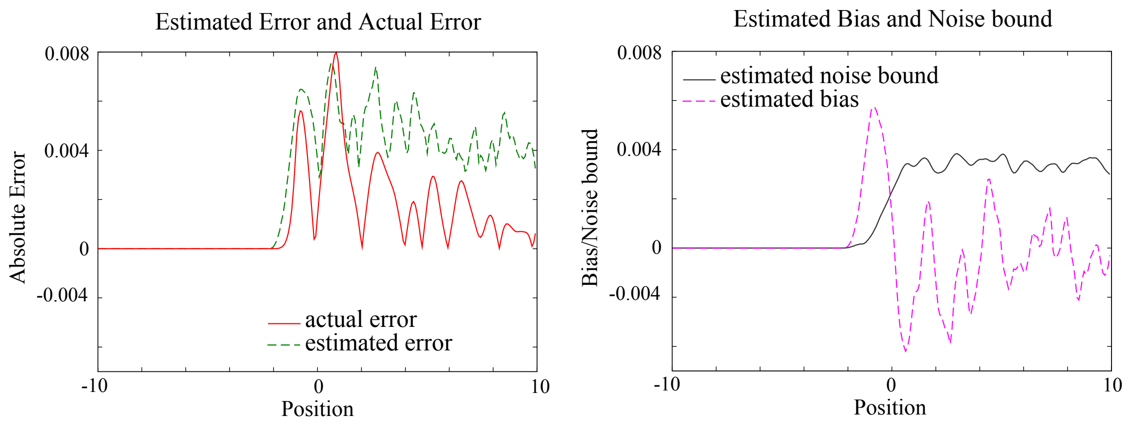


Figure 6.4: Error estimation on the cubic ramp test. The graphs show estimated error (95 %) and actual error, and estimated bias and estimated noise bound with 640000 samples.

error is in fact above the actual error in almost all locations as specified. The right graph further demonstrates the influence of bias estimation and noise bound estimation by plotting each of these components separately. Note that the proposed framework does not estimate bounds of bias, but bias itself. This graph shows that the bias estimation captures error due to bias around the slope of the function, whereas the noise bound estimation captures error due to sampling noise around the flat region of the function.

We tested our framework on three example scenes. The reference renderings are shown in Figure 6.7. Each iteration uses 15k emitted photons and the confidence was specified at 90 %. The first scene is a Cornell box with a glass sphere and a metal sphere. Here noise dominates the error because the scene is composed of mostly large flat regions of fairly uniform illumination. The second scene consists of three flashlights models. We modeled the filament, lens and reflector of the flashlight in order to simulate a realistic lighting pattern. This scene is expected to have larger bias due to sharp caustics from the flashlights. The third scene is a room with lighting fixtures. Each lighting fixture consists of a filament, a glass casing, and reflectors which replicates light transport of real-world lighting fixtures. This scene contains both sharp illumination features due to caustics as well as flat and continuous illumination, which is a realistic mixture of bias and noise in error. Note that both the flashlights scene and the room scene have light paths which are difficult for unbiased rendering methods to handle (such as an SDS path from a very small light source).

Figure 6.7 shows behavior of the actual error and the estimated error for the test scenes. We picked three points to investigate the actual error and the estimated error as shown in the leftmost column of Figure 6.7. One might notice that our estimated error does not bound actual error at some points, such as the point A in the room scene. This is indeed the expected behavior because our error estimation is motivated by stochastic error bounds. Stochastic error bounds, by definition, bound the actual error with some probability, so even if our error estimate is perfect, some points where the actual error is higher than the estimation are to be expected. In practice, since bias estimation is an approximation, it may also cause extra overestimation or underestimation of the actual error in addition to what is expected from exact stochastic error bounds.

What is important in practice is whether the estimated error respect the user-specified confidence. Figure 6.6 shows the computed actual confidence on the test scenes. The ideal behavior is that the actual confidence stays at the user-specified confidence all the time. We calculate the actual confidence as the percentage of all error estimates that are above the true errors obtained using the reference image. We tested 50 % and 90 % as the user-specified confidence. Although the calcuated confidence is not exactly the same as the specified confidence, the deviation from the user-specified confidence is within 5 % across many iterations for both 50 % and 90 % in difference scenes.

Figure 6.5 shows the ratio of estimated bias to estimated noise bound using 15M photons. The estimated error captures noise in flat regions as well as bias due to sharp features in practical scenes. The bias-to-noise ratio images demonstrate the importance of each of these components to our overall error estimate. The resulting images show bias estimation dominates the error estimation around sharp features (shown as red) and the noise bound estimation dominates the error estimation within flat regions (shown as green).

6.5.1 Rendering Termination

Predicting a sufficient total number of photons to obtain a desired quality for a particular scene is difficult. Unfortunately, this is currently set by tedious trial-and-error by the user. As an example application of our error estimation framework, this section proposes an automatic way to stop the rendering process without specifying the total

Table 6.2: The number of triangles in each scene and the number of iterations to reach the specified error threshold. Note that processing time for each iteration is different for each scene.

Scene	Triangles	$E^{\text{thr}} = 0.5$	$E^{\text{thr}} = 0.25$	$E^{\text{thr}} = 0.125$	$E^{\text{thr}} = 0.0625$
Cornell Box	9666	6	55	340	1969
Flashlights	38118	2	36	229	1410
Room	181168	13	93	570	3465

number of photons or the total render time. Instead, the rendering process stops when the average estimated error over the image is below a user specified threshold. This will provide a more meaningful termination criterion than the number of photons or the total rendering time, since it allows the user to focus on the quality of the resulting image and not abstract parameters involved in the rendering algorithm.

Each iteration simply computes the average estimated error over all the measurement points:

$$E^{\text{ave}} = \frac{1}{N_m} \sum_{j=1}^{N_m} \frac{E_i(x_j) + |B_i'(x_j)|}{L_i(x_j)}, \tag{6.33}$$

where N_m is the number of measurement points. The rendering process stops when $E^{\text{ave}} \leq E^{\text{thr}}$ for a given error threshold E^{thr}. Note that our error estimation can be applied to individually terminate the computation of *each* measurement point since the error is estimated per measurement point. Here, average error was used as a proof of concept that shows our error estimation is indeed useful for a practical application. Since we are interested in the average error, not per-pixel error, this application uses rather lower 50 % confidence in order to get tighter error estimation.

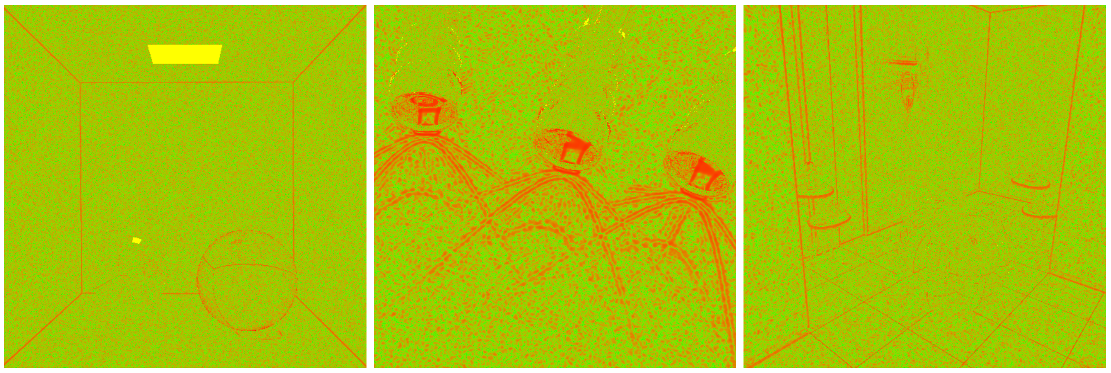


Figure 6.5: The ratio of the estimated bias and the estimated noise bound in error estimation with 15M emitted photons. The red pixel indicates the bias is dominant and the green pixel indicates the noise is dominant in the estimated error.

Figure 6.8(a) shows a sequence of color-coded images of actual error with different user-specified thresholds. The graphs of actual average relative error and the average of estimated error is in Figure 6.8(b). Although the actual rendering process terminated based on 50 % confidence, both graphs show also the resulting estimated average error using 90 % confidence for reference. The average of the error estimate using 50 % confidence successfully predicts, without any user input other than the threshold, that different scenes need a different number of iterations to obtain sufficient quality.

Table 6.2 shows the number of iterations used for achieving the specified threshold. Note that, even though the flashlight scene has a more complex scene configuration than the Cornell box scene, the number of iterations to achieve the same threshold is less than that of the Cornell box. The actual average relative errors achieved with $E^{\text{thr}} = 0.0625$ are 0.0465, 0.0448, 0.0437 for the Cornell box scene, the flashlight scene, and the room scene respectively. The statement that different scenes need different number of samples may sound trivial, however, note that we do not know how many samples are needed unless we actually render an image. The common belief that complex scenes need more computation is not necessarily true as Table 6.2 demonstrated.

Runtime overhead due to the error estimation is very small. In our implementation, each pass including the error estimation took 681 ms, 1291 ms, and 720 ms on average for the Cornell box scene, the flashlight scene, and the room scene respectively, and the overhead due to the error estimation are 2.2 ms, 5.9 ms, and 3.3 ms which are all less than 1 % of the rendering time without error estimation.

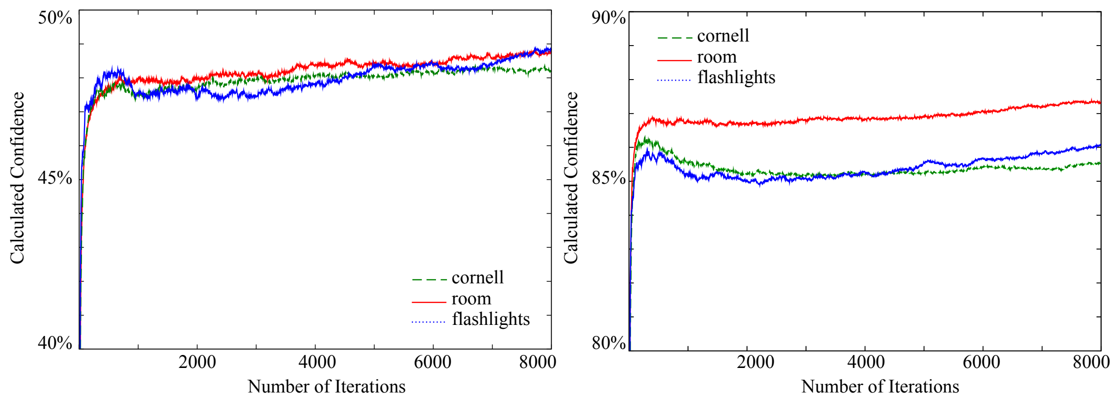


Figure 6.6: Calculated confidence of the estimated error in different scenes. The confidence is calculated as the percentage of estimated error that is actually larger than actual error. The graph on the left uses 50 % and the graph on the right uses 90 % as the user-specified confidence.

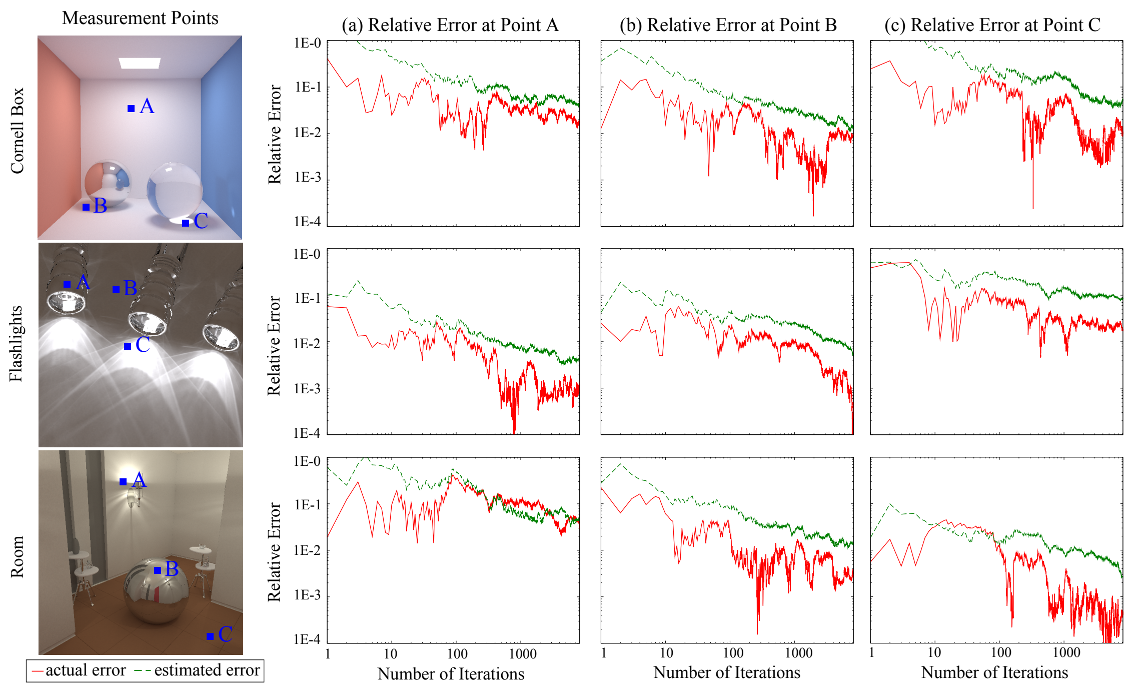


Figure 6.7: Error estimation on test scenes. Each row shows the reference rendering (left), as well as the actual error (red) and estimated error (green) at the three points (a,b,c) shown in the reference images. The specified confidence is 90 %, and each iteration uses 15k photons.

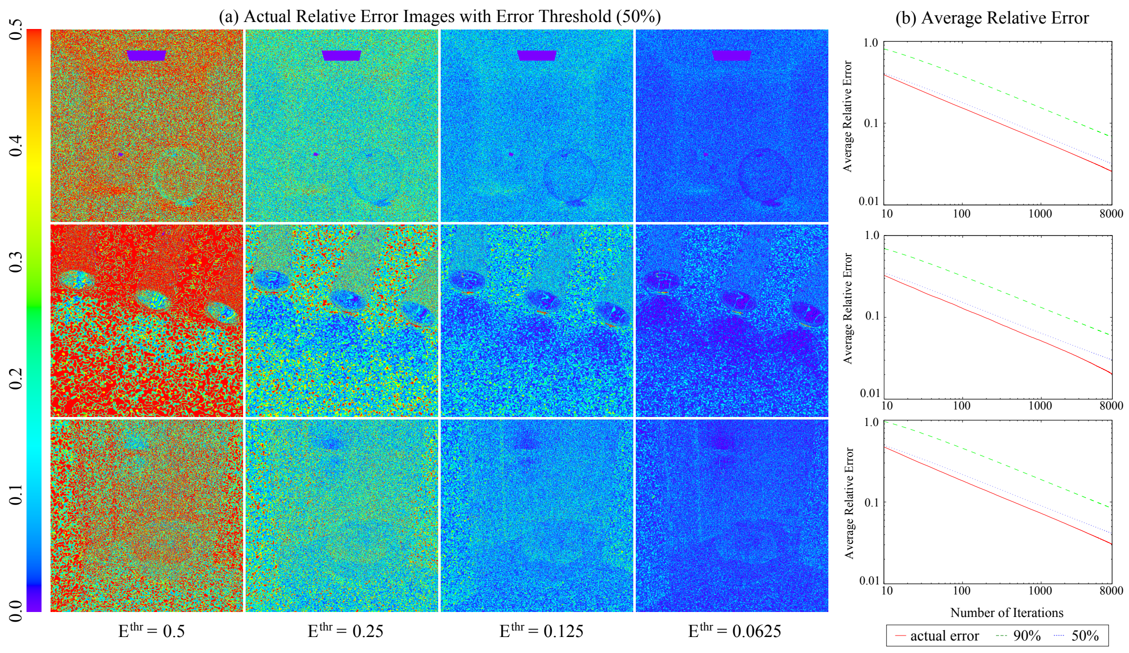


Figure 6.8: Color-coded relative error images with specified average estimated error. The confidence used is $1-\beta = 50$ % for estimating the average error. The rendering process of each column is terminated after the average estimated relative error estimate is smaller than the threshold which is shown below. The graphs on the right show actual average relative error (red) and the estimated average relative errors with different confidence values (green: 90 % confidence, blue 50 % confidence).

6.6 Discussion and Limitation

The proposed error estimation framework depends on the number of assumptions. In general, our experience along with previous work concludes that making assumptions or approximations is probably inevitable and providing theoretical guarantees that error estimation works is challenging. This section briefly discusses some issues due to those assumptions to help readers understand the limitations, rather than suggesting directions to remove those assumptions.

First, the derivations in Equation 6.10 are exact only if we know the true bias and variance. Since the framework approximates bias and variance in our error estimation with a finite number of samples, the resulting error estimation can be arbitrary far away from the exact error. This might seem to be a critical flaw as an error estimation framework, however, the variance-based error estimation have also made the approximation of assuming normal distribution of error which is not always valid.

Even with such an approximation, variance-based error estimation has been used in many practical applications (Section 6.1). The fact that there are approximations therefore do not render the proposed error estimation completely useless in practice. However, one should be careful that the framework does not well in all possible scene settings, and there should be cases where the estimated error is completely useless. Theoretically characterizing when the proposed estimation fails or succeeds, however, is not trivial. This could be a fundamental limitation of any error estimation framework as one could only estimate error from a finite amount of information and if the full information of error is available, one could just correct any intermediate solution to be equal to the correct solution.

Second, related to the above issue, the proposed bias estimation is not a consistent estimator of the true bias (i.e., does not give us the true bias in the limit). This in turn affects variance estimation, which assumes i.i.d. samples after bias correction. Equation 6.17 approximates bias using the Laplacian of radiance and one additional approximation between Equations 6.18 and 6.19 is also needed in order to make bias estimation practical. The approximation by Laplacian can cause significant error in bias estimation as either underestimation or overestimation of the correct bias, if the radiance distribution contains significant higher order derivatives (higher or equal to 4th order) and the radius is relatively large. However, again, there seems to be little hope of obtaining correct bias estimation for any biased estimator in general. It is however interesting to investigate if alternative ways to estimate bias improves the accuracy of error estimation in practice.

Finally, the application of error estimation for automatic rendering termination may not be well-suited for some usage. For example, if we want to avoid generating redundant samples by terminating the rendering process after the desired rendering error is achieved, our application will result in oversampling due to slight overestimation of the rendering error. In such cases, more accurate error estimation is needed to avoid oversampling. Although our overestimation is generally around 1.4 times throughout many iterations in our experiments as in Section 6.5.1, this could mean taking at least $1.4^2 = 1.96$ times more samples than absolutely necessary, according to $O(i^{-\frac{1}{2}})$ error convergence rate of Monte Carlo integration. Not generating redundant samples would require the estimated error to be equal to the exact error, which is challenging to accomplish in general.

6.7 Conclusion

This chapter introduced an error estimation framework for progressive density estimation in the context of light transport simulation. Based on the derivations of stochastic error bounds, we characterized error using estimated bias and an estimated noise bound. The proposed framework estimates bias using derivatives of radiance, and for this purpose we have extended the progressive density estimate to work with arbitrary smooth kernels. This kernel-based approach allows us to compute the derivatives of radiance by using the derivatives of the kernel. This bias estimate also constructs a variance estimator for the progressive density estimate. The estimated error captures error due to both bias and noise under complex illumination given user-defined confidence. The results demonstrated that the proposed error estimate can be used to automatically stop the rendering process.

6.8 Appendix

6.8.1 Kernels

In order to compute the first and second order derivatives, the kernel function must be at least twice differentiable. We use a sixth-order smooth kernel [94] for the progressive kernel density estimation. The kernel is defined as:

$$K(t) = 1 - 6t^5 + 15t^4 - 10t^3. \tag{6.34}$$

Since K is a polynomial, its derivatives can easily be computed as:

$$K'(t) = -30t^4 + 60t^3 - 30t^2, \qquad K''(t) = -120t^3 + 180t^2 - 60t.$$

Note that this kernel and its derivatives become 0 at $t = 1$. This property is very important in progressive density estimation as we reduce radius. If the kernel value does not become

0 at $t = 1$, each radius reduction incorrectly cut down energy accumulated inside the radius. The normalization constants are $k_1 = \frac{2\pi}{7}$, and $k_2 = \frac{10\pi}{168}$.

6.8.2 Derivatives of the Kernel

First-Order Derivatives: To compute the density derivatives, we need to evaluate the *spatial* derivatives of the kernel $\nabla K(t) = \left[\frac{\partial K(t)}{\partial x}, \frac{\partial K(t)}{\partial y}, \frac{\partial K(t)}{\partial z}\right]$. The partial derivative with respect to the x-component can be computed using the chain rule as follows:

$$\frac{\partial K(t)}{\partial x} = \frac{\partial K(t)}{\partial t}\frac{\partial t}{\partial x} = K'(t)\frac{\partial t}{\partial x}. \tag{6.35}$$

The first term is the derivative of the kernel itself. Appendix 6.8.1 discusses this point in more detail.

To compute the second term, we substitute the definition $t = \frac{\|x-x_p\|}{R}$, which results in:

$$\frac{\partial t}{\partial x} = \frac{1}{R}\frac{\partial \|x-x_p\|}{\partial x} = \frac{1}{R}\frac{x-x_p}{\|x-x_p\|}. \tag{6.36}$$

Note that in reality the radius R depends on the position x, but the above expression assumes that it is constant for the sake of computing derivatives.

The partial derivatives with respect to the other axes are computed analogously, resulting in the following compact vector-valued expression:

$$\nabla K(t) = \frac{K'(t)}{R}\frac{x-x_p}{\|x-x_p\|}. \tag{6.37}$$

Second-Order Derivatives: Estimation of the bias and error needs computation of the Laplacian of the density, which involves deriving second-order derivatives of the kernel function, $\nabla^2 K(t) = \frac{\partial^2 K(t)}{\partial x^2} + \frac{\partial^2 K(t)}{\partial y^2} + \frac{\partial^2 K(t)}{\partial z^2}$. The bias estimation does not require cross derivatives, but the same principal can be applied for estimating cross derivatives. To compute the second-order derivatives, we start by differentiating the x-component of the first order derivatives from Equation 6.37 and then applying the product and chain rules:

$$\begin{aligned}\frac{\partial^2 K(t)}{\partial x^2} &= \frac{\partial}{\partial x}\left(\frac{K'(t)}{R}\frac{x-x_p}{\|x-x_p\|}\right), \\ &= \frac{1}{R}\left(\frac{K''(t)(x-x_p)^2}{\|x-x_p\|^2 R} + K'(t)\frac{\|x-x_p\|^2-(x-x_p)^2}{\|x-x_p\|^3}\right).\end{aligned} \tag{6.38}$$

The second-order derivatives with respect to the y- and z-components are computed analogously.

6.8.3 Alternative Approaches for Estimating Derivatives

Instead of using a smooth kernel, we can also apply the idea of finite differences to density estimation with the superposition of multiple constant kernel functions to estimate derivatives.

In finite differences, first order derivative and second order derivative of the function $L(x)$ are approximated as follows:

$$L'(x) \approx \frac{L(x+\Delta x) - L(x-\Delta x)}{2\Delta x}, \tag{6.39}$$

$$L''(x) \approx \frac{L(x+\Delta x) + L(x-\Delta x) - 2L(x)}{\Delta x^2}, \tag{6.40}$$

where Δx is a small value often called *step size* in finite differences. We use a 1D example for simplicity, but the result would extend to higher dimensional estimates. The basic idea is to estimate $L(x+\Delta x)$ and $L(x-\Delta x)$ using density estimation, such that Δx goes to 0 as we add more samples. To estimate $L(x+\Delta x)$ and $L(x-\Delta x)$, we use the following kernels:

$$K_1^+(t) = \begin{cases} 1 & (h > t \geq 0) \\ 0 & (t < 0), \end{cases} \tag{6.41}$$

$$K_1^-(t) = \begin{cases} 1 & (h > -t \geq 0) \\ 0 & (-t > 0), \end{cases} \tag{6.42}$$

We omitted the normalization constant for clarity. Based on these kernels, $L(x+\frac{h}{2})$ is estimated using K_1^+, and $L(x-\frac{h}{2})$ is estimated using K_1^-. Note that the position that estimate $L(x)$ depends on h with this kernel. We can think of density estimation as a method to compute the average density over the kernel support. The position that we estimate the average value is the centroid of the constant kernel function. Given estimations of $L(x+\frac{h}{2})$ and $L(x-\frac{h}{2})$, $L'(x)$ is computed just like using finite differences:

$$L'(x) \approx \frac{L(x+\frac{h}{2}) - L(x-\frac{h}{2})}{h}. \tag{6.43}$$

We can apply the same idea to compute second order gradient. In addition to the kernel to estimate $L(x)$, we use the following kernel:

$$K_2(t) = \begin{cases} 1 & (3h > |t| \geq 0) \\ 0 & (3h \leq |t|). \end{cases} \tag{6.44}$$

We can consider that density estimation using this kernel gives us $\frac{L(x+h)+L(x-h)+L(x)}{3}$. Since we already have the estimate of $L(x)$ using the standard progressive density estimate, the second order derivative is computed as

$$\begin{aligned} L''(x) \approx& 3\frac{\frac{L(x+h)+L(x-h)+L(x)}{3} - L(x)}{h^2} \\ =& \frac{L(x+h)+L(x-h)-2L(x)}{h^2}. \end{aligned} \tag{6.45}$$

In order to get correct results, we also need to modify the flux correction. Flux for $L'(x)$ should be multiplied by $\left(\sqrt{\frac{N+\alpha M}{N+M}}\right)^3$, and flux for $L''(x)$ should be multiplied by $\left(\sqrt{\frac{N+\alpha M}{N+M}}\right)^4$, because there is division by h^3 for $L'(x)$ and h^4 for $L''(x)$.

To estimate density in 3D, we set the shape of K_1^+ and K_1^- to be a half circle and the shape of K_2 to be a rectangular aligned to the direction of derivative (i.e., elongated along the direction of derivative). Note that we need to use the centroid of half circle to estimate h for the first order derivative. We found this approach works as well as the derivative kernels approach described in the previous sections, though we did not employ this approach in any of the results demonstrated in this chapter.

6.9 Acknowledgements

The material in this chapter is, in part, a reproduction of the material published in Toshiya Hachisuka, Wojciech Jarosz, and Henrik Wann Jensen. “A Progressive Error Estimation Framework for Photon Density Estimation”, ACM Transactions on Graphics, 29(6), 144:1–144:12, 2010 [41]. The dissertation author was the primary investigator and author of this paper. We would like to thank Youichi Kimura (Studio Azurite) for providing us the flashlight model and the room model. The dissertation author was supported by the 2008 ATI Fellowship Program.

Chapter 7

Stochastic Progressive Density Estimation

This chapter extends progressive photon mapping for simulating global illumination with effects such as depth-of-field, motion blur, and glossy reflections. We introduce a new formulation of progressive density estimation, *stochastic progressive density estimation*, which makes it possible to compute the correct average density value for a region. The key idea is to use shared statistics within the region rather than isolated statistics at a point. The resulting light transport algorithm algorithm, stochastic progressive photon mapping, is easy to implement and efficiently handles scenes with distributed ray tracing effects with the robustness of progressive photon mapping for difficult lighting configurations.

7.1 Related Work

The problem we are dealing with in this chapter is the computation of the average density/radiance value over an unknown region. The shape of the region is unknown before computation, meaning there is not explicit representation of the shape. Such a problem arises when distributed ray tracing [17] is used for adding depth-of-field, motion blur, and glossy reflections/refractions. For example, with motion blur, each pixel compute the average radiance that is coming toward this pixel over time. A set of the points that contribute to this pixel forms the region for computing the average. Unbiased Monte Carlo ray tracing methods can include these effects without changing the algorithms; however, this class of methods is not robust to complex illumination settings as Chapter 5 already demonstrated. This chapter extends progressive photon mapping in order to develop a new rendering algorithm that is robust to the combination of complex illumination settings and distributed ray tracing effects.

Computing the average density/radiance over an unknown region is not a typical problem setting in density estimation methods outside graphics. We are not aware of existing work in density estimation literatures that deals with this kind of problem settings outside graphics (refer to [105, 128] for example). In computer graphics, there are few related methods that extend the density estimation for computing average density values. Time dependent photon mapping [12] computes the average density of photons over time by extending the space of density estimation into 4D (3D position and time). The beam radiance estimate [60] computes the integration of density values on a line of sight, which is essentially a weighted average of the density values, in order to accelerate rendering of participating media. Unfortunately, these methods still store photons in a photon map similar to the standard photon mapping, thus suffering from unbounded memory consumption for the correct solutions. On the other hand, the proposed method in this chapter can compute the correct average density values with bounded memory consumption.

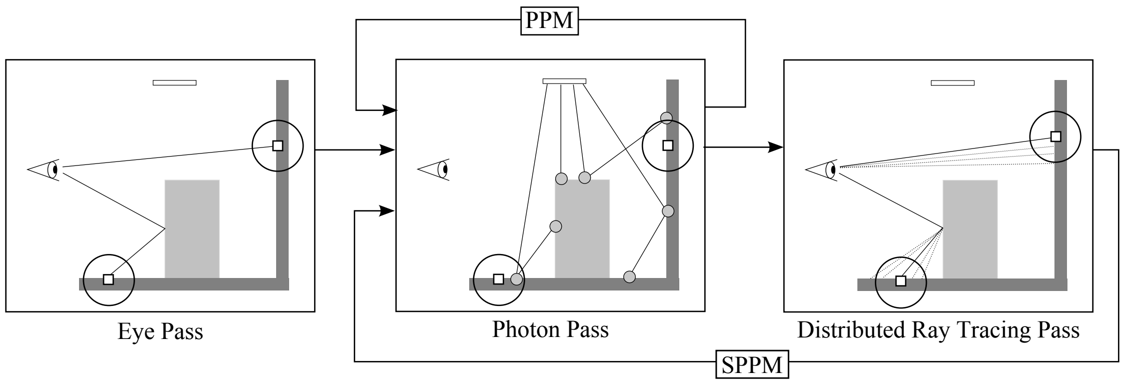


Figure 7.1: Difference between the algorithms of progressive photon mapping (PPM) and stochastic progressive photon mapping (SPPM). In order to compute the average radiance values, SPPM adds a new distributed ray tracing pass after each photon tracing pass. The photon tracing algorithm itself stays the same, but PPM uses a fixed set of hit points (shown as squares), whereas SPPM uses randomly generated hit points by the distributed ray tracing pass.

7.2 Overview

The motivation is that we need to compute the average density of photons over a region in order to simulate distributed ray tracing effects. For example, motion blur requires computing the average radiance value over a visible part of a scene for a given shutter time, and depth-of-field needs the average radiance value over a part of scene that is visible through a lens. The original progressive density estimate is restricted to computing the correct density of photons at a point $\vec{x}$. The next section describes why this is inconvenient for computing the average density values.

The key idea is to use shared statistics over a region that we would like to compute the average density value for. Using the shared statistics, the stochastic progressive density estimate approximates the average density of photons (i.e., radiance) $L(S, \vec{\omega})$ over the region S as:

$$L(S, \vec{\omega}) \approx \frac{\tau_i(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2}, \tag{7.1}$$

where i is the number of photon passes as before, $\tau_i(S, \vec{w})$ is the shared accumulated flux over the region S, and $R_i(S)$ is the shared search radius. The updating procedure of the shared statistics is:

$$N_{i+1}(S) = N_i(S) + \alpha M_i(\vec{x}_i) \tag{7.2}$$

$$R_{i+1}(S) = R_i(S)\sqrt{\frac{N_i(S) + \alpha M_i(\vec{x}_i)}{N_i(S) + M_i(\vec{x}_i)}} \tag{7.3}$$

$$\Phi_i(\vec{x}_i, \vec{\omega}) = \sum_{p=1}^{M_i(\vec{x}_i)} f_r(\vec{x}_i, \vec{\omega}, \vec{\omega}_p)\Phi_p(\vec{x}_p, \vec{\omega}_p) \tag{7.4}$$

$$\tau_{i+1}(S, \vec{\omega}) = (\tau_i(S, \vec{\omega}) + \Phi_i(\vec{x}_i, \vec{\omega}))\frac{R_{i+1}(S)^2}{R_i(S)^2}, \tag{7.5}$$

where $\vec{x}_i$ is a randomly generated position within S and $N_i(S)$ is the shared local photon count. Note that the updating procedure is the same as before, except that our formulation uses a randomly generated position $\vec{x}_i$ in S. The next section describes why this change allows to estimate the average radiance value over S. We call this extension as stochastic progressive density estimation since the algorithm stochastically generates a random position $\vec{x}_i$ in S.

The only algorithmic change from the original progressive photon mapping is that each hit point is randomly generated within the region S by distributed ray tracing after each photon pass. Figure 7.1 summarizes the difference between the algorithms of progressive photon mapping and the extended algorithm. Even though the modification is simple, the stochastic progressive density estimate converges to the correct average density over S for $i \to \infty$ without the explicit knowledge of S in advance:

$$L(S, \vec{\omega}) = \lim_{i \to \infty} \frac{\tau_i(S, \vec{\omega})}{N_e(i)\pi R_i(S)^2}. \tag{7.6}$$

Note that stochastic progressive density estimation turns into the original progressive density estimation if S is a point.

7.3 Extended Formulation using Shared Statistics

In order to explain the new formulation, we first describe how to compute the average radiance value using the original progressive density estimate. Suppose that we have n sampled positions over the region S, $\vec{x}_1, \ldots, \vec{x}_n$. Using a Monte Carlo integration method, the original progressive density estimate can approximate the average density of photons in the region S as:

$$\begin{aligned} L(S,\vec{\omega}) &= \frac{1}{\|S\|}\int_S L(\vec{x},\vec{\omega})\,\mathrm{d}\mu(S) \\ &= \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n} L(\vec{x}_k,\vec{\omega}) \approx \frac{1}{n}\sum_{k=1}^{n} L(\vec{x}_k,\vec{\omega}) \\ &= \frac{1}{n}\sum_{k=1}^{n}\lim_{i\to\infty}\frac{\tau_i(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(\vec{x}_k)^2}. \end{aligned} \tag{7.7}$$

We consider uniform sampling of $\vec{x}_k$ for brevity of the discussion. Using non-uniform sampling follows the same discussion with premultiplied weight for $L(\vec{x}_k,\vec{\omega})$.

This Monte Carlo integration approach is not scalable for a large n because the progressive density estimate needs to keep track of statistics at each radiance sample (i.e., storing n sets of statistics in total). Moreover, the memory requirement for computing the correct average radiance value again becomes unbounded because n needs to be infinite. The goal of stochastic progressive density estimation is to compute the correct average density value without storing infinite sets of photon statistics. The rest of this section describes how our formulation achieves this goal.

The extended formulation assumes that the initial radius R_0 is constant within S, and the value of α is also constant within S. R_0 and α can still vary between different S (e.g., different R_0 per pixel). In addition, the derivation in the following only consider non-adaptive photon tracing. To be precise, we assumed that the photon tracing strategy is not affected by the photon statistics. Note that it does not mean the photon tracing cannot use anything other than uniform random sampling, since importance sampling is still considered non-adaptive under this definition. Under these assumptions, we obtain the following equation:

$$\begin{aligned} R_{i+1}(\vec{x}) &= R_i(\vec{x})\sqrt{\frac{N_i(\vec{x})+\alpha M_i(\vec{x})}{N_i(\vec{x})+M_i(\vec{x})}} \\ &= R_i(\vec{x})\sqrt{\frac{C_N L(\vec{x})R_i(\vec{x})^2+\alpha C_M L(\vec{x})R_i(\vec{x})^2}{C_N L(\vec{x})R_i(\vec{x})^2+C_M L(\vec{x})R_i(\vec{x})^2}} \\ &= R_i(\vec{x})C_P, \end{aligned} \tag{7.8}$$

where C_N, C_M, and C_P are constants independent of $\vec{x}$. This equation states that the rate of radius reduction is independent of the position $\vec{x}$ in S, thus $R_i(\vec{x})$ itself is also independent of $\vec{x}$ if R_0 is constant. We used the property that the number of new photons and local photon count $N_i(\vec{x})$ are both proportional to the search area $\pi R_i(\vec{x})^2$ and its true radiance $L(\vec{x})$ (or true photon density in general). In practice, this equation is only approximately true because $N_i(\vec{x})$ and $M_i(\vec{x})$ are stochastic variables. However, we found that it is reasonably true as we see that the reduction rate of the radius is almost constant in the corresponding graph of radius in Chapter 5 (Figure 5.3). The following work by Knaus and Zwicker [70] also theoretically confirmed that this is a valid assumption.

7.3.1 Shared Radius

Based on the above observation, we can use a single radius value $R_i(\vec{x}_0)$ instead of $R_i(\vec{x}_k)$, in order to compute the average.

$$\frac{1}{n}\sum_{k=1}^{n}\lim_{i\to\infty}\frac{\tau_i(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(\vec{x}_k)^2}=\frac{1}{n}\sum_{k=1}^{n}\lim_{i\to\infty}\frac{\tau_i(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(\vec{x}_0)^2}. \tag{7.9}$$

This formulation removed the dependency of $R(\vec{x})$ on $\vec{x}$. We now describe how the dependency on an arbitrary location $\vec{x}_0$ can be further removed by using the shared radius $R_i(S)$ from Equation 7.3. Using the shared radius, the average radiance value is computed as:

$$\begin{aligned} L(S,\vec{\omega}) &= \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}L(\vec{x}_k,\vec{\omega}) \\ &= \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\lim_{i\to\infty}\frac{\tau_{R(S),i}(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(S)^2}. \end{aligned} \tag{7.10}$$

where $\tau_{R(S),i}(\vec{x}_k,\vec{\omega})$ is modified accumulated flux by changing the radius to $R_i(S)$. We show how this equation is derived in the following. Since accumulated flux is proportional to the area of the search region $\pi R_i(\vec{x}_0)^2$, $\tau_{R(S),i}(\vec{x}_k,\vec{\omega})$ is defined as:

$$\tau_{R(S),i}(x_k,\vec{\omega})=\frac{R_i(S)^2}{R_i(x_0)^2}\tau_i(\vec{x}_k,\vec{\omega}). \tag{7.11}$$

Using this equation, we obtain:

$$\begin{aligned}
\lim_{i\to\infty} \frac{\tau_{R(S),i}(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(S)^2} &= \lim_{i\to\infty} \frac{R_i(S)^2}{R_i(\vec{x}_0)^2} \frac{\tau_i(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(S)^2} \\
&= \lim_{i\to\infty} \frac{R_i(S)^2}{R_i(\vec{x}_0)^2} \frac{R_i(\vec{x}_0)^2}{R_i(S)^2} \frac{\tau_i(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(\vec{x}_0)^2} \\
&= C_R \frac{1}{C_R} L(\vec{x}_k,\vec{\omega}) = L(\vec{x}_k,\vec{\omega}).
\end{aligned} \tag{7.12}$$

The last step is valid only if $\lim_{i\to\infty} R_i(S)^2/R_i(\vec{x}_0)^2 = C_R$ with a non-zero constant C_R. In other words, $R_i(S)$ should be reduced on the same order of $R_i(\vec{x}_0)$ in order this step to be valid. We provide the derivations that show this is true in Appendix A. Note that an arbitrary $R_i(S)$ does not necessarily satisfy this condition, and our choice of $R_i(S)$ is crucial in this step. As a result of this derivation, we can replace each radius $R_i(\vec{x}_k)$ by a single shared radius $R_i(S)$, and its estimated radiance value still converges to the correct radiance value at $\vec{x}_k$. The shape of search region for range queries is still a sphere and its radius is defined by the shared radius $R_i(S)$.

7.3.2 Shared Accumulated Flux

Equation 7.10 still needs storing $\tau_i(\vec{x}_k,\vec{\omega})$ at each $\vec{x}_k$ in order to compute the correct average radiance value over S. This approach needs to keep track of an infinite number of accumulated flux values and positions over S, which is infeasible. Approximation using a fixed number of $\tau_i(\vec{x}_k,\vec{\omega})$ is possible, but this approach is not consistent. The shared accumulated flux value in Equation 7.5 solves this problem by storing a single accumulated flux value with a random position $\vec{x}_i$ at each photon pass. This section describes how this can be achieved. Replacing $\tau_i(\vec{x}_k,\vec{\omega})$ by the shared accumulated flux $\tau_i(S,\vec{\omega})$, we obtain the following estimate:

$$L(S,\vec{\omega})' = \lim_{i\to\infty} \frac{\tau_i(S,\vec{\omega})}{N_e(i)\pi R_i(S)^2}. \tag{7.13}$$

Table 7.1: Rendering statistics of the experiments. We show a single rendering time for each scene because both PPM and SPPM used the same rendering time. PPM passes are the numbers of photon tracing passes, and SPPM passes are the numbers of photon tracing passes as well as the numbers of distributed ray tracing passes. We used 500,000 emitted photons per pass in both methods.

Scene	Triangles	Time [min]	PPM	SPPM
Cornell Box	38	50	899	742
Cornell Box with Wall lights	7660	50	234	220
Furry Bunny	371247	132	1722	1197
Transparent Dices	370860	110	287	195
Alarm clocks	119856	480	1101	1202
Tools	56486	200	353	484

In order to show $L(S,\vec{\omega})' = L(S,\vec{\omega})$, we take the difference as:

$$\begin{aligned}
&L(S,\vec{\omega}) - L(S,\vec{\omega})' \\
&= \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\left(\lim_{i\to\infty}\frac{\tau_{R(S),i}(\vec{x}_k,\vec{\omega})}{N_e(i)\pi R_i(S)^2}\right) - \frac{\tau_{R(S),i}(S,\vec{\omega})}{N_e(i)\pi R_i(S)^2} \\
&= \lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\lim_{i\to\infty}\frac{\tau_{R(S),i}(\vec{x}_k,\vec{\omega}) - \tau_i(S,\vec{\omega})}{N_e(i)\pi R_i(S)^2} \\
&= \lim_{n\to\infty}\lim_{i\to\infty}\frac{E_i}{N_e(i)\pi R_i(S)^2},
\end{aligned} \tag{7.14}$$

where we defined $E_i = 1/n\sum_{k=1}^{n}(\tau_{R(S),i}(\vec{x}_k,\vec{\omega}) - \tau_i(S,\vec{\omega}))$. This difference converges to zero if the denominator $N_e(i)\pi R_i(S)^2$ diverges with an infinite number of passes and $|E_{i|}$ is bounded. The former is true because $\lim_{i\to\infty} N_e(i)\pi R_i(\vec{x}_0)^2$ is divergent (one of the conditions of consistency) and $R_i(S)^2/R_i(\vec{x}_0)^2$ is a non-zero constant as in Section 4.1., which shows that $\lim_{i\to\infty} N_e(i)\pi R_i(S)^2$ is also divergent. We provide the details how $|E_i|$ is bounded in Appendix B. Although the final result looks simple, our choice of $\tau_i(S,\vec{\omega})$ is again crucial to bound $|E_i|$. Finally, we can estimate the correct average radiance value as:

$$L(S,\vec{\omega}) = L(S,\vec{\omega})' = \lim_{i\to\infty}\frac{\tau_i(S,\vec{\omega})}{N_e(i)\pi R_i(S)^2}. \tag{7.15}$$

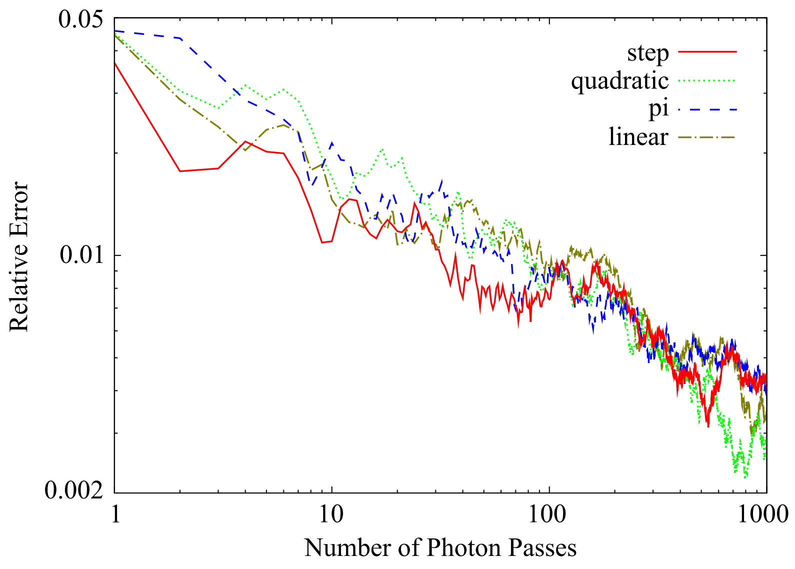


Figure 7.2: Error plot of 1D integration tests. We performed numerical integration of functions using our method. The functions are $y0(x) = 0.75 + 0.25\,\mathrm{sgn}(x - 0.5)$ (step), $y1(x) = x^2$ (quadratic), $y2(x) = 4\sqrt{1 - x^2}$ (pi), and $y3(x) = x$ (linear), where the range of integration is $x \in [0, 1]$ for all functions. We compute the relative error as $E(y) = \left|\frac{Y-y}{Y}\right|$, where Y is the analytical value of integration and y is a current estimate. The plot is using the average over 10 different random number sequences. The errors converge to zero as the number of photon passes increases.

7.4 Results

This section presents results based on our implementation of progressive photon mapping (PPM) and our stochastic progressive photon mapping (SPPM). As we noted initially, the implementation of SPPM is almost the same as PPM. One change is generating a set of new hit points by distributed ray tracing [17] after each photon pass. Another change is to assign shared statistics to each pixel, not each hit point. This is because the statistics are shared over the region S, and S is usually assigned to each pixel.

All of our test scenes have been rendered on a 2.4 GHz Intel Core 2 Q6600 using one core. The resolution of the images is 640×480, except for the bunny scene and the Cornell box scenes which use 512×512. All rendering comparisons are equal time, except for the progressive sequences in Figure 7.4. In all the scenes, each photon pass traced 500,000 photons and α is set to 0.7. Table 7.1 summarizes the statistics of our experiments.

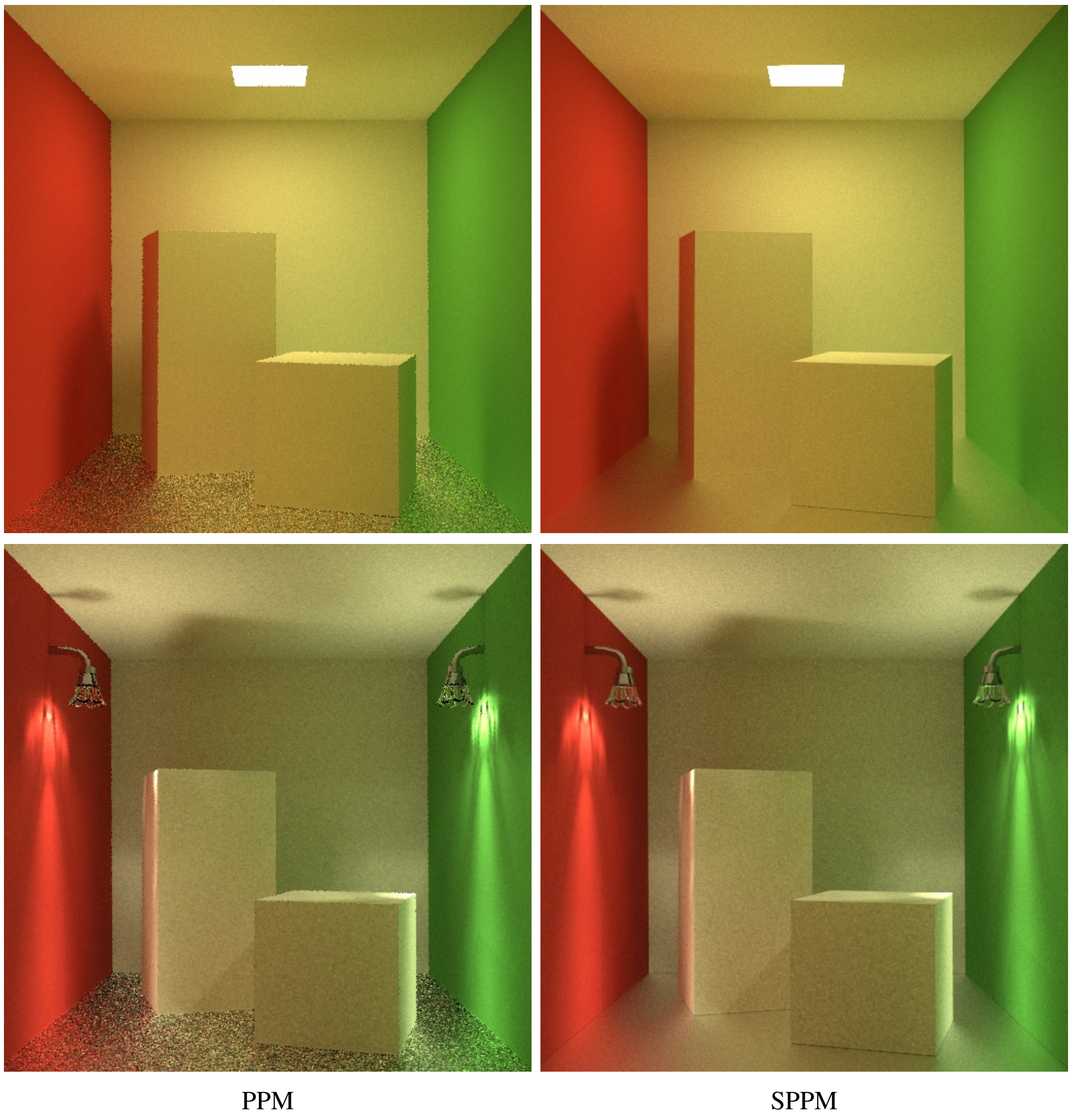


Figure 7.3: Cornell box with a glossy floor. The BRDF of the floor is the modified Phong model with a glossiness value of 25. The results of PPM are noisy because glossy reflections cause large variations of photon contributions to viewing directions. The bottom row replaced the area light source in the original Cornell box with two wall lights. Even though this change is simple from a user's perspective, it makes unbiased methods inefficient because of highly glossy reflections of caustics from the wall lights. SPPM is still as robust as PPM for this illumination setting, while rendering glossy reflections with less noise in the same rendering time.

Figure 7.2 shows a numerical validation of SPPM with a 1D function integration. We used 1D functions where the results of integrations over $[0,1]$ are known, and performed numerical integrations by estimating the average values over $S = [0,1]$ with SPPM. In each photon pass, photons (samples) are generated using rejection sampling so that they are distributed according to the function. We used 50,000 photons per pass and took the average error over 10 different runs in this experiment. As we can see, errors of SPPM are converging to zero as the number of photon tracing passes increases.

In order to highlight the improvement over PPM, we rendered the Cornell box with a glossy floor as in Figure 7.3. The floor uses the modified Phong mode [72] with a glossiness value of 25. Note that the results with PPM have significant noise on the floor. PPM is not efficient for glossy reflections because incoming directions of photons that contribute to the viewer direction are narrow. SPPM efficiency handles glossy reflections by tracing once bounce camera rays according to the BRDF and computing the hemispherical integration, similar to the final gathering step in standard photon mapping [62]. Note that a higher glossiness value makes PPM further inefficient because a smaller range of incoming directions contributes to the viewer direction. On the other hand, the performance of SPPM stays the same regardless of the glossiness value since extra rays are sampled according to the BRDF.

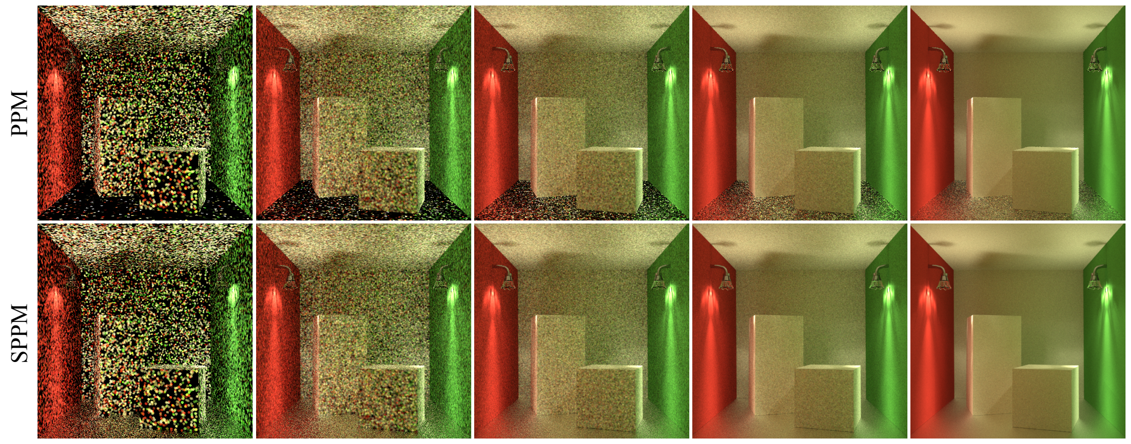
We could perform the same procedure with the original PPM by tracing rays and storing multiple hit points per pixel. However, this approach would require many hit points to obtain visually pleasing results, and the results are not converging to the correct solution with a fixed number of samples per pixel. SPPM does not require to store extra hit points, and the results converge to the correct solution just by computing more passes. Note that unbiased methods become inefficient to render this scene if the area light source is replaced by a light bulb. Such an example is shown in the bottom row of Figure 7.3, where the area light is replaced by two wall lights. This is because light bulbs illuminate the scene with caustics, causing SDS paths of the highly glossy reflections of caustics which are difficult to render with unbiased methods.

Figure 7.4 shows a progressive sequences of renderings for the Cornell box scene and the RMS errors compared to a reference result. For the reference result, we used the converged result generated using PPM with a large number of photon passes. We show images from both PPM and SPPM using the same number of photon passes, not equal time comparisons. Note that each pass of SPPM takes approximately 10 percent longer rendering time than PPM, because of additional distributed ray tracing per pass. However, SPPM can render visually pleasing results with a smaller number of photon passes in comparison to PPM, which makes SPPM worth the additional computational cost per pass. In addition, the RMS errors of SPPM are consistently lower than PPM.

Figure 7.5 shows the applications to anti-aliasing and motion blur. PPM used 1 sample per pixel to have equal memory consumption with SPPM. In order to correctly render motion blur with PPM, we sampled the time when hit points are generated (16 samples per pixel are used). The sampled time value is assigned to each hit point so that each photon contributes to hit points in the same time sample. Using the same rendering time, SPPM can include anti-aliasing and motion blur with a constant amount of memory consumption independent of the number of samples per pixel. PPM needs an increasing amount of memory in order to increase the number of samples per pixel.

Figure 7.6 shows a rendering with depth-of-field. PPM used 16 hit points (i.e., 16 radiance samples) per pixel, which consumed approximately 1GB of memory in our implementation. Note that the scene is dominated by SDS paths because all the illumination is due to the desk lamp with a light bulb outside the view, and we observe the scene through a lens to achieve depth-of-field. Although PPM can handle SDS paths, the result is noisy because the number of radiance samples per pixel is insufficient to remove noise due to depth-of-field. SPPM renders the same scene with less noise using 16 times less memory consumption. We also provide the comparison with bidirectional path tracing (BDPT) with multiple importance sampling [118] in this scene. The comparison confirms that this scene cannot be rendered efficiently by BDPT due to SDS paths.

Finally, we show a rendering of a scene with depth-of-field and glossy reflections in Figure 7.7. PPM used 16 hit points per pixel as before. We used the modified Phong model with the glossiness value of 100. The entire scene is illuminated by the flashlight in front, which causes highly glossy reflections of caustics on the metallic parts of plier and bolts. The result with PPM suffers from both noise due to both glossy reflections and depth-of-field. PPM consumed approximately 16 times more memory than SPPM because PPM uses 16 samples per pixel. In the same rendering time with smaller memory consumption, SPPM renders this scene with less noise.



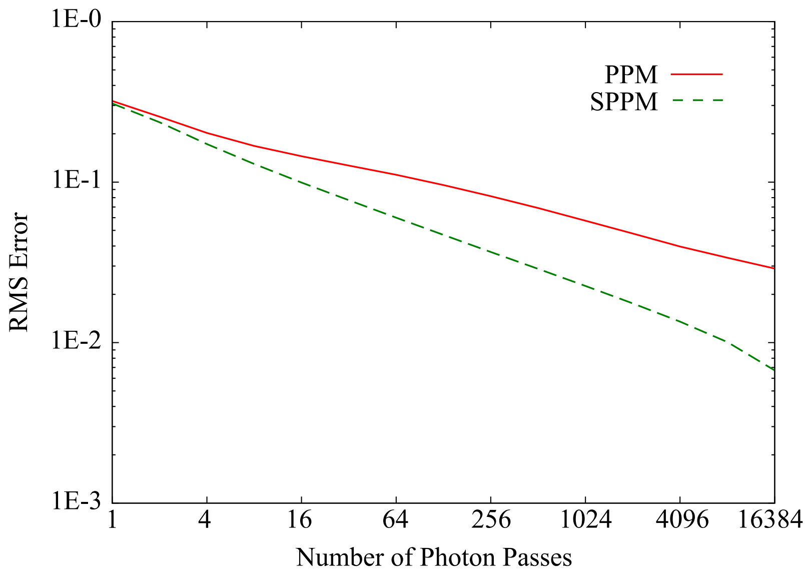


Figure 7.4: Progressive sequences of rendering of Cornell box and the RMS errors. The number of photon passes of the images is 1, 8, 64, 512, and 4096 correspondingly. The graph shows the RMS errors from the converged result with PPM. The result with SPPM converges visually pleasing results with a smaller number of photon passes, and the RMS errors are consistently lower than the result with PPM. The rendering time of SPPM is approximately 10 percent longer than that of PPM for the same number of photon passes.

7.5 Conclusion

This chapter presented an extension of progressive density estimation, called stochastic progressive density estimation, that makes it possible to compute the correct average density value over a region. We modify progressive density estimation by adding a stochastic process to each iteration that generates new locations of measurement points.

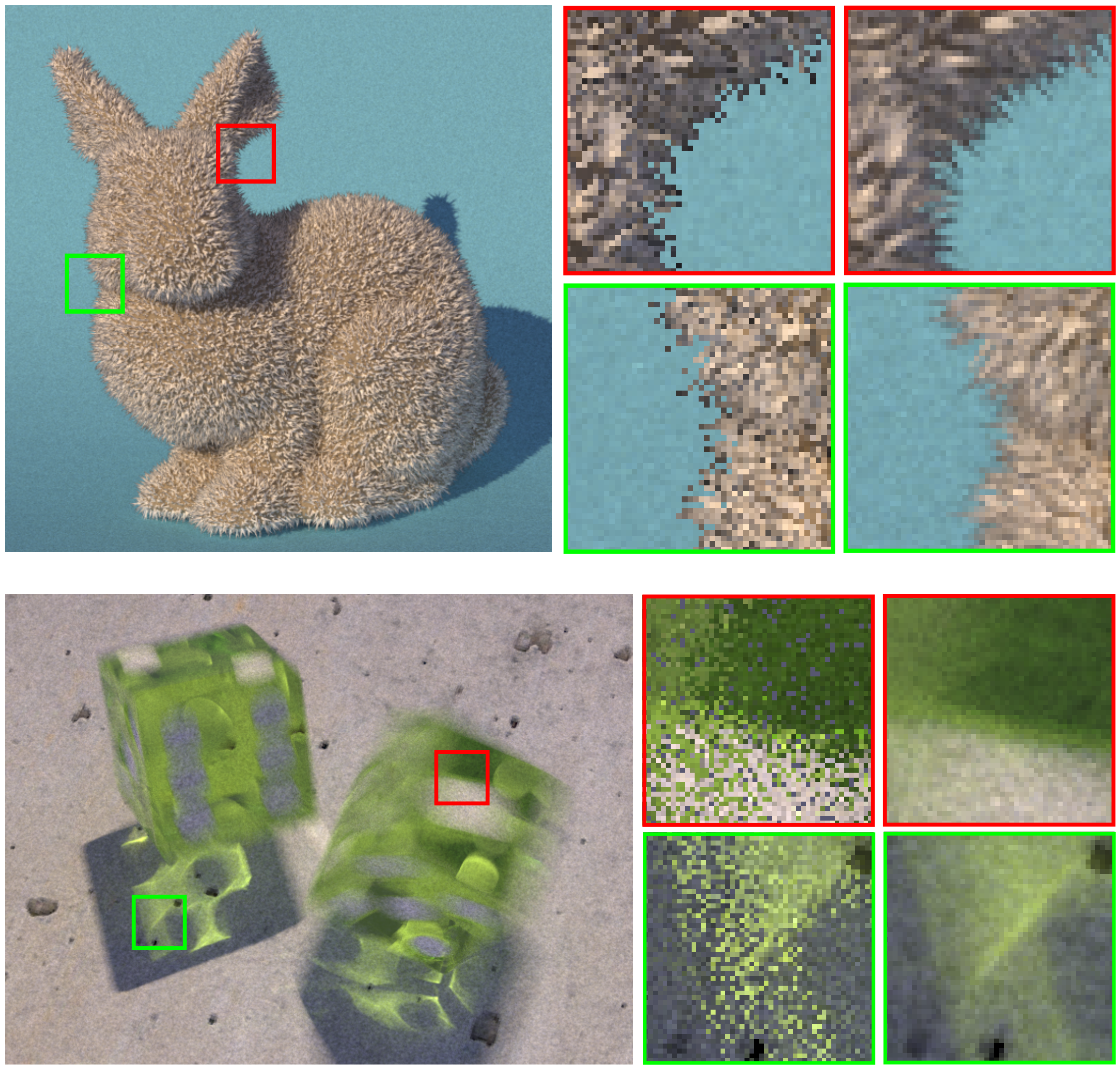


Figure 7.5: Furry bunny illuminated by the skylight and moving transparent dices with motion blur. The dices are illuminated by the sunlight, and blur of caustics and shadows is due to motion blur of the dices. The close-ups compare the results with PPM and SPPM (right columns), and we show the entire rendered images with SPPM. The comparisons are equal time and equal memory consumption (i.e., PPM used 1 sample per pixel).

The key insight is to use shared statistics over the region for taking the average. Based on this extension, we presented an improvement of progressive photon mapping as stochastic progressive photon mapping. The results show that stochastic progressive photon mapping is robust in scenes with complex illumination settings including distributed ray tracing effects, such as depth-of-field, motion blur, and glossy reflections/refractions.

7.6 Appendix

7.6.1 Convergence of Ratio of Radius

This section shows that the ratio $\lim_{i\to\infty} R_i(S)^2/R_i(\vec{x}_0)^2$ is convergent and non-zero in the following. We first expand the ratio using its definition:

$$\lim_{i\to\infty}\frac{R_i(S)^2}{R_i(\vec{x}_0)^2}=\prod_{j=0}^{\infty}\frac{(N_j(S)+\alpha M_j(\vec{x}_j))(N_j(\vec{x}_0)+M_j(\vec{x}_0))}{(N_j(S)+M_j(\vec{x}_j))(N_j(\vec{x}_0)+\alpha M_j(\vec{x}_0))} \tag{7.16}$$

In the following derivations, we use the notations $N_S = N_j(S)$, $N_0 = N_j(\vec{x}_0)$, $M_j = M_j(\vec{x}_j)$, and $M_0 = M_j(\vec{x}_0)$ for readability. Using these notations, we further expand the equation and obtain the upper bound as:

$$\begin{aligned}
&=\prod_{j=0}^{\infty}\frac{(N_S+\alpha M_j)(N_0+M_0)}{(N_S+M_j)(N_0+\alpha M_0)}\\
&=\prod_{j=0}^{\infty}\left(1+\frac{(1-\alpha)(N_SM_0-N_0M_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}\right)\\
&=\prod_{j=0}^{\infty}\left(1+\frac{(1-\alpha)N_0M_0(\frac{N_S}{N_0}-P_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}\right)\\
&=\prod_{j=0}^{\infty}\left(1+Q_j\right).
\end{aligned} \tag{7.17}$$

where $P_j = \frac{M_j}{M_0}$ and we defined Q_j as:

$$Q_j=\frac{(1-\alpha)N_0M_0(\frac{N_S}{N_0}-P_j)}{N_SN_0+\alpha N_SM_0+M_jN_0+\alpha M_jM_0}. \tag{7.18}$$

Taking the logarithm of both sides, this infinite product converges if and only if the infinite sum

$$Q=\sum_{j=0}^{\infty}Q_j \tag{7.19}$$

converges as each term of the infinite product is always positive. Note that Q_j is a random variable. Assuming non-adaptive photon tracing, we can consider $M_0 = M_j(\vec{x}_0) = 1$ for a large enough j. We thus obtain

$$\lim_{j\to\infty}\frac{N_S}{N_0}=\lim_{j\to\infty}\frac{\alpha\sum_j M_j}{\alpha\sum_j M_0}=\lim_{j\to\infty}\frac{\sum_j M_0P_j}{\sum_j M_0}=\mathrm{E}[P_j], \tag{7.20}$$

so the numerator $\frac{N_S}{N_0} - P_j$ is symmetric around zero, therefore, $\mathrm{E}[Q_j] = 0$. Furthermore, we obtain the following:

$$\sum_{j=0}^{\infty} \mathrm{Var}[Q_j^2] = \sum_{j=0}^{\infty} \mathrm{E}[Q_j^2] < \sum_{j=0}^{\infty} \frac{C_1}{(j+1)^2} < \infty, \tag{7.21}$$

where C_1 is a constant because the numerator of Q_j is bounded. This derivation also uses the fact that the lower bound of N_S is $\alpha(j+1)$ (i.e., only one photon is captured), and the upper bound of M_0 is $Ne(1)$ (i.e., all the emitted photons are captured).

Using $\mathrm{E}[Q_j] = 0$ and $\sum_{j=0}^{\infty} \mathrm{Var}[Q_j^2] < \infty$ with Kolmogorov's one series theorem, the infinite sum Q almost surely converges. Therefore, the infinite product also almost surely converges. We then show that the ratio is non-zero by showing the reciprocal of the ratio is bounded. Namely, we show that $\lim_{i\to\infty} R_i(\vec{x}_0)^2/R_i(S)^2$ is convergent. Similar to before, we expand the ratio as follows:

$$\begin{aligned}
\lim_{i\to\infty} \frac{R_i(\vec{x}_0)^2}{R_i(S)^2} &= \prod_{j=0}^{\infty} \frac{(N_S + M_j)(N_0 + \alpha M_0)}{(N_S + \alpha M_j)(N_0 + M_0)} \\
&= \prod_{j=0}^{\infty} \left(1 + \frac{(1-\alpha)(N_0 M_j - N_S M_0)}{N_S N_0 + N_S M_0 + \alpha M_j N_0 + \alpha M_j M_0}\right) \\
&= \prod_{j=0}^{\infty} \left(1 + Q'_j\right).
\end{aligned} \tag{7.22}$$

The rest of the derivation is the same as before, but with the opposite sign of random variables and use Q'_j instead of Q_j.

7.6.2 Bound of $|E_i|$

This section shows that $|E_i|$ is bounded by a constant. In the following derivation, we first consider $n|E_i|$ for the purpose of discussion. We expand $n|E_i|$ using the definition of $\tau_i(S, \vec{\omega})$ and $\tau_{R(S),i}(\vec{x}_k, \vec{\omega})$:

$$\begin{aligned}
n|E_i| &= \left| \sum_{k=1}^{n} \left(\tau_{R(S),i}(\vec{x}_k, \vec{\omega}) - \tau_i(S, \vec{\omega})\right) \right| \\
&= \left| \sum_{k=1}^{n} \left(\sum_{j=0}^{i} \Phi_j(\vec{x}_k) - \sum_{j=0}^{i} \Phi_j(\vec{x}_j) \right) \right|.
\end{aligned} \tag{7.23}$$

Note that we can use the same $\Phi_j(\vec{x})$ to expand both $\tau_{R(S),i}(\vec{x}_k,\vec{\omega})$ and $\tau_i(S,\vec{\omega})$ because both accumulated flux values use the same search radius $R(S)$ as a result of Section 4.1. Since photon flux $\Phi_j(\vec{x})$ is proportional to the radiance value $L(\vec{x})$ (i.e., $\Phi_j(\vec{x})$ divided by the area is radiance) and radius is the same everywhere in S (Equation 7.8), we can further expand the above equation into:

$$\begin{aligned} &= \frac{1}{L(\vec{x}_0)}\left|\sum_{k=1}^{n}\left(\sum_{j=0}^{i} L(\vec{x}_k)\Phi_j(x_0) - \sum_{j=0}^{i} L(\vec{x}_j)\Phi_j(x_0)\right)\right| \\ &= \frac{1}{L(\vec{x}_0)}\left|\sum_{k=1}^{n}\sum_{j=0}^{i}\Phi_j(\vec{x}_0)(L(\vec{x}_k)-L(\vec{x}_j))\right| \\ &= \frac{1}{L(\vec{x}_0)}\left|\sum_{j=0}^{i}\Phi_j(\vec{x}_0)\sum_{k=1}^{n}\big(L(\vec{x}_k)-L(\vec{x}_j)\big)\right|. \end{aligned} \tag{7.24}$$

$\Phi_j(\vec{x}_k) = \frac{L(\vec{x}_k)\Phi_j(x_0)}{L(\vec{x}_0)}$ is only approximately true because $\Phi_j(\vec{x}_k)$ is estimated using a finite number of photons per pass. However, this error does not diverge as $j \to \infty$ (i.e., we at least have one photon per pass), so we can ignore this error in this derivation. Since photon flux $\Phi_j(\vec{x})$ is monotonically decreasing as $j \to \infty$ and $i = n$ in our formulation, we obtain the upper bound as:

$$\begin{aligned} n|E_i| &\leq \frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\sum_{j=0}^{i}\left|\sum_{k=1}^{n}\big(L(\vec{x}_k)-L(\vec{x}_j)\big)\right| \\ &= \frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\sum_{j=0}^{n}\left|\sum_{k=1}^{n}L(\vec{x}_k)-nL(\vec{x}_j)\right| \end{aligned} \tag{7.25}$$

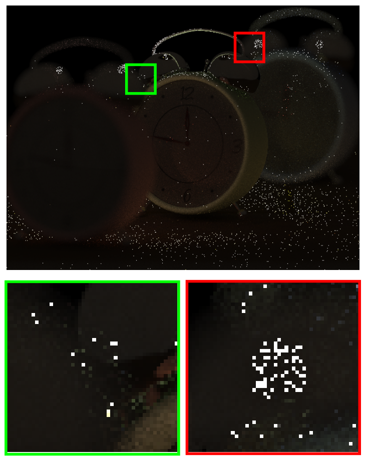
Assuming the variation of $L(\vec{x})$ is bounded (i.e., $0 \leq L(\vec{x}) < \infty$), the double summation above can be written as $C_2 n$, where C_2 is a constant. Dividing the both side by n, we obtain:

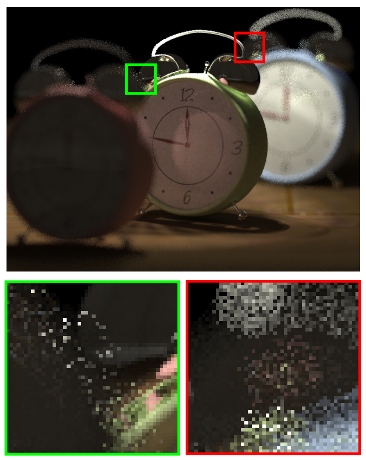
$$|E_i| \leq \frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}\frac{1}{n}C_2 n = \frac{\Phi_0(\vec{x}_0)}{L(\vec{x}_0)}C_2, \tag{7.26}$$

Therefore, $|E_i|$ is bounded by a constant.

7.7 Acknowledgements

The material in this chapter is, in part, a reproduction of the material published in Toshiya Hachisuka and Henrik Wann Jensen. “Stochastic Progressive Photon Mapping”, ACM Transactions on Graphics, 28(5), 141:1–141:8, 2009 [42]. The dissertation author was the primary investigator and author of this paper. We would like to thank Youichi Kimura (Studio Azurite) for providing the flashlight model. The clock, bolts, and plier models are from ShareCG.com. We would also like to thank Antoine Bouthors and Leonhard Grünschloß for comments on the original material. The dissertation author was supported by the 2008 ATI Fellowship Program.


Bidirectional Path Tracing


Progressive Photon Mapping

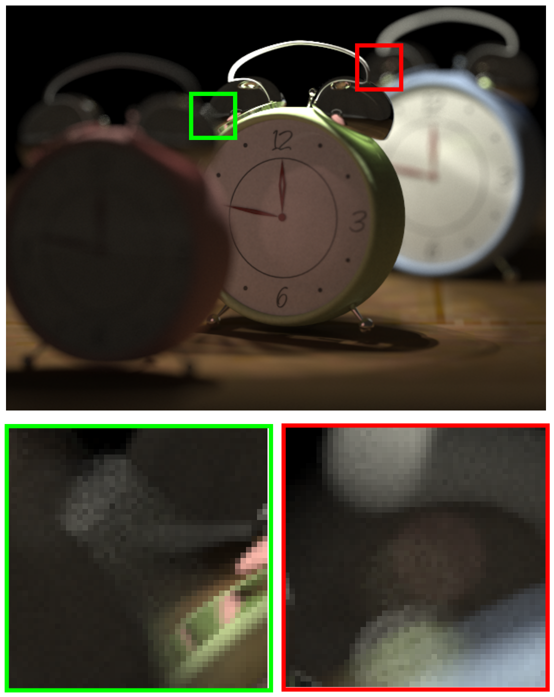
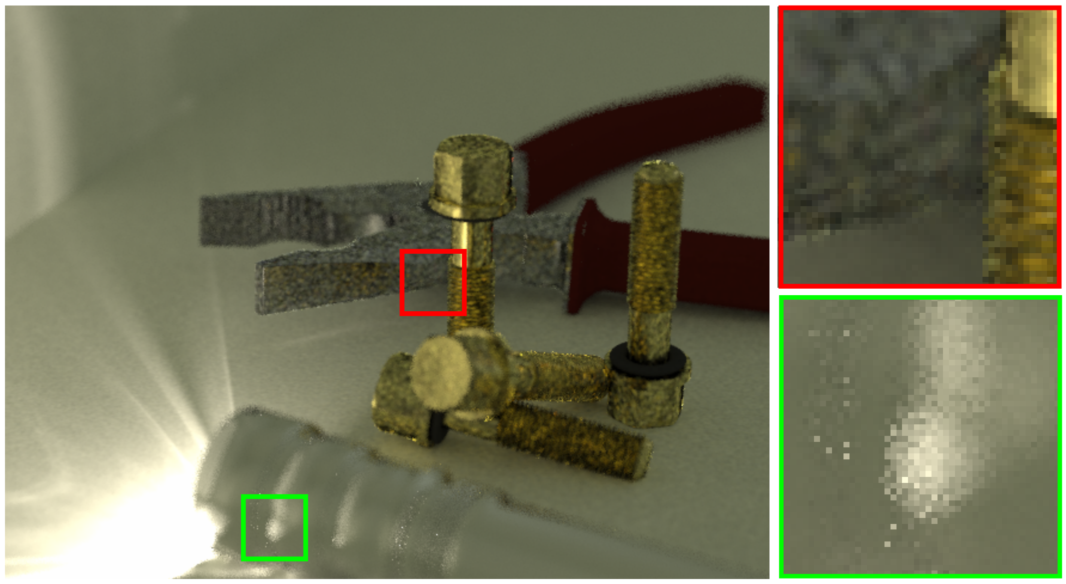

Stochastic progressive photon mapping

Figure 7.6: Alarm clocks illuminated by a desk lamp. The desk lamp with a light bulb is outside the view and illuminates the clocks. The scene is rendered using a thin lens model for depth-of-field. The combination of the lens for depth-of-field and caustics from the desk light makes the entire scene dominated by SDS paths, which is difficult to render with unbiased methods such as BDPT. PPM can handle such an illumination setting, but the close-ups show rendering of depth-of-field causes visually noticeable noise due to the fixed number of samples per pixels. SPPM robustly handles illumination by the desk lamp as well as depth-of-field in the same rendering time with less memory consumption.


Progressive photon mapping

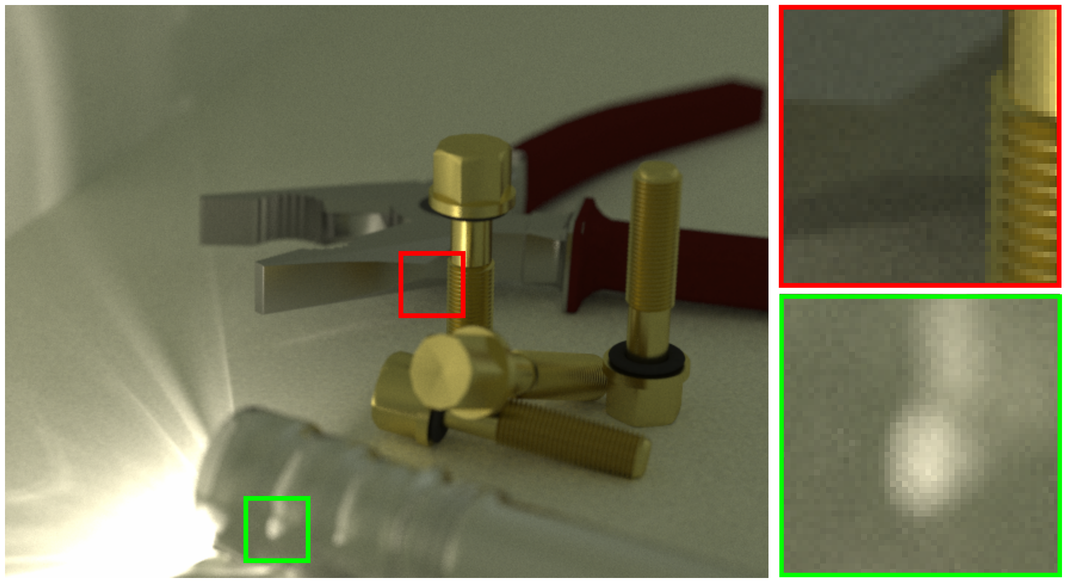

Stochastic progressive photon mapping

Figure 7.7: Tools with a flashlight. The scene is illuminated by caustics from the flashlight, which cause SDS paths on the flashlight and highly glossy reflections of caustics on the bolts and plier. The flashlight and the plier are out of focus. Using the same rendering time, our method (bottom) robustly renders the combination of the complex illumination setting and the distributed ray tracing effects where progressive photon mapping is inefficient (top).

Chapter 8

Adaptive Photon Tracing using Path Visibility

In order to simulate light transport using density estimation, we need to generate samples according to the rendering equation. This process is typically called photon tracing as we trace a path of photon that is scattering around a scene. This chapter introduces an efficient and robust adaptive sampling algorithm for photon tracing. The algorithm specifically targets difficult lighting conditions where only a fraction of the generated photons contribute to the final image. The key contribution is the use of photon path visibility as the importance function as well as two recent developments in Markov chain Monte Carlo methods: adaptive Markov chain sampling and replica exchange. The algorithm adaptively mutates samples by self-learning the importance function so no parameter tuning is required. It also incorporates a uniform sampling approach that prevents a Markov chain from getting stuck in local regions of the sampling space.

8.1 Related Work

Photon tracing has been used for solving the light transport problem in various algorithms. Backward ray tracing [8] is the first application of photon tracing, where illumination is recorded on discretized grids on surfaces. Light tracing [23] accumulates the power of each photon at directly visible diffuse surface intersection points onto the image. Photon mapping [61] avoids discretization of illumination data by using kNN density estimation. Bidirectional path tracing [71, 118] combines photon tracing and eye ray tracing, which improves the robustness in the presence of light paths such as caustics. Our method uses progressive photon mapping as its rendering algorithm since it can robustly render scenes that are difficult to render with other methods (e.g., specular-diffuse-specular paths from a directional light source).

Photon tracing methods can become inefficient when only a small part of the illuminated scene is visible. A related problem was addressed by Veach and Guibas [119], who introduced Metropolis light transport as an application of Markov chain Monte Carlo methods to rendering. The underlying assumption is that, given a sampled light path that we already know contributes to the image, similar light paths will be likely to contribute to the image as well. In order to exploit this observation, Metropolis light transport generates a new path by slightly perturbing the previous path. This strategy has been demonstrated to work well in difficult settings, such as illumination coming through a small gap. Markov chain Monte Carlo methods have been successfully used in other forms as well, such as Multiple try Markov chain Monte Carlo methods [103] and energy redistribution path tracing [15].

Most related to our work is the method by Fan et al. [25], who applied Metropolis light transport to photon tracing. In order to obtain the full information of a photon path including whether it contributes to the image, they proposed to sample a complete path from the path space that connects a light source and the eye. Common to existing MCMC rendering methods and their work, however, is that they all use sampling on the exact path space. This unfortunately becomes inefficient in the presence of some light paths, such as specular-diffuse-specular paths from small light sources. This is because these paths will have nearly zero volume (or zero in the case of point light sources and directional light sources) in the path space, which cannot be efficiently sampled by random sampling or random mutations [117]. For example, specular reflections of caustics from a point light source form delta functions in the path space, where any random sampling method including Markov chain Monte Carlo methods with random mutation have zero probability of sampling the path. Even paths from small light sources can result in path sampling being arbitrarily inefficient depending on the relative size of the light sources. These lighting scenarios are rather common in the real world (e.g., illumination coming from the filament in a light bulb). Our method avoids this issue by sampling a blurred path space using progressive photon mapping, where blurring vanishes as we add more samples.

Adaptive Markov chain Monte Carlo methods [40] are modified Markov chain Monte Carlo methods that adaptively adjust mutations by self-learning the target distribution. It has been known that the mutation parameters, thereby mutation kernels affect the performance of Markov chain Monte Carlo methods significantly. Searching for the best set of parameters have typically been done by the user. However, this is a time consuming task and not always straightforward. This is also true in applications of Markov chain Monte Carlo methods in rendering, where users need to fine tune parameters of the given mutation strategies to get the best image quality. Adaptive Markov chain Monte Carlo methods automate this process by using information obtained from past samples. We introduce adaptive Markov chain Monte Carlo methods to rendering, and as far as we know, our method is the first application of adaptive Markov chain Monte Carlo methods in graphics.

The replica exchange Monte Carlo method uses multiple target distributions or an extended space of the target distribution with auxiliary parameters in order to introduce inter-distribution mutations [111]. This alleviates the problem of a Markov chain being trapped within a single mode in a multi-modal distribution by taking a “detour” of the Markov chain through different distributions. Kitaoka et al. [69] applied this algorithm to Metropolis light transport, and reported an improvement over existing Markov chain Monte Carlo rendering methods. We provide a formulation of replica exchange Monte Carlo method in the context of our visibility importance function. The key difference is that our formulation results in a strikingly simple algorithm that is independent from scene settings.

8.2 Method

8.2.1 Overview

The idea is to define a visibility function of photon paths and to perform importance sampling on this visibility function. We define the space of this function as a hypercube similar to the one that was proposed by Kelemen et al [67]. Note that a point in this space corresponds to a set of random numbers. We then employ local importance sampling for choosing light sources and sampling BRDFs and Russian roulette, in order to generate a photon path from given random numbers. In order to efficiently sample this function, we propose a combination of adaptive Markov chain sampling and replica exchange as we will describe in the next sections.

8.2.2 Sampling Space and Visibility Function

We first define our sampling space and the importance function. Given a photon path, $\vec{u}$, in the hypercube, we define a *photon path visibility function*, $V(\vec{u})$, where $V(\vec{u}) = 1$ if any photon due to this photon path contributes to the image and $V(\vec{u}) = 0$ otherwise. The importance function is simply the normalized version of this visibility function $V(\vec{u})$, which is $F(\vec{u}) = \frac{V(\vec{u})}{V_c}$ where $V_c = \int V(\vec{u})\mathrm{d}\vec{u}$. Figure 8.1 illustrates the definition of our sampling space and visibility function. The use of this function has the additional advantage that there is no local peak in the function, which is prone to high autocorrelation of samples in Markov chain Monte Carlo methods (e.g., a chain gets stuck in a very bright path). We can also easily evaluate $V(\vec{u})$ by checking if a photon path splats any photon into any of measurement points in the photon splatting implementation of progressive photon mapping that we describe below.

8.2.3 Photon Splatting Implementation

Progressive photon mapping has three user-defined parameters, α, the number of emitted photons at each iteration N^e, and the initial radius $R_0(x)$. A result computed using progressive photon mapping always converges to the correct solution regardless of these parameters. In particular, we found that the behavior of the algorithm is not sensitive to

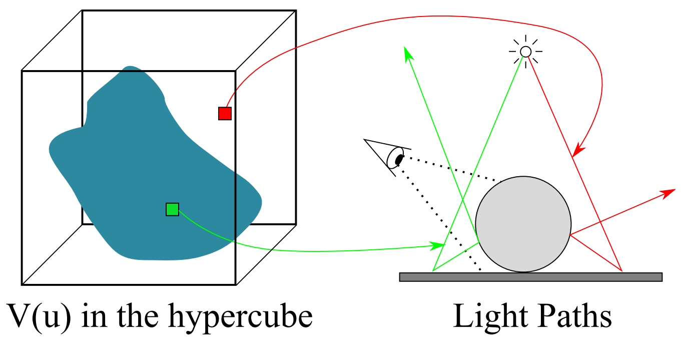


Figure 8.1: Sampling space of our method. We define a function $V(x)$ in the hypercube of random numbers. The function returns 1 if a corresponding photon path contributes to the image (the green point in the shaded region) and 0 otherwise (the red point outside the shaded region).

the number of photons per iteration. Even using $N^e = 1$ still results in almost identical convergence behavior. This leads to an important variation of the implementations of progressive photon mapping, which uses photon splatting.

In this variation of progressive photon mapping, we construct an acceleration data structure of measurement points, not a photon map. In the succeeding photon passes, instead of storing photons as a photon map, we perform a range query of measurement points at each photon's position. In other words, this algorithm is *splatting* photons into the measurement points, instead of *gathering* photons at each measurement point. We then apply the radius reduction and the flux correction normally to all of affected measurement points. The radiance estimation at each measurement point can be done as usual. Resampling of measurement points in stochastic progressive photon mapping is done after tracing a user-specified number of photons, which controls the frequency of eye ray tracing. We use this splatting implementation in order to immediately utilize the visibility information of the current photon path to next photon tracing. Note that a single photon path can potentially contribute to multiple measurement points.

8.2.4 Algorithm

The rendering algorithm we combine with our photon tracing method is the splatting variation of stochastic progressive photon mapping that we described in the previous section. The pseudocode for the algorithm is shown in Figure 8.2.

```
$\text{MutationSize} \leftarrow 1, \text{Accepted} \leftarrow 1, \text{Mutated} \leftarrow 0, \text{UniformCount} \leftarrow 1$
repeat
  $\text{CurrentPath} \leftarrow \textsc{Uniform}()$
until $\textsc{IsVisible}(\text{CurrentPath})$
for $i \leftarrow 1$ to NumTotalPhotons
  do
    $\text{UniformPath} \leftarrow \textsc{Uniform}()$
    if $\textsc{IsVisible}(\text{UniformPath})$
      then
        $\text{CurrentPath} \leftarrow \text{UniformPath}$
        $\text{UniformCount} \leftarrow \text{UniformCount} + 1$
      else
        $\text{CandidatePath} \leftarrow \textsc{Mutate}(\text{CurrentPath}, \text{MutationSize})$
        $\text{Mutated} \leftarrow \text{Mutated} + 1$
        if $\textsc{IsVisible}(\text{CandidatePath})$
          then
            $\text{CurrentPath} \leftarrow \text{CandidatePath}$
            $\text{Accepted} \leftarrow \text{Accepted} + 1$
        $\text{R} \leftarrow \text{Accepted}/\text{Mutated}$
        $\text{MutationSize} \leftarrow \text{MutationSize} + (\text{R} - 0.234)\,/\text{Mutated}$
    $\textsc{Splat}(\text{CurrentPath})$
    $\textsc{Display}(\text{UniformCount}/i)$
procedure $\textsc{Mutate}(\text{CurrentPath}, \text{MutationSize})$
  for $k \leftarrow 1$ to DIMENSION
    do
      $d \leftarrow \textsc{pow}(\textsc{rand}(), 1/\text{MutationSize} + 1)$
      $t \leftarrow \text{CurrentPath.u}[k] + \text{sgn}(2 * \textsc{rand}() - 1) * \text{d}$
      $\text{CurrentPath.u}[k] \leftarrow t - \textsc{floor}(t)$
```

Figure 8.2: Our photon tracing algorithm. UNIFORM() samples a hypercube using uniform random numbers, and MUTATE() returns a mutated path with a given mutation strategy parameter (MutationSize). SPLAT() finds nearby measurement points of each photon and accumulates photon statistics. ISVISIBLE() returns true if the given photon path splats any photon into any of measurement points. DISPLAY() takes its argument as a scaling factor, and computes pixel values using current photon statistics. sgn(x) returns 1 if $x > 0$ and -1 otherwise. DIMENSION is the number of dimensions of the hypercube. CurrentPath.u$[k]$ stores a k-th random number of the photon path. $t - \textsc{floor}(\text{t})$ ensures that the mutated value is always within $(0, 1)$.

8.3 Adaptive Markov chain Monte Carlo

8.3.1 Overview

One notable difficulty in Markov chain Monte Carlo methods is that the optimal mutation strategy is problem dependent. For example, if we render an object on a plane with a point light source, depending on the relative size of the plane and the object, the range of photon paths that intersect with the object in the hypercube dramatically changes. In general, no single preset of mutation strategies will work well in all scene settings. We could have as many mutation strategies as possible to hope that at least one of them are effective, but this approach wastes computation on other ineffective mutations. Adaptive Markov chain Monte Carlo methods provide a way to automatically adjust mutation strategies during the computation by learning the target distribution as we sample. Since adaptive Markov chain Monte Carlo methods in general cover many different algorithms, we only provide an overview of the method that we use, which is *controlled Markov chain Monte Carlo method* [5]. For a more comprehensive overview of other methods, please refer to a survey such as the one by Andrieu and Thoms [6].

The idea of a controlled Markov chain Monte Carlo method is to adjust the mutation parameters based on the previous samples. Given a vector of the initial parameter values, $\vec{\theta}_1$, a controlled Markov chain Monte Carlo method updates the current parameters $\vec{\theta}_i$ as:

$$\vec{\theta}_{i+1} = \vec{\theta}_i + H(i, \vec{\theta}_i, \vec{u}_i, \ldots, \vec{u}_1), \tag{8.1}$$

where $\vec{u}_i, \ldots, \vec{u}_1$ are all samples up to the ith iteration and H is a function that computes the changes of the parameters according to the history of samples and the last parameters. There are many possible choices of the function H, however, one important condition that H needs to satisfy in order to maintain the stationary distribution which is known as *Diminishing adaptation principle* [6];

$$\lim_{i \to \infty} H(i, \vec{\theta}_i, \vec{u}_i, \ldots, \vec{u}_1) = 0. \tag{8.2}$$

Although there are many ways to adapt the parameters, one approach that is used in existing adaptive Markov chain Monte Carlo methods is changing the parameters such that the acceptance ratio of the Markov chains reaches the optimal acceptance rate ("acceptance ratio" is the fraction of accepted mutations over all mutations). In fact, for separable distributions, the optimal asymptotic acceptance ratio has been derived $A^* = 23.4\%$ [98]. Adapting toward the optimal acceptance ratio has another benefit that computation being inexpensive. Equation 8.1 is thus simplified as:

$$\vec{\theta}_{i+1} = \vec{\theta}_i + H(i, A^*, A_i), \tag{8.3}$$

where A_i is the current acceptance ratio of samples up to i.

While it is true that the target distribution will not be separable in many cases for the light transport problem, previous work confirmed that using 23.4% works well in practice with nonseparable distributions [99]. Furthermore, the general principle that the acceptance ratio should not be too high or too low will apply in any case, so any target acceptance ratio that is not too close to 0% or 100% will work. We therefore use $A^* = 23.4\%$ in all the examples, which is indeed optimal if the target distribution happens to be separable. The key is that the same target acceptance ratio will work well for a wide range of target distributions and the user does not need to tweak the target acceptance ratio scene by scene as we will demonstrate.

8.3.2 Our Formulation

We use a simple form of a controlled Markov chain Monte Carlo method, which adjusts a single mutation parameter in a power function. A mutation of each coordinate of a given point is done by adding

$$\Delta u = \mathrm{sgn}(2\xi_0 - 1)\xi_1^{\frac{1}{\theta_i}+1} \tag{8.4}$$

to each coordinate while keeping it within $(0,1)$. θ_i is the adaptive mutation size at the ith Markov chain, $\mathrm{sgn}(x)$ is a function that returns the sign of x, and ξ_0 and ξ_1 are uniform random numbers within $(0,1)$. The mutation size is shared between all samples regardless of the current state. Note, that $\theta_i = \infty$ corresponds to uniform random sampling which generates as large mutation as possible, and $\theta_i = 0$ corresponds to staying at the same position all the time.

The acceptance probability of a mutated path is easily computed since the mutation is symmetric and $V(\vec{u}) = 1$. Specifically, given a set of mutations as a vector

$\Delta\vec{u} = (\Delta u, \ldots, \Delta u)$ the acceptance probability is

$$a(\vec{u}+\Delta\vec{u} \leftarrow \vec{u}) = \frac{F(\vec{u}+\Delta\vec{u})}{F(\vec{u})} = \frac{V(\vec{u}+\Delta\vec{u})}{V(\vec{u})} = V(\vec{u}+\Delta\vec{u}), \tag{8.5}$$

which means a mutation is accepted if the mutated path is visible. In contrast to existing Markov chain Monte Carlo rendering methods, there is no need for generating another random number to decide whether we accept a mutation or not.

We compute the acceptance ratio, A_i, by counting the number of accepted mutations. We then update θ_i as follows:

$$\theta_{i+1} = \theta_i + \gamma_i(A_i - A^*). \tag{8.6}$$

where $\gamma_i = 1/i$ and $\theta_1 = 1$. The intuition behind this equation is that an acceptance ratio that is too large ($A_i - A^* > 0$) would indicate that the mutation size is too small thus we increase the mutation size, and likewise if the acceptance ratio is too small ($A_i - A^* < 0$) it indicates that the mutation size is too large so we decrease the mutation size. Note, that the difference $A_i - A^*$ can never converge to zero in some scenes – e.g. a scene where all paths are visible ($A_i = 100\%$). However, using $\gamma_i = 1/i$ ensures that we always satisfy Equation 8.2. Another condition for convergence, *Bounded Convergence*, requires the product of the state space and the space of mutations to be finite, which is satisfied in practice with our sampling space and bounding θ_i to a large finite range. Since A^* is "embedded" into the algorithm and users will not touch it, and the algorithm is parameter-free.

8.4 Replica Exchange Monte Carlo

8.4.1 Overview

The replica exchange Monte Carlo method is an extended ensemble Monte Carlo method where we sample Markov chains from multiple distributions simultaneously (refer to Iba [56] for an overview of this class of algorithms). The basic idea is to facilitate exploration of the sampling space by bridging multiple distant peaks using a smoother distribution. For example, if we use a standard Markov chain Monte Carlo method to sample from a distribution with two peaks separated by zeros, the Markov chain tends to get trapped within one peak for a long time. The replica exchange Monte Carlo method can avoid this problem by using an extra Markov chain from a uniform distribution. Even if a Markov chain gets stuck in one peak, it can be exchanged with another Markov chain from the uniform distribution without breaking the resulting sample distribution. Figure 8.3 illustrates this idea. We first describe a formal definition of replica exchange Monte Carlo method in this section and explain our formulation in the next section.

Given a set of multiple target distributions, $F_1(\vec{u}), \ldots, F_Q(\vec{u})$, we define a set of independently generated samples from each distribution as $\vec{U} = \{\vec{u}_1, \ldots, \vec{u}_Q\}$. Under these definitions, $\vec{U}$ can be considered a single sample from the following product distribution:

$$\tilde{F}(\vec{U}) = \prod_{k=1}^{Q} F_k(\vec{u}_k), \tag{8.7}$$

where $\vec{u}_k$ is a sample (or a state of the Markov chain) in the target distribution F_k.

The key idea of the replica exchange Monte Carlo method is to perform an inter-distribution exchange such that the above product distribution of samples remains unchanged. This can be achieved by exchanging states of two chains, $\vec{u}_i$ and $\vec{u}_j$, with the probability

$$r\left(\vec{u}_i, \vec{u}_j\right) = \min\left(1, \frac{F_i(\vec{u}_j)F_j(\vec{u}_i)}{F_i(\vec{u}_i)F_j(\vec{u}_j)}\right). \tag{8.8}$$

As a result, each sequence of Markov chain $\vec{u}_k$ still distributes according to $F_k(\vec{u})$, but possibly with a better exploration of the sampling space due to inter-distribution exchanges.

One application of the replica exchange Monte Carlo method to the light transport problem has recently been done by Kitaoka et al. [69] in the context of improving Metropolis light transport, where they defined target distributions of separated light paths in the heuristic order of difficulty. Unfortunately, this separation is highly scene dependent and its implementation becomes relatively complex compared to the regular Metropolis light transport algorithm. Our algorithm is far simpler than their work in terms of implementation and independent from scene configurations. Since our method is based on progressive photon mapping, our method is robust in the presence of difficult light paths (such as specular-diffuse-specular paths) without relying on a complex separation of light paths.

8.4.2 Our Formulation

Although the replica exchange Monte Carlo method can use multiple different distributions, we only use two distributions: one is the target distribution $F(\vec{u})$ as defined by the visibility function, and a uniform distribution $I(\vec{u}) = 1$. Note, that even though $F(\vec{u})$ only returns either zero or $1/V_c$ without any local peak, samples in standard Markov chain Monte Carlo methods can still be trapped within one region in the hypercube for a large number of iterations (e.g., only sampling light from one window out of two windows in a room).

We consider two Markov chains $\vec{u}_F$ and $\vec{u}_I$ from $F(\vec{u})$ and $I(\vec{u})$. Using Equation 8.8, we exchange those samples across the distributions with probability $r(\vec{u}_I, \vec{u}_F)$;

$$r(\vec{u}_I, \vec{u}_F) = \frac{F(\vec{u}_I)I(\vec{u}_F)}{F(\vec{u}_F)I(\vec{u}_I)}. \tag{8.9}$$

For a general target distribution, this equation requires computing importance functions at arbitrary sample locations in different distributions as in Equation 8.8. However, we can simplify the equation in our method since we know that the sample $\vec{u}_F$ always returns $F(\vec{u}_F) = \frac{1}{V_c}$ by definition and it is always $I(\vec{u}) = 1$;

$$r(\vec{u}_I, \vec{u}_F) = \frac{F(\vec{u}_I)1}{\frac{1}{V_c}1} = V(\vec{u}_I). \tag{8.10}$$

The end result is straightforward. Since the sample $\vec{u}_I$ is from uniform sampling, there is no need to keep track of a Markov chain of $\vec{u}_I$; if uniform independent sampling generates a useful path (when $r(\vec{u}_I, \vec{u}_F) = V(\vec{u}_I) = 1$) we replace the current photon path, otherwise, we keep the current photon path and mutate normally (when $r(\vec{u}_I, \vec{u}_F) = V(\vec{u}_I) = 0$). Note, that we only need to keep a single Markov chain, $\vec{u}_F$, as a result.

8.4.3 Progressive Estimation of the Normalization Term

Using the uniform distribution in the replica exchange Monte Carlo method gives us another benefit than just preventing samples from getting stuck in a local region. By counting the samples from the uniform distribution, we can compute the normalization term V_c in a progressive fashion. This term is usually computed in a separate pass

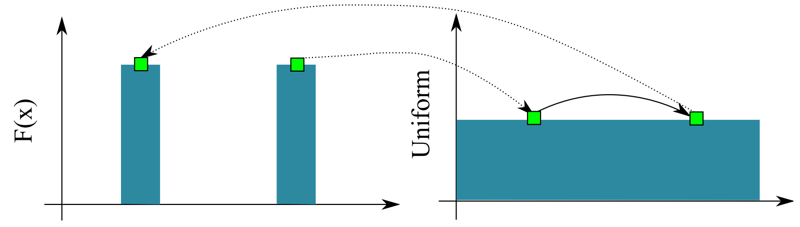


Figure 8.3: Example of the replica exchange Monte Carlo method with two distributions. The target distribution (left), $F(x)$, can have multiple peaks that are difficult to sample with standard Markov chain Monte Carlo methods. The replica exchange Monte Carlo method can improve exploration of a Markov chain by combining a uniform distribution with inter-distribution exchanges (right). A Markov chain in one peak can move to the uniform distribution, and later return to the other peak.

in existing Markov chain Monte Carlo rendering methods. The normalization term is estimated as:

$$V_c = \int V(\vec{u})\mathrm{d}\vec{u} \approx \frac{N_{I,V(\vec{u})=1}}{N_{I,\text{total}}}, \tag{8.11}$$

where $N_{I,V(\vec{u})=1}$ is the number of visible paths from uniform distribution, and $N_{I,\text{total}}$ is the total number of generated paths from a uniform distribution. Note, that $N_{I,total}$ is in fact equal to the total number of generated photon paths from $F(\vec{u})$, N_i^e, because we always generate a sample from the uniform distribution (Figure 8.2). Since this value is already kept for the purpose of radiance computation, the above computation of V_c requires keeping only one additional value, $N_{I,V(\vec{u})=1}$. The normalization constant, V_c, uniformly scales the radiance estimate computed for each pixel.

8.4.4 Differences from Independent Mutations

If we consider a sample from a uniform distribution, $\vec{u}_I$, as a "mutation", the acceptance probability is computed as:

$$a(\vec{u}_I \leftarrow \vec{u}_F) = \frac{F(\vec{u}_I)}{F(\vec{u}_F)} = \frac{\frac{1}{V_c}V(\vec{u}_I)}{\frac{1}{V_c}} = V(\vec{u}_I), \tag{8.12}$$

which indeed results in the same probability as Equation 8.10. One might therefore consider that our formulation of replica exchange Monte Carlo method as equivalent to a standard Markov chain Monte Carlo with independent mutations, such as the large step mutation [67]. This is not entirely true. There are two important differences in our formulation of the replica exchange Monte Carlo compared to independent mutations.

First, the formulation using the replica exchange Monte Carlo method shows that mutations and exchanges can be attempted at the same time. If we consider a sample from uniform distribution as an independent mutation, we need to exclusively select either an independent mutation or other mutations. This results in extraneous interference among independent mutations and other mutations, since we never perform both at the same time but select one of them with some user-specified probability. For example, Kelemen et al. [67] has such probability as "large step probability" and reported that the efficiency of the algorithm changes according to the value of this probability. Our formulation of replica exchange Monte Carlo method shows that this exclusive selection is not necessary. This in turn removes the extra user-specified parameter of the selection probability of an independent mutation.

Second, the replica exchange Monte Carlo method gives us a cleaner theoretical reason for keeping track of samples from the uniform distribution to compute the normalization term as we described. In the replica exchange Monte Carlo method, we can theoretically separate samples in the uniform distribution and the target distribution because they are simply from different distributions. In the formulation using independent mutations, we need to consider independent mutations as special cases and keeping track of all mutated samples. Note, that it usually does not make sense to keep track of mutated samples from a particular mutation strategy. For example, keeping all mutated samples from caustics perturbation [119] does not provide immediately meaningful information.

Although the implementation looks similar to independent mutations at a glance, the replica exchange Monte Carlo method keeps our method parameter-free and automatic, which is not the case with independent mutations. For example, if a scene only has visible photon paths, our algorithm shown in Figure 8.2 will automatically perform exactly the same as uniform sampling, which indeed is desirable. In other words, our method adaptively combines uniform sampling and Markov chain Monte Carlo sampling depending on how frequent uniform sampling generates a visible photon path. This is not the case with independent mutations, where sporadic local mutations introduce extra correlation among samples and the user needs to manually specify the selection

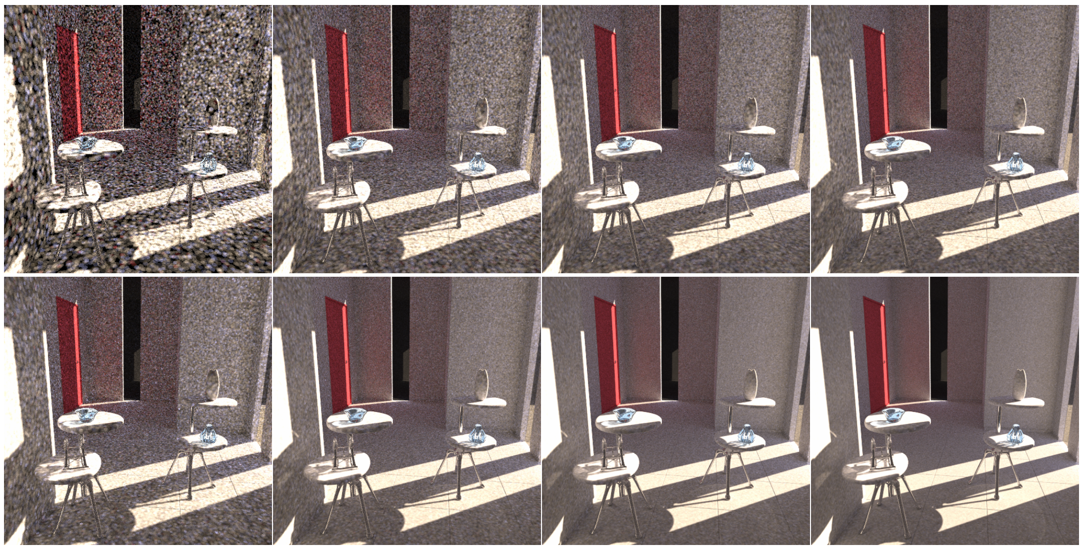


Figure 8.4: Sequences of rendered images of a room illuminated by a directional light source. The top row shows results with uniform sampling and the bottom row shows results with our method using the same rendering time (1, 15, 30, and 60 min). Our photon tracing method robustly and automatically handles scenes that are considered difficult to render with a photon tracing approach. The illumination is coming through the glass window and only photon tracing approaches can handle such paths without ignoring specular reflections and refractions at the window.

probability of independent mutations depending on scene settings.

8.5 Results

We implemented the uniform random photon tracing algorithm and our algorithm using the described splatting variation of stochastic progressive photon mapping. All scenes have been rendered on a 2.67GHz Intel Core i7 920 using one core. The alpha value is 0.7. We trace 200K photons per one eye ray tracing pass. The initial radius are manually chosen for each scene as a constant to get approximately four pixel-wide contribution on the image from each photon at the beginning. This manual tweaking of the initial radius is orthogonal to our claim that the proposed photon tracing algorithm is automatic and parameter-free. The resolution of the images is 512^2. Table 8.1 summarizes various statistics of our experiments. The calculated acceptance ratio is 23.4% for all scenes except the ones where our method does not provide an improvement.

Table 8.1: Statistics of our experiments. The table shows the number of triangles, ratio of the number of visible photons in total between our method/uniform sampling (larger value means more photons are visible in our method), rendering time in minutes, and the adaptive mutation size at the end of the rendering process. Cognac0-4 correspond to different zoom ratios (far to near), and Buddha (far/near) corresponds to the far/near viewpoint.

Scene	Triangles	Visible Photons Ratio	Time	Mutation Size
Cognac0	16456	103.1	120	0.274
Cognac1	16456	363.8	120	0.176
Cognac2	16456	885.2	120	0.174
Cognac3	16456	910.4	120	0.168
Cognac4	16456	3284.2	120	0.056
Pocket Watch	152434	270.3	90	0.194
Room	160400	41.9	60	0.253
Cornell w/door	730	55.9	90	0.096
Cornell	36	2.0	90	9.852
Box	3462	3.9	90	3.877
Buddha (far)	378731	6.67	90	0.173
Buddha (near)	378731	54.4	90	0.106

Figure 8.5 compares rendering of a cognac glass illuminated by a directional light source with different zoom ratios. The caustic below the glass is a specular-diffuse-specular path due to a directional light source, and consequently eye ray tracing approaches cannot render this scene since the probability that a path started from the eye hits a directional light source is zero. Only progressive photon mapping can render this scene robustly. To illustrate the effect of our sampling method, we zoom into the caustics such that the illuminated region that is visible from the viewer becomes increasingly small. This means that a large number of photons land outside the view with uniform sampling. Our sampling method focuses photons in the visible region, and we can obtain a significantly better images in the same rendering time regardless of the viewpoint. The ratios of visible photons in Table 8.1 also show that our method focuses increasingly more photons compared to uniform sampling.

Figure 8.4 shows a sequence of rendered images of a room illuminated through a glass window by a directional light source. The images generated by our method quickly converge to visually pleasing results compared to uniform sampling. Our method also

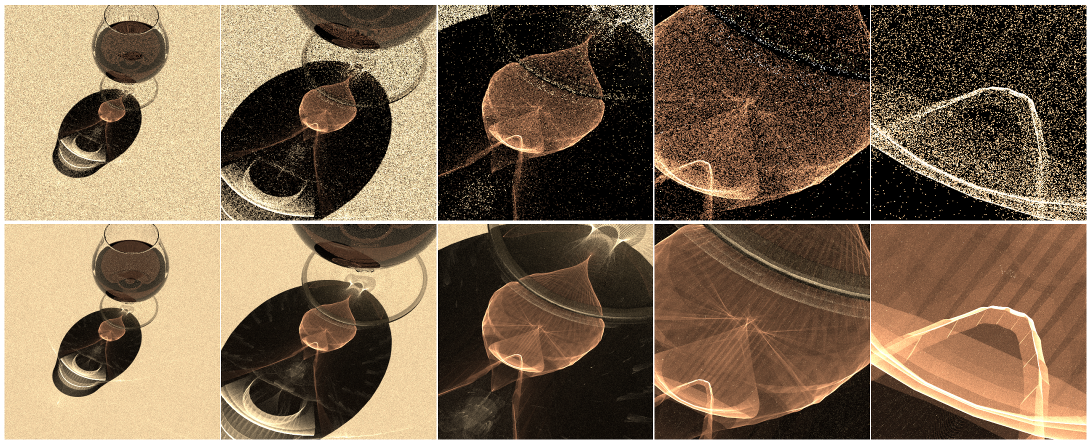


Figure 8.5: Cognac glass illuminated by a directional light source. We compare rendered images using progressive photon mapping with the same rendering time (120 min), but with different photon tracing algorithms. The images on the top row are rendered using random sampling of photons, which become increasingly noisy as we zoom in on the caustic. Using our photon tracing method (bottom row), we can focus tracing photons that contribute to the image without any portal and render the close-ups with less noise in the same rendering time. Note that no other existing global illumination methods can render illumination under the cognac glass accurately, since this illumination comes from specular-diffuse-specular paths from a light source with zero solid angle. The combination of progressive photon mapping and our photon tracing technique is the first method that works effectively and robustly in this scene. The stripe patterns in the caustic are not artifacts of our method - they are caused by the tessellation of the cognac glass.

has a lower numerical error in the same rendering time as shown in Figure 8.6. For example, the average error achieved in 25 minutes using our method is as close as the average error achieved in 300 minutes using uniform sampling.

Figure 8.7 highlights the effect of the adaptive mutation size. A mutation size that is too large results in an image with as much noise as the image rendered using uniform sampling, and mutation size that is too small is noisy as well compared to the adaptive mutation size. Our photon tracing method based on adaptive Markov chain Monte Carlo gives us the best results without any tuning of parameters. Note, that in a standard Markov chain Monte Carlo method, we do not know which mutation size works well unless we actually try a wide range of mutation size (0.01 to 4.0 in our example) and compare rendered images.

Figure 8.11 shows examples where our method will not provide a benefit as most

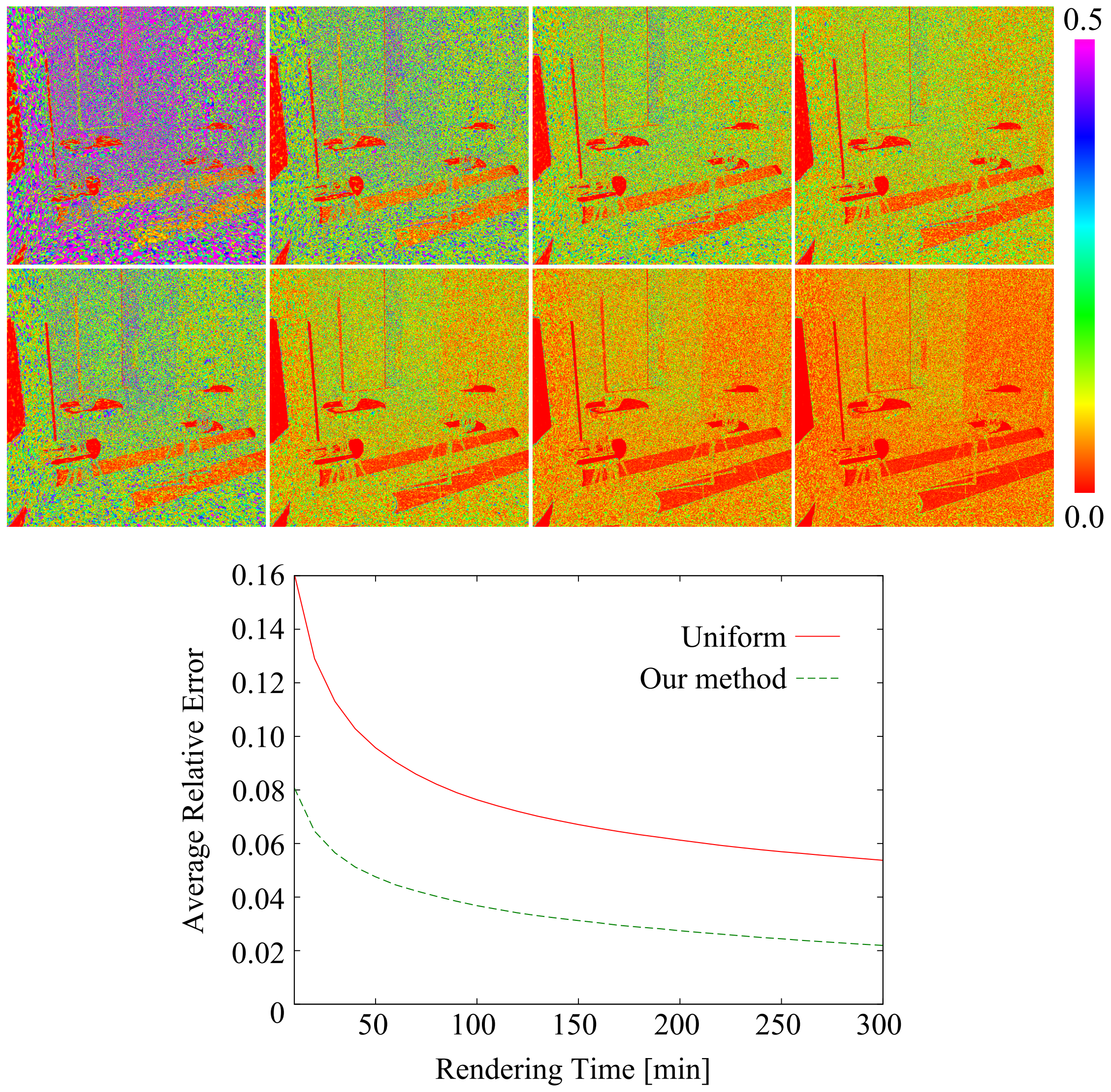


Figure 8.6: Color-coded error images of Figure 8.4 and the graph compares the average errors of rendered images with uniform sampling and our method. Our method not only generates visually smoother images, but also results more accurate solutions in the same rendering time.

of paths are already visible. For such scenes, sample correlation introduced by a Markov chain Monte Carlo sampler would just add extra noise. Although no improvement can be seen with our method, the results show no visible negative effect when we compared them to uniform random photon tracing method. This is because the exchange by uniform random sampling happens often in such scenes, and our algorithm automatically uses uniform random sampling for many photons (Figure 8.2). Even though some paths are not visible, the adaptive Markov chain Monte Carlo method automatically increases the mutation size to minimize correlation of samples due to dependent sampling scheme of Markov chain Monte Carlo method. The mutation sizes in Table 8.1 for Cornell and Box therefore are noticeably larger than other scenes as we wanted. Note also that this Box scene is exactly the same scene that has been demonstrated to be rendered poorly with other rendering methods. Figure 8.13 shows that our method preserves robustness of progressive photon mapping, thus being able to handle scenes that are difficult to unbiased methods.

Table 8.1 provides some idea on when we will see benefits using our photon tracing method. The column of "Visible Photon Ratio" shows how many times more photons become visible using our method. For the scenes that show no benefit (Cornell and Box), the ratio is less than 10. As we can see in Figure 8.4 and Figure 8.6, the Room scene already shows some benefit with our method, and the ratio is 41.9, thus the ratio may need to be more than a few tens to obtain improvement using our method.

We also demonstrate the effect replica exchange in Figure 8.10. We rendered the same scene with and without the replica exchange procedure. In this example, we used the same fixed mutation size to isolate the consequence of using replica exchange. Without replica exchange, photon paths can be trapped within one of the windows for long time due to isolated regions of $V(\vec{u}) = 1$. The replica exchange Monte Carlo method alleviates this issue automatically, resulting in plausible images in the sane rendering time. Since we use a hypercube of random numbers as the sampling space, our method can handle difficult local lighting similar to Metropolis light transport without any modification as shown in Figure 8.12.

We show effect of using different target acceptance ratio other than 23.4% in Figure 8.8. We have rendered the same scene as in Figure 8.12 using different target acceptance ratios. Note that difference in the two images with 23.4% and 40% as the target ratios is rather small, which indicates our method is not very sensitive to the target acceptance ratio. The images with extreme target acceptance ratios (close to 0% or 100%) are slightly noisier than the image with the target acceptance ratio of 23.4%, which supports that the adaptation based on the general principle is valid.

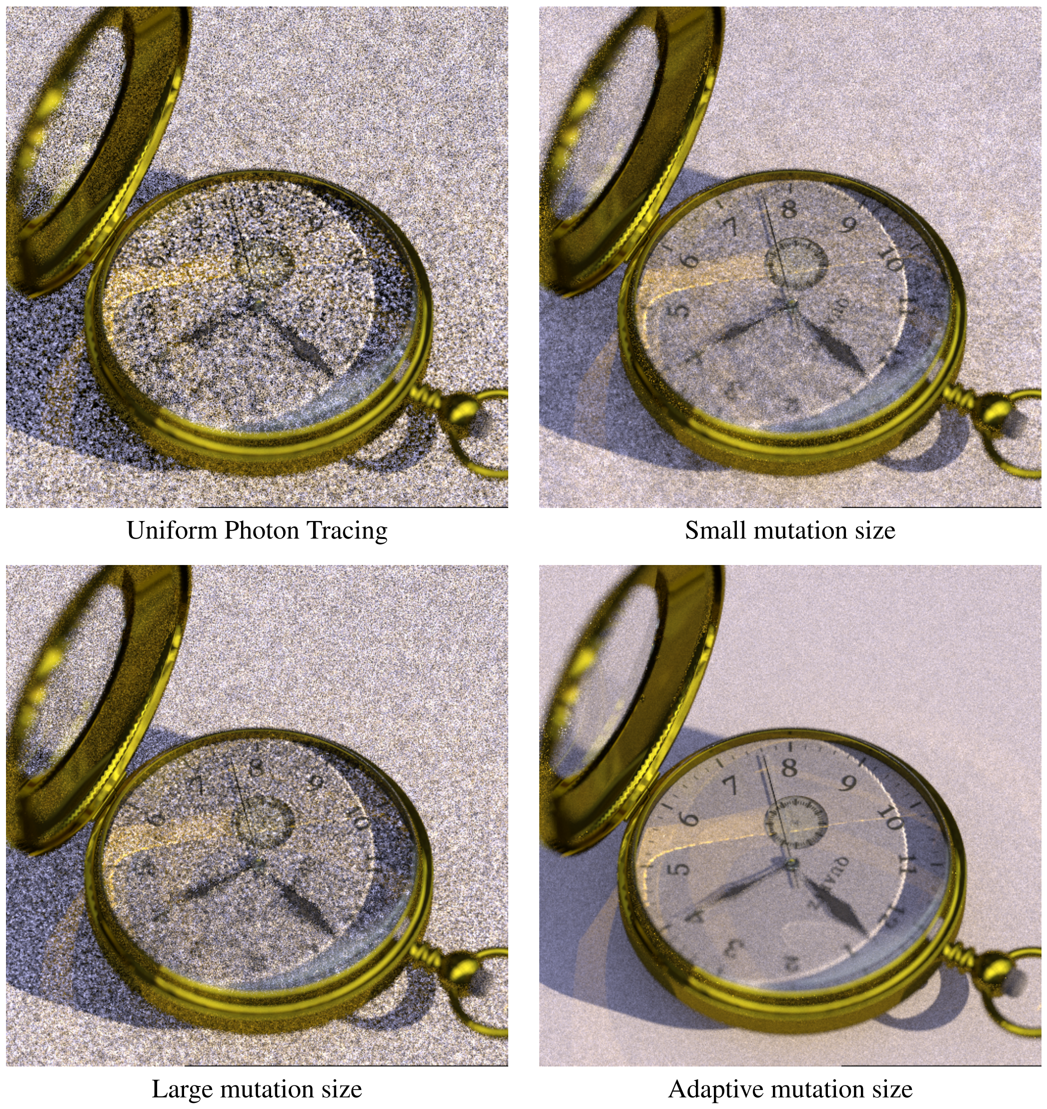


Figure 8.7: Effect of adaptive mutation size. A pocket watch is illuminated by a hemispherical light source and a directional light source and is rendered with depth-of-field. The illumination on the dial plate is due to caustics from the glass cover and the metal lid. The images shown are rendered by uniform random sampling and our photon tracing method in the same rendering time. The top-right image uses a mutation that is too small ($d_i = 0.01$) and the bottom-left image uses a mutation that is too large ($d_i = 4.0$). The adaptive Markov chain Monte Carlo used in our method produces the best result without any parameter tuning.

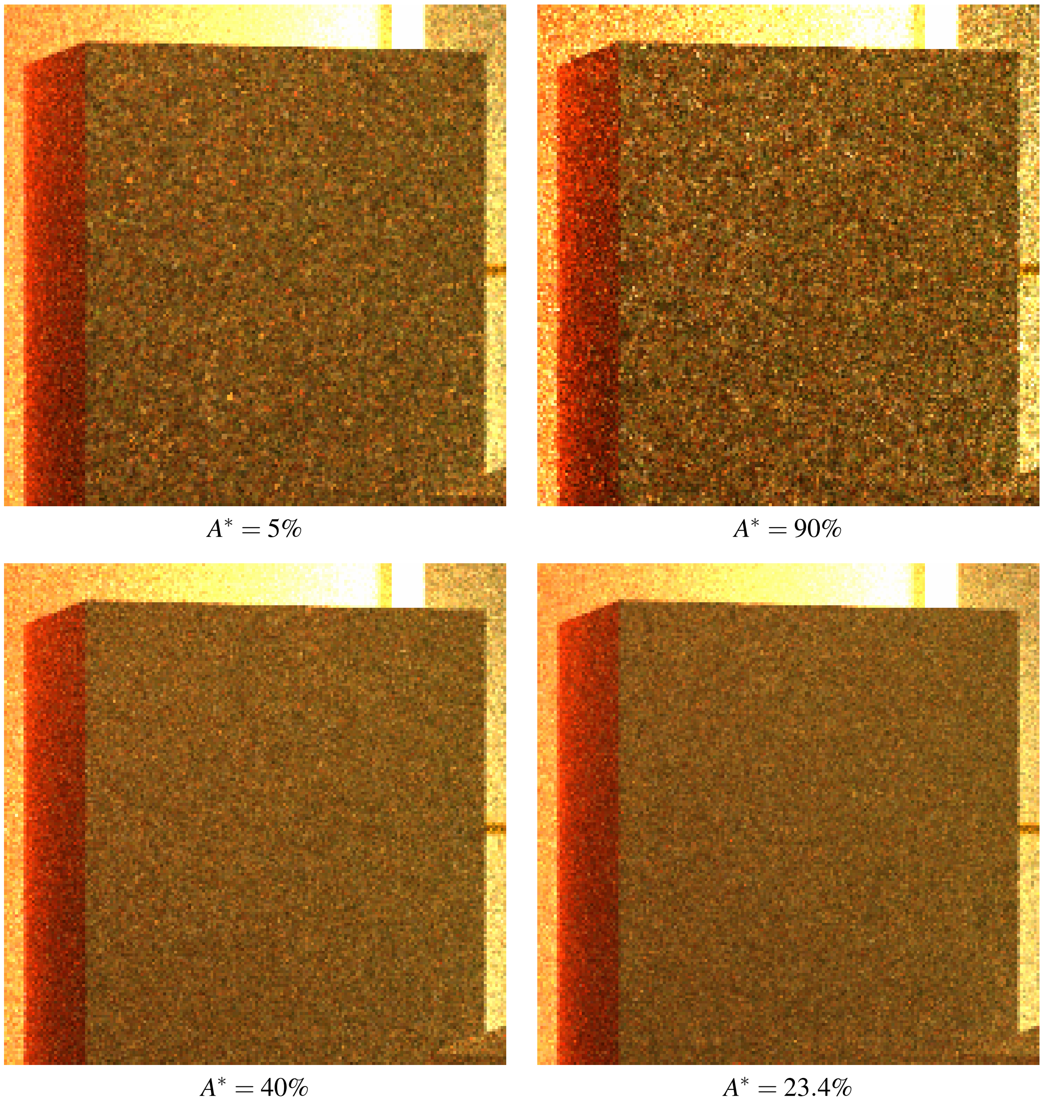


Figure 8.8: Close-ups of rendered images using different target acceptance ratios. We have rendered the same scene as in Figure 8.12 using different target acceptance ratios using the same rendering time (90 min). Calculated acceptance ratios achieved are the same as the target ratio. The target acceptance ratios closer to 0% or 100% result in slightly noisier images, but using intermediate values would not affect the efficiency of our algorithm.

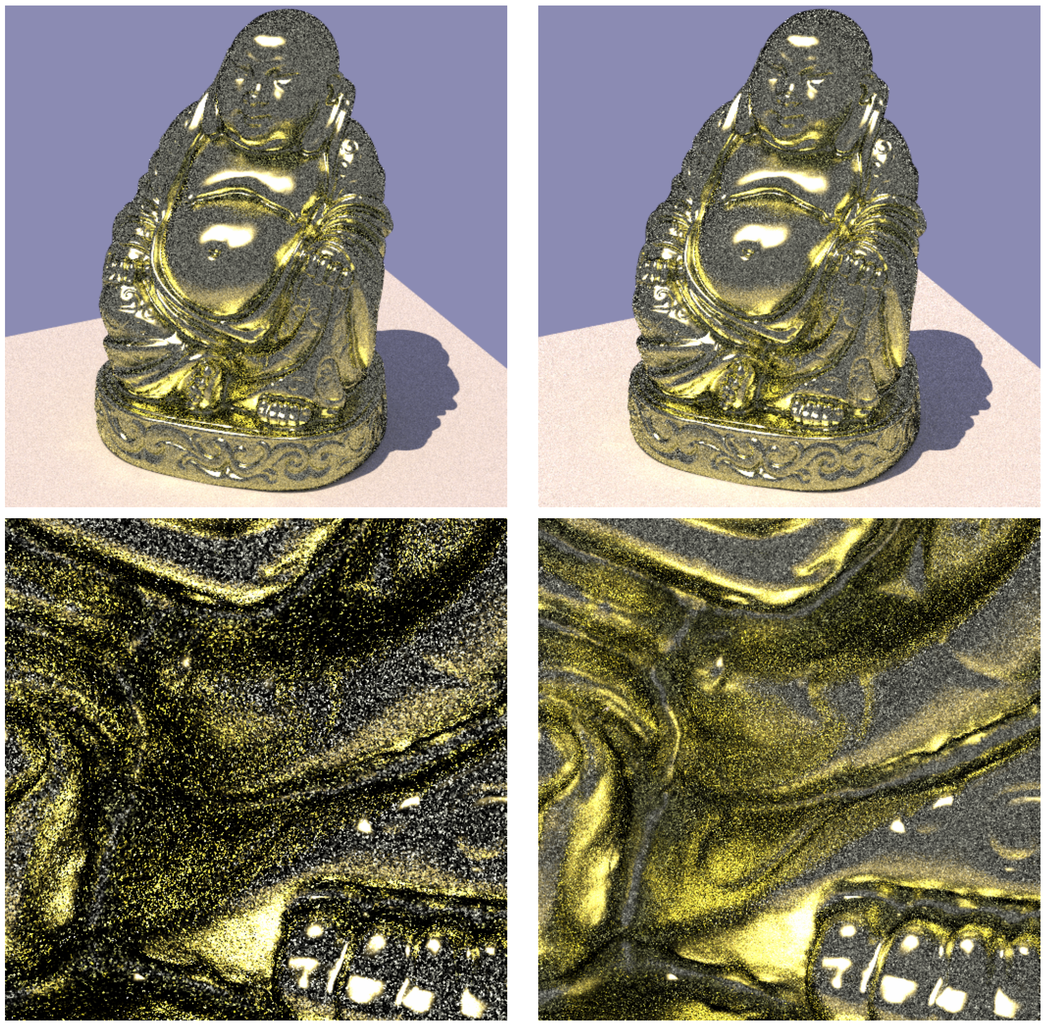


Figure 8.9: Example scene where the variation in BRDFs is the main source of rendering error. The statue is rendered using modified-Phong with the exponent 100 under a hemispherical light source and a directional light source. The rendering time is 90 min. Although our method shows improvement in the close-up rendering, both uniform random sampling and our method perform about the same with the distant viewpoint as our method does not resolve flux variations due to BRDFs. Note, that we did not employ stochastic progressive photon mapping in this scene to highlight this limitation. Stochastic progressive photon mapping and our new sampling method would render this scene without problems.

8.6 Discussion

8.6.1 Comparisons with Other Importance functions

In the standard implementation of Metropolis light transport, the importance function is usually given as a brightness of each path [67, 119]. Using this importance function, each path contributes the same brightness to the image. This importance function achieves importance sampling according to outgoing radiance within each pixel. This choice is probably motivated by that a probability density function proportional to the integrand results zero error in importance sampling with a regular Monte Carlo method.

However, the issue is that samples are distributed across an image in Metropolis light transport, not just within a single pixel. This essentially means we solve multiple integrations at the same time. As a result, bright pixels get more samples compared to dark pixels as we mentioned in Section 8.2.2, which results in poor stratification of samples over the image. Energy redistribution path tracing [15] improves stratification of samples by starting multiple independent chains from stratified points on the image. However, even if the initial points are well distributed, Markov chains can get stuck in paths with very large brightness for a long time, which results in notorious "firefly" noise. Cline et al. thus proposed post-process filtering, which unfortunately makes the algorithm inconsistent (i.e., does not converge to the correct solution). Although our target function does not provide stratification on the image, it does not have any local peak that is prone to this issue. Isolated visible paths can still lead to firefly noise, however, the replica exchange Monte Carlo method alleviate the problem in such cases since the exchange results in a completely different path as soon as we find another visible path by uniform sampling.

Another minor issue is that, with the conventional brightness function, we cannot define contribution of each photon path as a single value because a single photon path can contribute to multiple pixels. In order to apply Markov chain Monte Carlo methods, we need to define a single importance value, which is not well defined with the brightness function in combination with photon tracing. Our visibility function is well defined since any visible photon path.

An interesting observations on alternative target functions has recently been made by Hoberock and Hart [55]. They proposed a multi-pass algorithm that adjusts the importance function using information from previous passes, such as brightness and variance of each pixel. Their work also supports that just using the brightness of each path is not necessarily the optimal choice. They additionally pointed that a constant importance function does not work because no importance sampling will be employed, which may be confusingly similar to our importance function. The important distinction is that our importance function returns 0 for a path that is not visible, and it is applied to photon tracing. Since photons are naturally distributed according to incoming radiance if proper local importance sampling and Russian Roulette is done (i.e., photon density is equal to radiance), our method still can perform importance sampling of incoming radiance, even with the very simple target function.

8.6.2 Limitations

One limitation of our method is that it ignores flux variation due to a BRDF lobe. For example, if a scene has a highly glossy material, just using visibility information will not resolve flux variation due to a BRDF lobe. Figure 8.9 demonstrates such an example scene, where most of noise is due to a highly glossy BRDF. Although our method still provides some improvement when we render the close-up, it does not resolve noise due to the glossy BRDF. Note that this scene itself might be efficiently rendered with other methods, such as path tracing with the next event estimation, but we chose this scene to highlight the limitation of our method. One possible solution is to use stochastic progressive photon mapping to perform importance sampling of a such BRDF from the eye, which resolves the flux variation. However, if a scene consists of only highly glossy materials, rendering will entirely be done by tracing rays from the eye, which could be highly inefficient if a light source is small and cannot be sampled with shadow rays (e.g., a filament within a light bulb). It is interesting as future work to investigate how we can take a glossy lobe into account in our method while maintaining its simplicity. Note that this is not a trivial problem, since a highly glossy BRDF returns a large value, which would unnecessarily attract more photons similar to bright pixels in Metropolis light transport.

Another limitation is that the adaptive procedure is done globally. Using locally adaptive mutation parameters might improve convergence speed as was proposed in computational statistics [6]. For example, we might be able to use an adaptive grid to store mutation parameters locally, and use those parameters depending on the current state (i.e., position in the hypercube). This however is challenging as our sampling space is in high dimensional space, where the cost of storing any local estimation including adaptive mutation parameters is often prohibitive and reliable estimation is difficult due to the curse of dimensionality.

Our algorithm does share the limitation with the other Markov Monte Carlo chain rendering algorithms that the samples are not stratified over an image. It is interesting to investigate as future work whether the adaptive importance function proposed by Hoberock and Hart [55] is applicable to our method in order to improve stratification.

8.6.3 Dynamic Target Distribution

One theoretical difficulty of applying any Markov chain Monte Carlo method to progressive photon mapping is that the target distribution, thereby the importance function, changes as the number of samples increases. This is because progressive photon mapping updates the radii of the measurement points to ensure convergence to the correct solution. In our approach, this results in changes of the region where $V(\vec{u}) = 1$. Although we have not found any apparent failure cases, the theoretical behavior of Markov chain Monte Carlo methods on the dynamic target distribution in progressive photon mapping is not fully analyzed. Our combination with the adaptive Markov chain Monte Carlo method further complicates this theoretical validation. Convergence of the normalization term may also require careful theoretical analysis. In this paper, we thus do not claim provable convergence to the correct solution using our photon tracing algorithm.

However, since we always use the current radii to distribute the photon power, the contribution is at least computed based on the current distribution. It is only the stationary sample distribution that is not analyzed. This separation might be helpful for further theoretical analysis. We also believe that, even without a formal theoretical validation, our method will be useful for many practical applications that do not require theoretical guarantees of consistency. In the end, provable convergence is only a theoretically appealing property as we cannot take infinite number of samples in practice. We see practical benefits of using our method through numerical experiments as we have demonstrated.

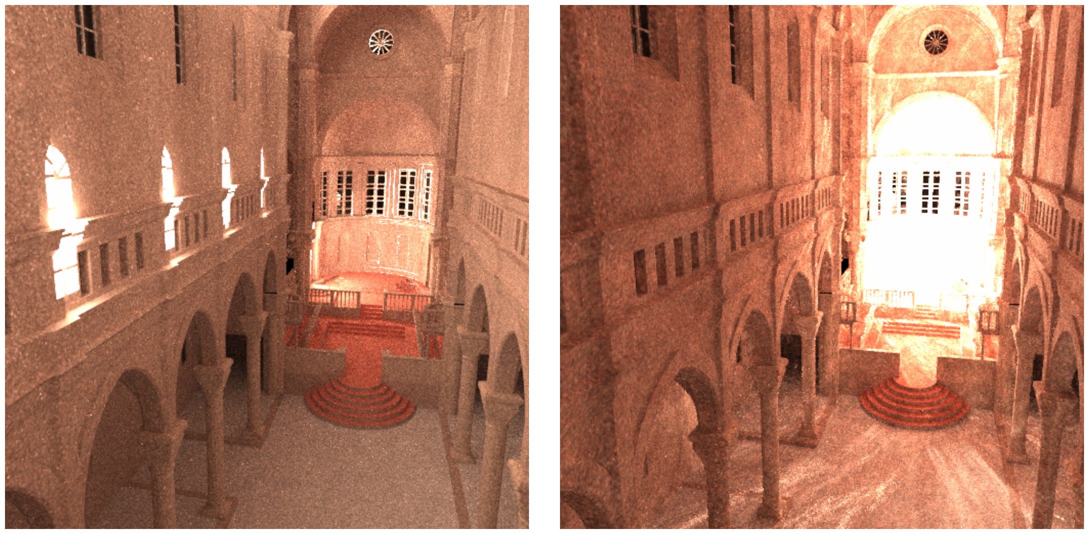


Figure 8.10: Effect of replica exchange. We rendered Sibenik cathedral with a directional light source as the only light source in 90 min with (left) and without (right) replica exchange. Without replica exchange, samples can get stuck within a small region (i.e., single window) for long time, resulting in a solution with very high correlation of samples.

8.7 Conclusion

We have presented a new photon tracing algorithm based on a simple and effective importance function using the visibility of photon paths. Our algorithm is based on recent developments in Markov chain Monte Carlo methods. The resulting algorithm does not have any parameters that require fine tuning by the user, and its implementation is strikingly simple. We have demonstrated that our algorithm robustly handles scenes that are difficult for existing photon tracing approaches and that it also works well in simple scenes. The combination of our algorithm and progressive photon mapping is an effective, unified, and robust solution to many light transport configurations. Although we used progressive photon mapping, we expect that the same importance function can be used for other light transport algorithms based on photon density estimation.

8.8 Acknowledgements

The material in this chapter is, in part, a reproduction of the material in Toshiya Hachisuka and Henrik Wann Jensen. “Robust Adaptive Photon Tracing using Photon Path Visibility”, ACM Transactions on Graphics, in publication, 2011 [44]. The dissertation author was the primary investigator and author of this paper. We would like to thank Youichi Kimura (Studio Azurite) for providing us the room model. The Buddha statue model is courtesy of Jotero.com and the Sibenik model is courtesy of Marko Dabrovic.

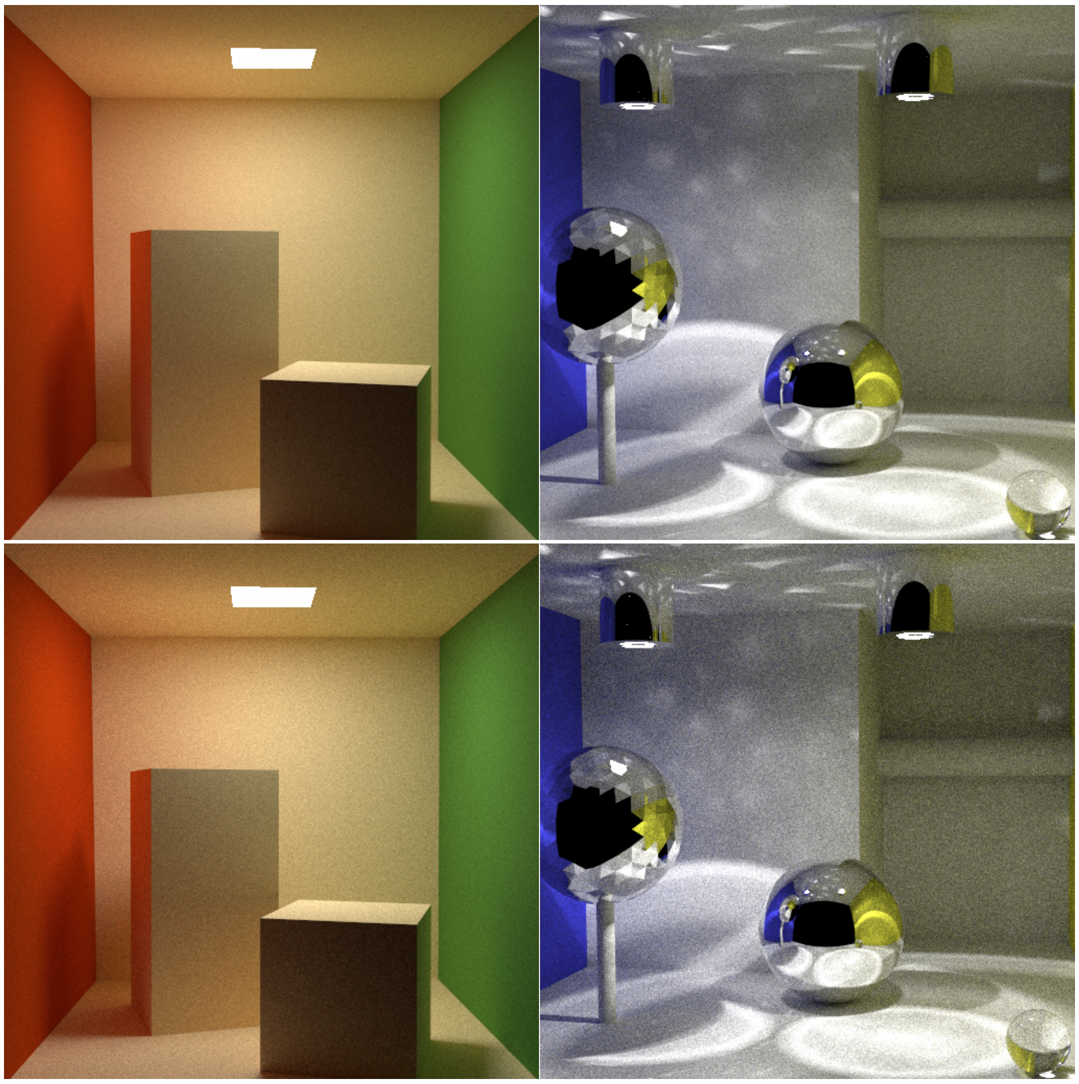


Figure 8.11: Rendered images of scenes where our method does not provide benefits. Our method (top) still performs as well as uniform random sampling (bottom) does in the same rendering time (90min). Note that Box scene is the one that is demonstrated to be rendered robustly only with progressive photon mapping.

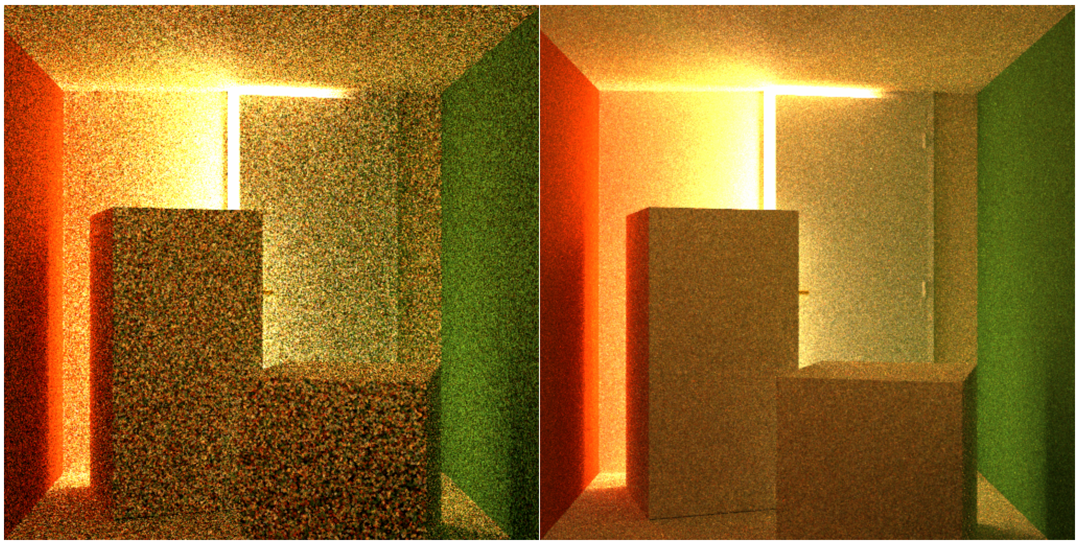


Figure 8.12: Cornell box with a small gap with a door. The images are rendered with uniform sampling (left) and our method (right) in 90 min. Our method works under local lighting without any modifications to the algorithm.

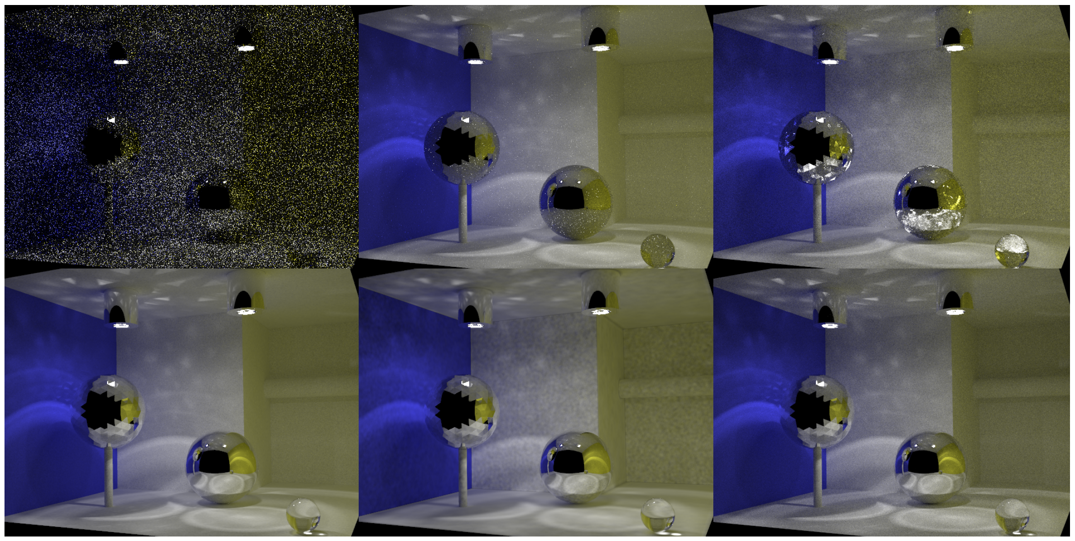


Figure 8.13: Equal-time comparisons with other rendering methods (four hours, except for conventional photon mapping which is limited by memory consumption). The images are rendered with, from left to right, top row: path tracing, bidirectional path tracing, Metropolis light transport, and bottom row: original progressive photon mapping, conventional photon mapping with 20M photons, and progressive photon mapping with our photon tracing method. Our method does not impair the robustness of original progressive photon mapping. The small differences in the result from our method are due to the use of different implementations.

Chapter 9

Hybrid Progressive Density Estimation

Progressive photon mapping robustly handles many light paths that existing unbiased Monte Carlo ray tracing cannot even capture. However, there are scene configurations where unbiased Monte Carlo ray tracing is more efficient than progressive photon mapping such as direct illumination from diffuse light sources, highly glossy reflections, and diffuse indirect illumination. This chapter introduces a unified framework based on multiple importance sampling [118] that adaptively combines progressive photon mapping and unbiased Monte Carlo ray tracing. The contributions are two-fold: first, we provide a unified view of local path sampling and photon density estimation, which will provide more solid ground for robust light transport simulation algorithms; second, we extend the theoretical analysis of optimal multiple importance sampling strategies by considering the presence of biased methods. The resulting framework improves the efficiency for rendering various scene settings such as light sources with sharp directional distributions, glossy reflections, complex caustics, and diffuse interreflections.

9.1 Background

Before going into the detail of the new framework, we summarize the original multiple importance sampling framework [118]. Multiple importance sampling is a powerful technique that allows the use of multiple sampling techniques with different probability densities to solve a single integral.

Suppose that we need to compute an integral

$$I = \int_{\Omega} f(\mathbf{x}) d\mu(\mathbf{x}) \tag{9.1}$$

over some domain Ω. Standard Monte Carlo integration generates samples in Ω according to a probability density p to build an unbiased estimator of I:

$$I \approx \frac{1}{N} \sum_{j=1}^{N} \frac{f(X_j)}{p(X_j)}. \tag{9.2}$$

With multiple importance sampling, we can use n different techniques to generate samples in Ω, each with a different probability density p_i approximating different parts of the integrand f. We then weight the contribution of individual samples to build an estimator for the given integral. If the i-th technique is used to generate N_i samples $\{X_{i,j} : j = 1, ..., N_i\}$, the following formula gives an unbiased estimator of I:

$$I \approx \sum_{i=1}^{n} \frac{1}{N_i} \sum_{j=1}^{N_i} \frac{w_i(X_{i,j}) f(X_{i,j})}{p_i(X_{i,j})} \tag{9.3}$$

as long as

$$\sum_{i=1}^{n} w_i(\mathbf{x}) = 1 \tag{9.4}$$

and $w_i(\mathbf{x}) = 0$ whenever $p_i(\mathbf{x}) = 0$.

Veach and Guibas [118] introduced a few weighting strategies which are provably good, in the sense that the resulting error is within a constant from that of the best possible weighting strategy. One such strategy is the *balance heuristic* defined as

$$w_i(\mathbf{x}) = \frac{N_i p_i(\mathbf{x})}{\sum_{k=1}^{n} N_k p_k(\mathbf{x})}. \tag{9.5}$$

The derivation of this strategy assumes that all techniques are unbiased since the derivation assumes that error is solely characterized by variance.

9.2 Overview

Multiple importance sampling is used in light transport simulation in the form of bidirectional path tracing [118]. Bidirectional path tracing considers a family of complete paths by connecting two sub-paths traced from the eye and a light source. By changing the length of sub-paths, we can obtain multiple path sampling techniques with different probability densities that can sample the exact same path. Veach and Guibas [118] showed that we can apply multiple importance sampling to combine those multiple techniques by considering probability densities of complete paths.

Bidirectional path tracing captures many different types of illumination. However, fundamental limitations of unbiased sampling methods do not allow bidirectional path tracing to capture some important light transport phenomena such as illumination due to specular-diffuse-specular light paths [117]. As previous chapters demonstrated, capturing such paths is only practical with progressive photon mapping at the moment.

At the same time, bidirectional path tracing can capture some illumination more efficiently than progressive photon mapping. For example, in Figure 5.6, although the result of bidirectional path tracing has significant amount of noise on the torus, the ground plane exhibits less noise than the result of progressive photon mapping. It is thus desirable to be able to adaptively use two different methods within the same scene.

The key idea of the new framework, *multisampled progressive photon mapping*, is to have a generalized rendering algorithm that adaptively combines both (stochastic) progressive photon mapping and Monte Carlo ray tracing by means of multiple importance sampling. Figure 9.1 illustrates this key idea.

Including progressive photon mapping in multiple importance sampling poses two theoretical challenges. First, we need to characterize the effect of bias in provably good strategies of multiple importance sampling. This is because progressive photon mapping is a biased estimator, where error is affected by both variance and bias, whereas the original derivation of such strategies [118] assumes that all the estimators are unbiased. Second, as we briefly discussed in Chapter 5, progressive photon mapping has slower asymptotic convergence rate than Monte Carlo ray tracing. We show that absorbing the difference in convergence rate is necessary in order to retain the optimality of a provably good combination in multiple importance sampling.

The proposed framework can also be seen as a principled way to remove a heuristic binary classification of materials for handling glossy reflections in stochastic progressive photon mapping. Such classification is necessary in stochastic progressive photon mapping since we trace additional bounce rays for materials classified as glossy. The framework removes such a classification by adaptively combining the results obtained by both choices of this classification under multiple importance sampling.

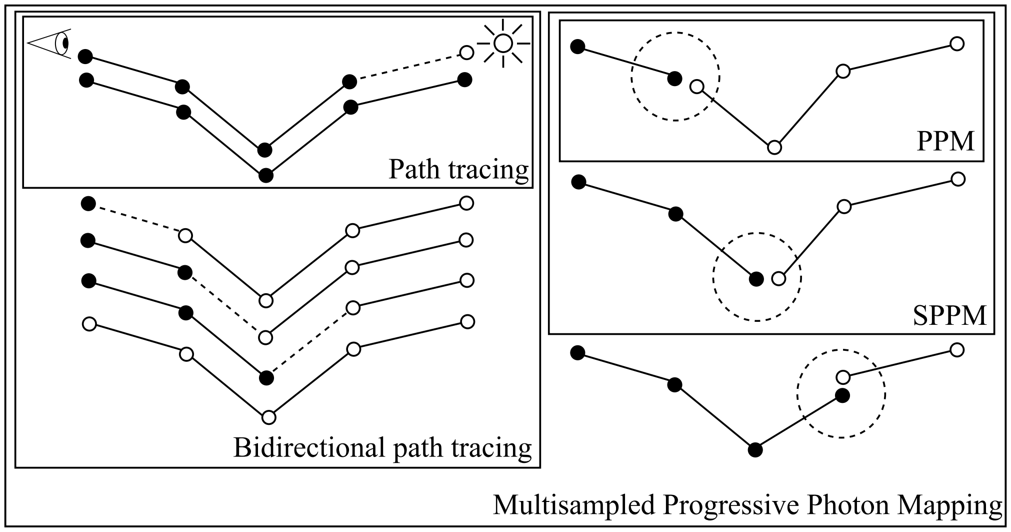

Path tracing
Bidirectional path tracing
PPM
SPPM
Multisampled Progressive Photon Mapping

Figure 9.1: Key idea of of multisampled progressive photon mapping. Multisampled progressive photon mapping subsumes all the previous sampling techniques including path tracing, bidirectional path tracing, progressive photon mapping (PPM), and stochastic progressive photon mapping (SPPM), and also introduces a new technique to sample the same path. The white circles indicate vertices sampled from the light source, and the black circles indicate vertices sampled from the eye. The dotted lines/circles show sub-path connections using shadow rays and photon density estimation, respectively.

9.3 Unified Framework of Generalized Path Sampling

Our framework introduces a new way to connect sub-paths from light sources and the eye by means of photon density estimation. Figure 9.1 illustrates the difference between the regular connection using a shadow ray in bidirectional path tracing and the new connection using photon density estimation. By considering photon density estimation as another technique to sample the full path, our framework provides a unified view over Monte Carlo path tracing such as bidirectional path tracing and photon density estimation methods such as progressive photon mapping.

In order to fully incorporate photon density estimation into multiple importance sampling, we need both the ability to evaluate the probability density function of photon density estimation for *any* path, and the ability to evaluate the probability density of *all other* techniques for a path that was generated by photon density estimation. In other words, we need to answer this question: "Given a path sampled by technique A, what would be the probability density of sampling the same path with technique B?". For example, technique A can be progressive photon mapping, and technique B can be any of the bidirectional path tracing techniques.

Table 9.1: Descriptions of the notations.

Notation	Description
e_i	ith vertex generated from the eye
l_i	ith vertex generated from the light source
E_i	ith segment of the eye sub-path ($E = [e_i, e_{i+1}]$)
L_i	ith segment of the light sub-path ($L = [l_i, l_{i+1}]$)
$p(S_i\|S_{i-1})$	PDF of generating the segment S_i given S_{i-1}

9.3.1 Path Probability Density using Segments

To succinctly define the probability density of photon density estimation, we introduce a notation of segments in our description. We define a *segment* of a path, $S_i = [s_i, s_{i+1}]$, as a part of the path that consists of two vertices s_i, s_{i+1}. We also define $p(S_i|S_{i-1})$ as the probability density of the segment S_i given the segment S_{i-1}. This probability density is usually proportional to probability density of sampling the direction along S_i given the incoming direction along S_{i-1} measured with respect to area. For example, this could be the probability density of a ray direction according to the BRDF at s_i times the geometric term between s_i and s_{i+1}. Figure 9.2 illustrates this concept.

Using our notations, the probability density of a complete path, $\bigcup_{i=1}^{M} S_i$, of length $M = M_L + M_E$ using photon density estimation with a light sub-path of length M_L and an eye sub-path of length M_E is

$$p\left(\bigcup_{i=1}^{M} S_i\right) = p\left(\bigcup_{i=1}^{M_L} L_i\right) p\left(\bigcup_{i=1}^{M_E} E_i\right), \tag{9.6}$$

where

$$p\left(\bigcup_{j=1}^{i+1} E_j\right) = p\left(\bigcup_{j=1}^{i} E_j\right) p(E_{i+1}|E_i)$$
$$p(E_1) = p(e_1)p(E_1|\emptyset) \tag{9.7}$$

and

$$p\left(\bigcup_{j=1}^{i+1} L_j\right) = p\left(\bigcup_{j=1}^{i} L_j\right) p(L_{i+1}|L_i)$$
$$p(L_1) = p(l_1)p(L_1|\emptyset). \tag{9.8}$$

We defined $p(E_1)$ and $p(L_1)$ as special cases since these terms denote probability densities of generating a first eye segment and a first light segment that do not have incoming directions.

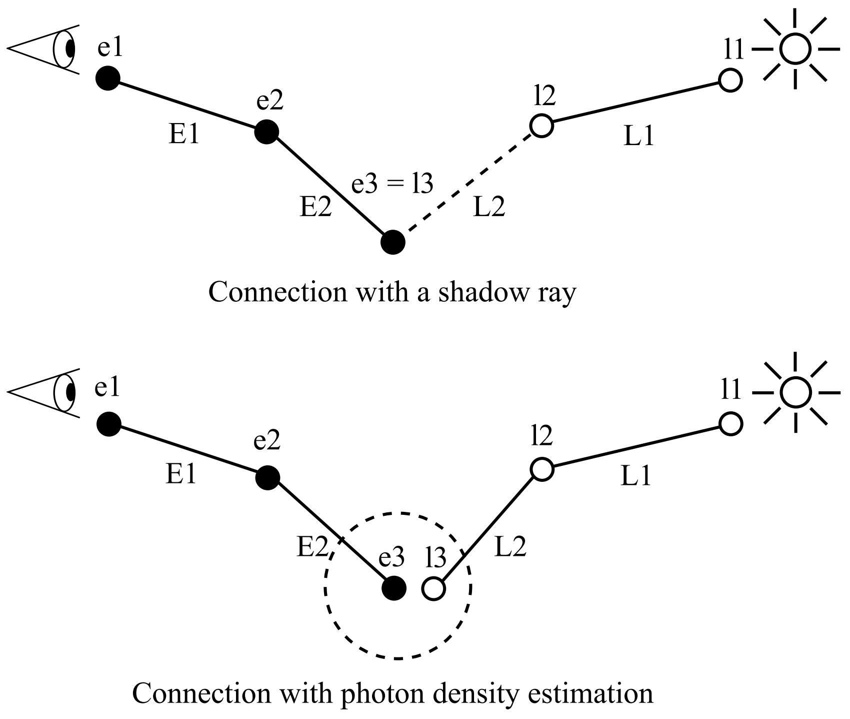

e1
e2
E1
E2
e3 = l3
L2
l2
L1
l1
Connection with a shadow ray
e1
e2
E1
E2
e3
l3
L2
l2
L1
l1
Connection with photon density estimation

Figure 9.2: Sub-paths connections with a shadow ray and photon density estimation. Bidirectional path tracing connects a light sub-path and an eye sub-path by tracing a shadow ray between the endpoints of the both sub-paths (top). We introduce another approach to connect light sub-paths and eye sub-paths based on photon density estimation (bottom). The figure also provides an example of the notation defined in Table 9.1.

This segment-based formulation can cleanly express the fact that photon density estimation is approximately connecting two sub-paths by simply considering the case where $e_{M_E+1} \neq l_{M_L+1}$. This is an important difference from the original formulation by Veach [117] that is based on path vertices.

When we evaluate the probability density of a path that was generated by another technique for photon density estimation, we just set $e_{M_E+1} = l_{M_L+1}$ and use the equation above. Since segments are just labels of vertices, this formulation is compatible with the vertex-based formulation by Veach and Guibas [118], when we need to compute probability densities of paths for unbiased path sampling techniques. We refer to their original paper for derivations of probability density functions for the unbiased sampling techniques. Note that light tracing [8, 23] is a special case of photon density estimation where forcing $e_2 = l_{M_L}$ by explicitly connecting e_1 with l_{M_L}.

Some care however must be taken when we evaluate the probability density of a path that was generated by photon density estimation for other techniques. Recall that, in order to fully leverage multiple importance sampling, we need to answer this question: "Given a path sampled by technique A, what would be the probability density of sampling the same path with technique B?". Since the two vertices at the connection using photon density estimation can never be different under unbiased techniques ($e_{M_E+1} \neq l_{M_L+1}$), we use the following heuristic to choose one of the vertices to compute the probability density in this case.

- **The technique would generate this vertex at the connection from the eye:** We use $p(E_{M_E}|[e_{M_E+1}, l_{M_L}])$ as $p(E_{M_E}|L_{M_L})$ for a technique that contains the segment L_{M_L} in the eye sub-path (i.e., the probability density function at e_{M_E+1}).

- **The technique would generate this vertex at the connection from the light:** We use $p([e_{M_E}, l_{M_L+1}]|L_{M_L})$ as $p(E_{M_E}|L_{M_L})$ for a technique that contains the segment L_{M_L} in the light sub-path (i.e., the probability density function at l_{M_L+1}).

Figure 9.3 illustrates this vertex selection heuristic.

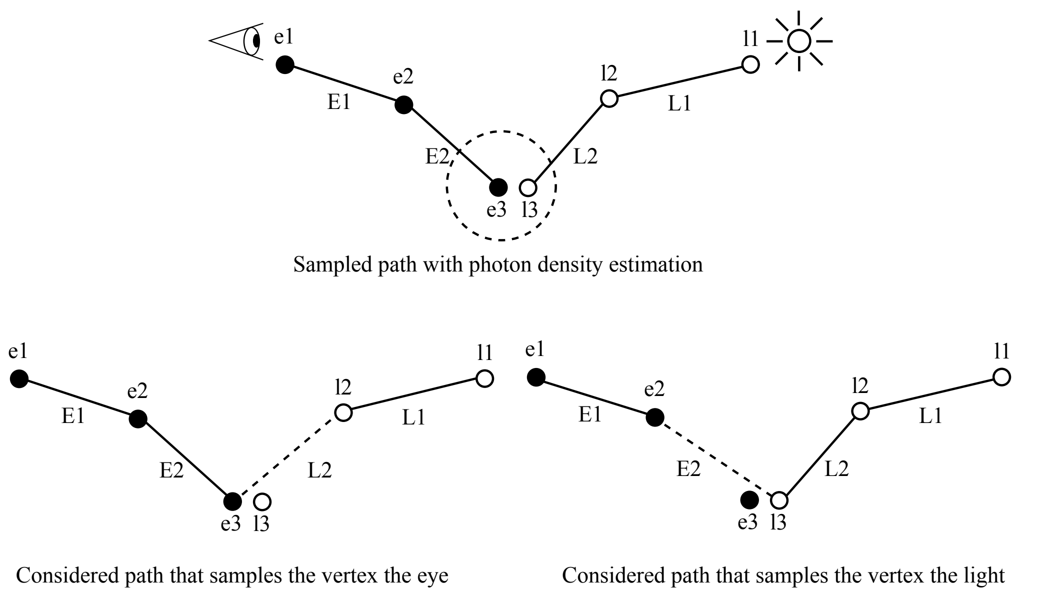

e1
e2
e3
l3
l2
l1
E1
E2
L2
L1
Sampled path with photon density estimation
Considered path that samples the vertex the eye
Considered path that samples the vertex the light

Figure 9.3: Selection of vertices under unbiased techniques for a path generated from photon density estimation. Given the sampled path on top, two different unbiased sampling techniques can sample the same path. The choice depends on how the vertex at the connection would be generated using the considered unbiased technique.

Similar to light tracing, photon density estimation is an efficient sampling technique when a light source has a directional peak. Handling directional distributions has significant practical importance since light sources in the real world, such as LEDs, often have such directional emission profiles. Including photon density estimation can be important even in a simple configuration like Figure 9.4.

9.4 Biased Multiple Importance Sampling

In order to fully utilize this new unified framework, we would like to have a provably optimal combination strategy such as the original balance heuristic in Equation 9.5. The original derivation of provably optimal weighting strategies in multiple importance sampling made two assumptions that are incompatible with the proposed framework.

First, the derivation considered all the techniques are unbiased. In other words, the derivation characterized error of each technique solely by variance. This assumption breaks in our method since progressive density estimation in progressive photon mapping is a biased technique. As we discussed in Chapter 6, error in progressive density

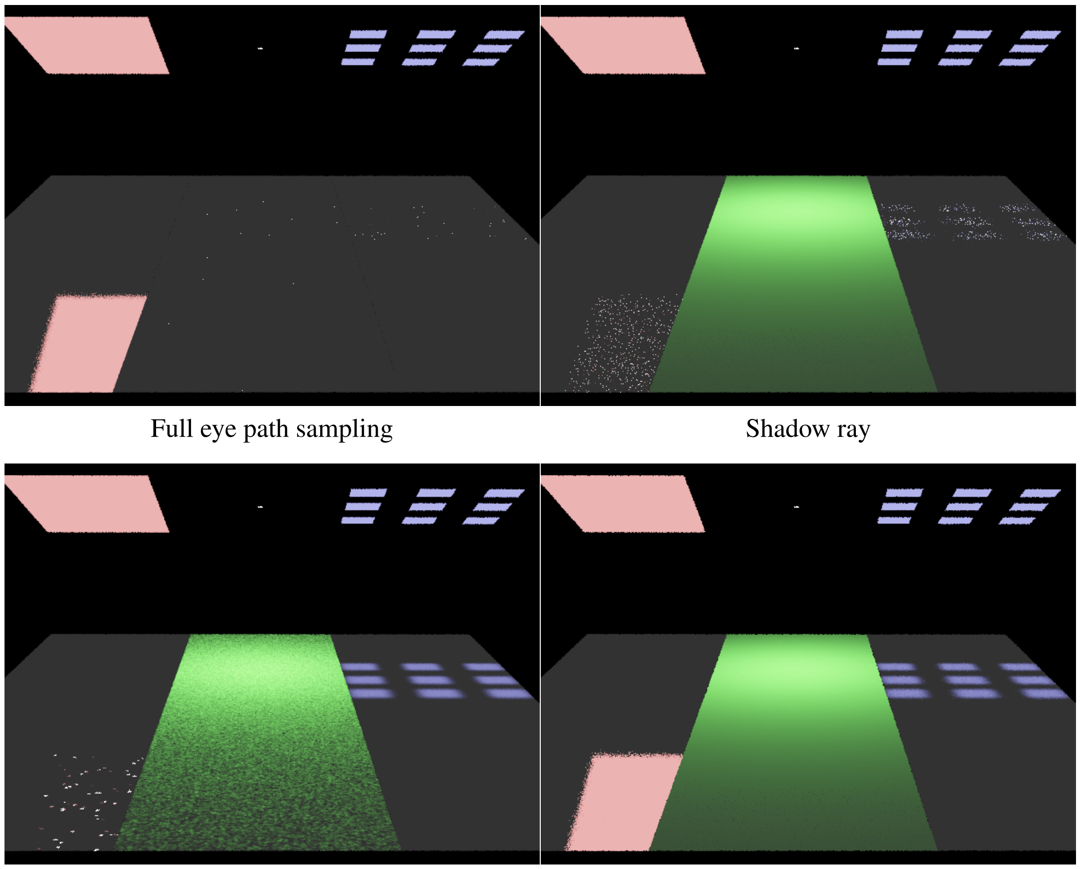


Figure 9.4: Efficiency of different techniques for direct illumination. This test scene features the combinations of a highly glossy material with a large diffuse light source (the left part within each image), a diffuse material with a small diffuse light source (the middle part), and a diffuse material with spotlights (the right part). All the images are rendered using four samples per pixel. Only our hybrid sampling works well across all the combinations. Note that we isolated influence of each light source to its corresponding surface in order to clarify the efficiency of each technique (e.g., green illumination does not appear on the right part).

estimation should take into account both variance and *bias*. Since a provably optimal combination should have error of the hybrid estimator within a constant from the truly optimal combination, effect of bias on such as strategy should be analyzed.

Second, the derivation assumed that all the techniques have the same convergence rate. This is a natural consequence based on the fact that unbiased Monte Carlo integration methods have $O(N^{-\frac{1}{2}})$ convergence rate regardless of probability density functions used. However, this assumption again breaks with progressive density estimation. Chapter 6 as well as the work by Knaus and Zwicker [70] show that progressive density estimation has asymptotic convergence rate of $O(N^{-\frac{1}{3}})$ at most, which is slower than that of unbiased Monte Carlo integration methods.

This section shows that bias indeed affects a provably optimal combination of multiple importance sampling and convergence rates of all the techniques should be balanced is necessary in order to retain the optimality of such a provably good combination. As a practical consequence of this derivation, we show that the alpha parameter of progressive photon mapping should be less than $\frac{2}{3}$ and the number of samples for unbiased techniques should be proportional to the number of *accumulated* photon paths in progressive photon mapping.

9.4.1 Problem Settings

In order to take bias into account in multiple importance sampling, we consider a biased method as an unbiased method for biased solutions. This makes it possible to characterize bias as a result of modifications to an original integrand. We denote such modifications by the ith sampling technique as $b_i(x)$. Following the same notation as Veach's [117], the jth sample from the ith technique has the following contribution:

$$F_{i,j} = \frac{w_i(X_{i,j})(f(X_{i,j}) + b_i(X_{i,j}))}{p_i(X_{i,j})}, \tag{9.9}$$

where $w_i(X_{i,j})$ is the weight function, $f(X_{i,j})$ is the integrand, $b_i(X_{i,j})$ is the modifications to the original integrand that introduces bias, and $p_i(X_{i,j})$ is the probability density function. The expected (and potentially biased) value from the ith technique is then given by

$$\mu_i = E[F_{i,j}] = \int_\Omega w_i(x)(f(x) + b_i(x))\mathrm{d}\mu(x). \tag{9.10}$$

Note that $b_i(X_{i,j})$ is not bias itself, but rather the contribution to the bias from the sample $X_{i,j}$. We can describe bias from the ith technique as the difference between the biased solution (i.e. μ_i) and the correct solution;

$$B[F_{i,j}] = E[F_{i,j}] - \int_\Omega w_i(x)f(x)\mathrm{d}\mu(x) = \int_\Omega w_i(x)b_i(x)\mathrm{d}\mu(x). \tag{9.11}$$

Both bias and expected values do not depend on the index of samples j.

Our goal is to find a weighting strategy which has expected error that is within a constant factor from the truly optimal weighting strategy. This is also what the original balance heuristic achieves. Our contribution is the derivation that shows necessary modifications to the original balance heuristic in order to keep this optimality in combination with a biased technique.

The goal of the derivation of the original balance heuristic is to minimize variance of the solution. This is because the error of unbiased Monte Carlo techniques is solely characterized by variance. However, in our setting, we also need to take bias into account in order to minimize error of the solution. We therefore need to minimize the squared error based on *bias-variance decomposition*:

$$E[(F-\hat{\mu})^2] = V[F] + B[F]^2, \tag{9.12}$$

where $\hat{\mu}$ is the correct solution, $V[F]$ is variance, and $B[F]$ is bias. Minimizing squared error including bias in general, however, is a very challenging task. This is because bias is a systematic error that happens because of various reasons, and it is often difficult to define general characteristics of bias in order to perform any theoretical analysis on quantities including bias. We therefore make a couple of assumptions that are often reasonable in rendering.

Firstly, we only consider the case where we have one nth biased technique in addition to other $n-1$ unbiased techniques. This can be true in our method if we restrict ourself to consider only one photon density estimation technique. Secondly, we assume that the contribution to bias from each sample is constant. This is also reasonable in a photon density estimator as we only consider neighboring photons which tend to cause similar error within each radiance estimate. We can thus set $b_i(x) = 0$ for $i \neq n$ and $b_n(x) = b_n$ for $i = n$. Notice that we are overloading the notation of b_n for readability.

We can then expand $E[(F-\hat{\mu})^2]$ as follows:

$$\begin{aligned}
E[(F-\hat{\mu})^2] &= V\left[\sum_{i=1}^{n}\frac{1}{n_i}\sum_{j=1}^{n_i}F_{i,j}\right] + B\left[\sum_{i=1}^{n}\frac{1}{n_i}\sum_{j=1}^{n_i}F_{i,j}\right]^2 \\
&= \sum_{i=1}^{n}\frac{1}{n_i^2}\sum_{j=1}^{n_i}V[F_{i,j}] + \left(\sum_{i=1}^{n}\frac{1}{n_i}\sum_{j=1}^{n_i}B[F_{i,j}]\right)^2 \\
&= \sum_{i=1}^{n}\frac{1}{n_i^2}\sum_{j=1}^{n_i}V[F_{i,j}] + \left(\sum_{i=1}^{n}B[F_{i,j}]\right)^2 \\
&= \left(\sum_{i=1}^{n}\frac{1}{n_i^2}\sum_{j=1}^{n_i}E[F_{i,j}^2] - \sum_{i=1}^{n}\frac{1}{n_i^2}\sum_{j=1}^{n_i}E[F_{i,j}]^2\right) \\
&+ \left(\int_\Omega\left(\sum_{i=1}^{n}w_i(x)b_i(x)\right)\mathrm{d}\mu(x)\right)^2 \\
&= \int_\Omega\left(\sum_{i=1}^{n-1}\frac{w_i^2(x)f^2(x)}{n_ip_i(x)} + \frac{w_n^2(x)(f(x)+b_n)^2}{n_np_n(x)}\right)\mathrm{d}\mu(x) \\
&+ \left(\int_\Omega w_n(x)b_n\mathrm{d}\mu(x)\right)^2 - \sum_{i=1}^{n}\frac{1}{n_i}\mu_i^2,
\end{aligned} \tag{9.13}$$

In the following derivations, we will show how to minimize the sum of the first two terms. The last term $\sum_{i=1}^{n}\frac{1}{n_i}\mu_i^2$ has the same bound as the original derivation by Veach thus the last term is independent from weighting functions.

9.4.2 Minimizing Squared Error

Even though we made some assumptions, Equation 9.13 is still difficult to minimize with respect to w_i, since bias introduced the integral term $(\int_\Omega w_n(x)b_n\mathrm{d}\mu(x))^2$. In order to yield the optimal w_i including this term, it seems that we need to solve an integral equation which is often intractable. However, we can show that minimizing a point-wise expression,

$$\sum_{i=1}^{n-1}\frac{w_i^2(x)}{n_ip_i(x)} + \frac{w_n^2(x)(1+r_n(x))^2}{n_np_n(x)} + Ar_n^2(x)w_n^2(x) \tag{9.14}$$

indeed suffices to minimize the full expression of $D[F]$ in Equation 9.13, where we defined $r_n(x) = \frac{b_n}{f(x)}$ and $A = \int_\Omega \mathrm{d}\mu(x)$ for readability.

We first start by trying to find an alternative expression for $(\int_\Omega w_n(x) b_n \mathrm{d}\mu(x))^2$. In order to obtain such an expression, we use the Cauchy-Schwarz inequality and consider the bound of this term;

$$\left(\int_\Omega w_n(x) b_n \mathrm{d}\mu(x)\right)^2 \leq A \int_\Omega w_n^2(x) b_n^2 \mathrm{d}\mu(x). \tag{9.15}$$

Minimizing the bound in general does not minimize the original term since the bound might not have the same extrema as the original function. In our case, however, the bound and the function happen to have extrema at exactly the same points since

$$\frac{\partial}{\partial w_n}\left(\int_\Omega w_n(x) b_n \mathrm{d}\mu(x)\right)^2 = A \frac{\partial}{\partial w_n} \int_\Omega w_n^2(x) b_n^2 \mathrm{d}\mu(x). \tag{9.16}$$

Note that this is possible because of our assumptions on bias. Since the latter is the upper bound of the function, minimizing the bound also minimizes the function given the fact that they have extrema at the same points.

Furthermore, since the sums of two functions $f(x)+g(x)$ and $f(x)+h(x)$ have the same extrema if $\frac{dg}{dx} = \frac{dh}{dx}$, minimizing

$$\int_\Omega \left(\sum_{i=1}^{n-1} \frac{w_i^2(x) f^2(x)}{n_i p_i(x)} + \frac{w_n^2(x)(f(x)+b_n)^2}{n_n p_n(x)} + A w_n^2(x) b_n^2 \right) \mathrm{d}\mu(x) \tag{17}$$

is equivalent to minimizing the corresponding sums in Equation 9.13. We can yield Equation 9.14 by dividing the integrand of this equation by $f^2(x)$.

9.4.3 Biased Balance Heuristic

Using the method of Lagrange multipliers, the minimum value of Equation 9.14 is attained when all $n+1$ partial derivatives (n derivatives for w_i and one for λ) of the expression

$$\sum_{i=1}^{n-1} \frac{w_i^2}{n_i p_i} + \frac{w_n^2(1+r_n)^2}{n_n p_n} + A r_n^2 w_n^2 + \lambda \left(\sum_{i=1}^{n} w_i - 1 \right) \tag{9.18}$$

are zero. Note that we dropped the notation (x) similar to the Veach's derivation since this is a point-wise minimization of the function. The solution to this equation yields

$$\hat{w}_i(\mathbf{x}) = \frac{n_i p_i'(\mathbf{x})}{\sum_{k=1}^{n} n_k p_k'(\mathbf{x})}, \tag{9.19}$$

where

$$p'(\mathbf{x})_i = \begin{cases} p_i(\mathbf{x}) & (i \neq n) \\ p_n(\mathbf{x}) \frac{1}{(1+r_n)^2 + n_n p_n(\mathbf{x}) A r_n^2} & (i = n), \end{cases} \tag{9.20}$$

A is a constant, and r_n is the relative magnitude of the contribution of the bias to the sampled value. Here the nth technique is biased. Note that if there is no bias $r_n = 0$, we obtain $p'_i = p_i$ and Equation 9.19 turns into the original balance heuristic in Equation 9.5. Note also that taking an infinite number of samples $\lim n_n \to \infty$ turns the weight for a biased method into zero, which makes the combined estimate converge to the correct solution even if $r_n \neq 0$.

9.4.4 Balancing Asymptotic Convergence Rates

When each technique has a different convergence rate, a solution with multiple importance sampling will be asymptotically dominated by the technique with the slowest convergence rate that might have the highest variance. For example, consider a part of a scene where the estimator A has higher variance, but the other estimator B has lower variance. Suppose also that the estimator B has slower convergence rate than the estimator A. The balance heuristic would put higher weight for the estimator B since it has lower variance. However, this combination asymptotically results in arbitrary larger error than other combination due to slower convergence of the estimator B. For instance, just using the estimator A can be arbitrary better even with finite time, which invalidates the optimality.

Recall that the asymptotic convergence rate of the squared error in progressive photon mapping is recently derived as [70];

$$O\left(\frac{1}{N^{\alpha}}\right) + O\left(\frac{1}{N^{2(1-\alpha)}}\right). \tag{9.21}$$

In order to balance the convergence rate, we first introduce a function $N'(k)$ that returns the number of samples for an unbiased technique for given k samples for progressive photon mapping. Since unbiased estimators have $O\left(\frac{1}{N'(N)}\right)$ as convergence rate in this case, the remaining task is to find a function $N'(k)$ such that

$$O\left(\frac{1}{N^{\alpha}}\right) + O\left(\frac{1}{N^{2(1-\alpha)}}\right) = O\left(\frac{1}{N'(N)}\right) \tag{9.22}$$

is satisfied. This equation has the following two solutions. For $\alpha < \frac{2}{3}$,

$$\begin{aligned} O\left(\frac{1}{N^{\alpha}}\right) + O\left(\frac{1}{N^{2(1-\alpha)}}\right) &= O\left(\frac{1}{N'(N)}\right) \\ O\left(\frac{1}{N^{\alpha}}\right) &= O\left(\frac{1}{N'(N)}\right) \\ N'(N) &= N^{\alpha}, \end{aligned} \tag{9.23}$$

and for $\alpha \geq \frac{2}{3}$,

$$\begin{aligned} O\left(\frac{1}{N^{\alpha}}\right) + O\left(\frac{1}{N^{2(1-\alpha)}}\right) &= O\left(\frac{1}{N'(N)}\right) \\ O\left(\frac{1}{N^{2(1-\alpha)}}\right) &= O\left(\frac{1}{N'(N)}\right) \\ N'(N) &= N^{2(1-\alpha)}. \end{aligned} \tag{9.24}$$

9.4.5 Practical Consequences

Equation 9.19 reveals some critical issues. In order to use this provably good weighting strategy of unbiased methods and a biased method, one would have to evaluate the magnitude of bias relative to the sampled value, r_i. Computing the exact r_i is probably not realistic in general, since if we knew the amount of bias, we could just subtract it out to get an unbiased estimator and use the original balance heuristic. Moreover, even if we had a method to estimate r_i such as the one in Chapter 6, the provably good weighting strategy would require the additional ability of estimating r_i of samples which were *n*ot even sampled by a biased technique in the first place.

We can however utilize the results of this derivation as a guidance to design a practical solution. Note that Equation 9.19 shows that the only modification should be made is the multiplication factor of the probability density of the biased technique p_n. This multiplication factor converges to one for $\alpha < \frac{2}{3}$ of progressive photon mapping since

$$\frac{1}{(1+r_n)^2 + n_n p_n A r_n^2} \in \frac{1}{1+O(N^{\alpha-1})+O(N^{3\alpha-2})} = \frac{1}{1+O(N^{\alpha-1})} \left(\text{for } \alpha < \frac{2}{3}\right). \tag{9.25}$$

Therefore, for $\alpha < \frac{2}{3}$, the biased balance heuristic is asymptotically getting close to the original balance heuristic as we take more samples. In other words, under this condition, the effect of bias in a provably optimal combination is getting smaller and smaller as we take more samples.

Considering the fact using bias in the biased balance heuristic is difficult, we opt to ignore bias in our practical implementation with the condition $\alpha < \frac{2}{3}$. Note that while ignoring bias is a practical choice that we made, the condition $\alpha < \frac{2}{3}$ is a theoretical results derived by taking the effect of bias into account in a provably good combination.

Under the condition that $\alpha < \frac{2}{3}$, Equation 9.23 revealed that we should use $N'(N) = N^{\alpha}$ samples for unbiased techniques for given N samples in progressive photon mapping. This is necessary to keep the combined estimator provably better than other combinations.

Fortunately, we can achieve this condition easily by realizing that the local photon count has $O(N^{\alpha})$. We therefore just need to set the number of samples for unbiased Monte Carlo methods proportional to the local photon count of progressive photon mapping. The excepted squared error of the final combined estimator has the convergence rate of $O(1/N^{\alpha})$.

9.5 Implementation

Given probability densities of paths, an implementation of multisampled progressive photon mapping is relatively straightforward. Our implementation is based on stochastic progressive photon mapping, where we store photon statistics for progressive photon mapping per pixel. One difference is that each pixel stores multiple measurement locations that we generate by tracing a whole eye sub-path. Note that we perform photon density estimation at all the eye vertices, not only at the first eye vertex (which corresponds to the original progressive photon mapping) or the second eye vertex as in Figure 9.1. Each pixel thus stores information of an entire eye path, which is used for scaling photon flux according to multiple importance sampling. The update procedure of photon statistics stays the same otherwise. We summarize the steps of the algorithm as follows:

1. **Generate a photon map with information of light paths.**
   Each photon stores a pointer to the photon at the previous bounce. For example, a photon stored at the third bounce has a pointer to the photon at the second bounce in the same path.

2. **Generate measurement points at each pixel by tracing eye rays with information of eye paths.**
   One difference from (S)PPM is that we trace eye rays until the path terminates according to Russian Roulette. Another difference is that we generate a measurement point at each intersection point, not just at the first/second point. We also store pointers to measurement points at previous bounces similar to Step 1.

3. **Gather neighboring photons using the radius associated to the pixel.**

4. **Weight contribution of each photon according to the balance heuristic and update the photon statistics.**
   Using the path information from Step 1 and Step 2, we compute probability densities of the whole path under all the techniques we consider (Section 3). These probability densities are used to compute the weight according to Equation 8 by ignoring bias (which is equal to Equation 7). The photon statistics are updated as in SPPM.

5. **Perform unbiased Monte Carlo ray tracing at the same pixel.**
   We perform path tracing at the same pixel we considered in Step 2-4, but weight its contribution using the balance heuristic. Again, we need to compute the probability densities of this path generated by path tracing under all the techniques including (S)PPM. The number of samples we take is equal to the number of photons we found in Step 3.

6. **Repeat Step 2-5 for all the pixels.**

7. **Iterate the whole steps.**

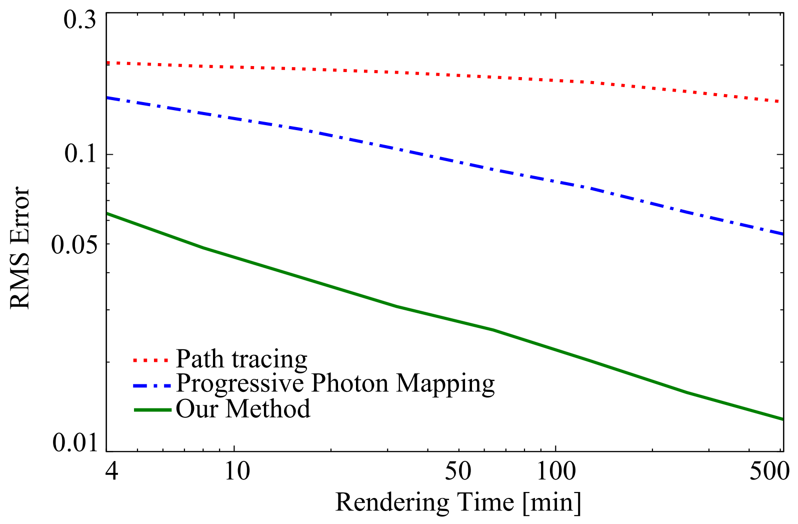


Figure 9.5: Graph that shows RMS errors of different methods using the same rendering time for the test scene in Figure 9.6. Our method not only gives us visually pleasing images, but also provides more accurate solutions for the same rendering time.

In the implementation, we used a separate image buffer to accumulate the contributions from the unbiased sampling techniques. For each new eye path in the eye pass in stochastic progressive photon mapping, instead of just computing the measurement location, we also add contribution from this eye path into the separate image buffer. Even though our formulation can incorporate full bidirectional path tracing, we ignore shadow ray connections with the light sub-path with more than one vertex in our implementation since the number of visibility tests would be significantly large in our setting. This separate image buffer therefore essentially contains a weighted solution of path tracing with next event estimation. The final image is simply a sum of this buffer and the weighted solution of stochastic progressive photon mapping.

Recall also that the previous section derived two conditions that should be met in the combined estimator. First, progressive photon mapping should use $\alpha < \frac{2}{3}$. Second, unbiased techniques should use $N'(N) = N^{\alpha}$ for given N samples in progressive photon mapping. The consequence to the final algorithm is very simple. We just use $\alpha = 0.6 < \frac{2}{3}$, and we perform the eye pass when the counter of newly accumulated photon paths becomes more than the number of pixels and reset the counter afterward.

Table 9.2: Statistics of our experiments. The numbers in the column of each method show the average numbers of samples per pixel and RMS errors (Path; path tracing, PPM: progressive photon mapping, and MPPM: multisampled progressive photon mapping). Note that the average number of samples for PPM and MPPM counts samples that did not contribute to the final image (e.g., photon path that missed the scene from the beginning), thus PPM could take more samples than path tracing for some scenes.

Scene	Path	PPM	MPPM	Time [min]
Buddha	365 (0.1900)	270 (0.1067)	151 (0.0399)	30
Conference	236 (0.0164)	N/A	119 (0.0270)	60
Sibenik	333 (0.3233)	N/A	133 (0.0549)	60
Torus	N/A	242 (0.0365)	138 (0.0235)	15
Treasures	1069 (0.0432)	1222 (0.1217)	720 (0.0131)	120
Wheels	682 (0.0417)	690 (0.0512)	369 (0.0156)	120

9.6 Results

We have implemented path tracing with next event estimations [66], progressive photon mapping, and our method using the same ray tracing core. Note that some of our test scenes feature specular-diffuse-specular paths which can only be rendered by progressive photon mapping at the moment. Note also that the goal is to develop a *single* algorithm that handles all the practical scenes equally well. Existing unbiased methods do not satisfy this goal to begin with as we already demonstrated in previous chapters. In the results, we use comparisons with path tracing mainly for demonstrating the effect that will be efficiently captured by an unbiased Monte Carlo ray tracing method.

We ran all the experiments on an Intel Core 2 Q6600 at 2.4GHz using a single core and the resolution of the images are 512×512 and 640×480. Table 9.2 summarizes various statistics of our experiments. We should note that optimizing absolute rendering time is not our main purpose, and it is likely that a more careful implementation would significantly reduce the overall rendering times. All the comparisons are equal-time.

Figure 9.6 highlights the advantage of our method. This scene contains diffuse illumination, highly glossy reflections, caustics and their highly glossy reflections to the viewer, and a light source with a sharp directional distribution. Path tracing is efficient for computing some contributions from direct illumination and glossy reflections, yet caustics cause noise. Progressive photon mapping handles caustics, reflections of caustics, and the highly directional light source robustly, but a sharp BRDF lobe of the highly glossy material becomes a source of noise. Our method adaptively combines both methods without any user intervention and produces a less noisy solution in the same rendering time. The graph in Figure 9.5 shows the convergence of the RMS (Root Mean Square) errors of the same scene with different methods. Figure 9.9 demonstrates the results of our method for more complex scenes.

Stochastic progressive photon mapping already demonstrated efficient rendering of glossy reflections by tracing one bounce rays from a visible point through each pixel. The issue however is that whether we trace such rays or not is based on a heuristic classification of diffuse/non-diffuse materials. Multisampled progressive photon mapping takes a different approach by always tracing such rays and weighting the result according to multiple importance sampling. This approach eliminates any such heuristic classification of the input scene. Since our method subsumes stochastic progressive photon mapping as shown in Figure 9.1, we can also consider our method as a generalization of stochastic progressive photon mapping (Figure 9.10).

As we mentioned in Section 9.3, photon density estimation is an efficient method for handling a light source with a sharp directional distribution. Figure 9.11 highlights this property in a realistic scene setting. This test scene uses Lambertian surfaces with light sources with a directional radiance distribution according to modified-Phong with the exponent 1000 to simulate spotlights. Although the scene setup is seemingly simple, path tracing becomes inefficient in this scene. This is because path tracing does not consider the directional radiance distribution of the light sources. Our method adaptively employs photon density estimation in such cases by putting greater weight to photon density estimation using multiple importance sampling.

Path tracing however provides a benefit if a scene consists of diffuse materials and non-directional light sources. Our method is indeed almost as efficient as path tracing for a such scene as shown in Figure 9.7. This property is useful since (stochastic) progressive photon mapping often produces much noisier results for diffuse illumination than path tracing (Figure 9.4). When an input scene contains paths that are difficult to capture with path tracing alone, our method again adaptively combines progressive photon mapping to robustly handle such paths while keeping the benefit of path tracing (Figure 9.8).

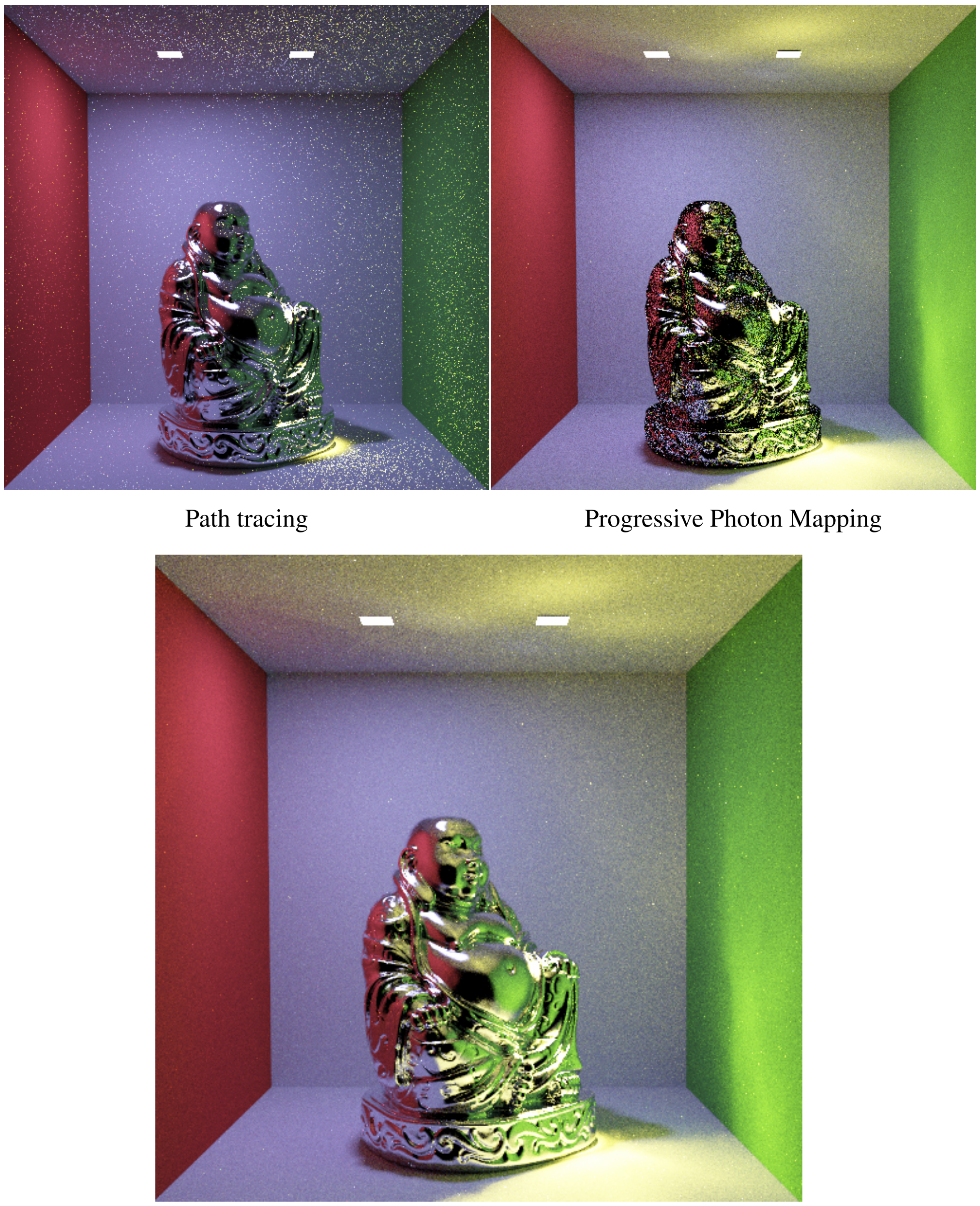

Path tracing
Progressive Photon Mapping
Multisampled Progressive Photon Mapping

Figure 9.6: Buddha statue with highly glossy metal illuminated by a blue diffuse light source and a yellow spotlight. Path tracing is efficient for computing direct illumination from the diffuse light source and glossy reflections, while being inefficient for caustics and indirect illumination from the spotlight. Progressive photon mapping robustly handles such caustics and indirect illumination, however, glossy reflections can cause noise. Multisampled progressive photon mapping automatically combines path tracing and progressive photon mapping using multiple importance sampling and produces a significantly less noisy solution in the same rendering time.

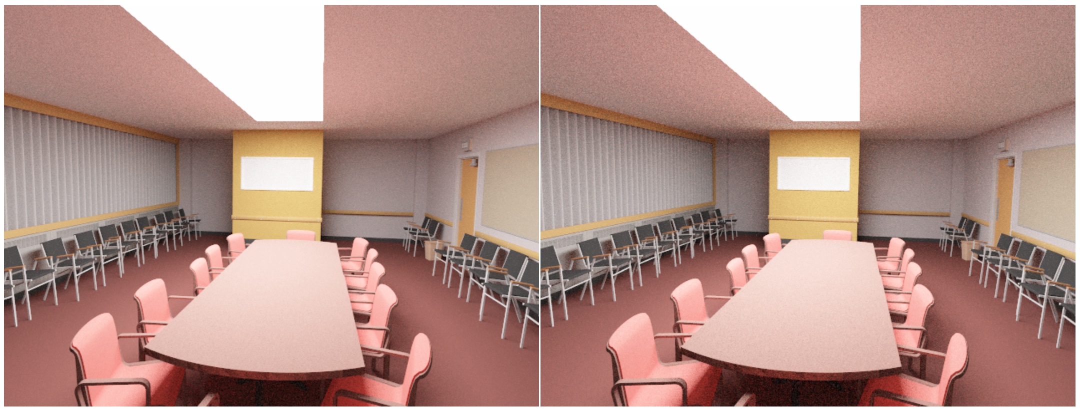


Figure 9.7: Conference room with diffuse surfaces and a large diffuse light sources. For this type of scenes, our method performs almost as well as path tracing since the computed component of path tracing automatically dominates the final image.

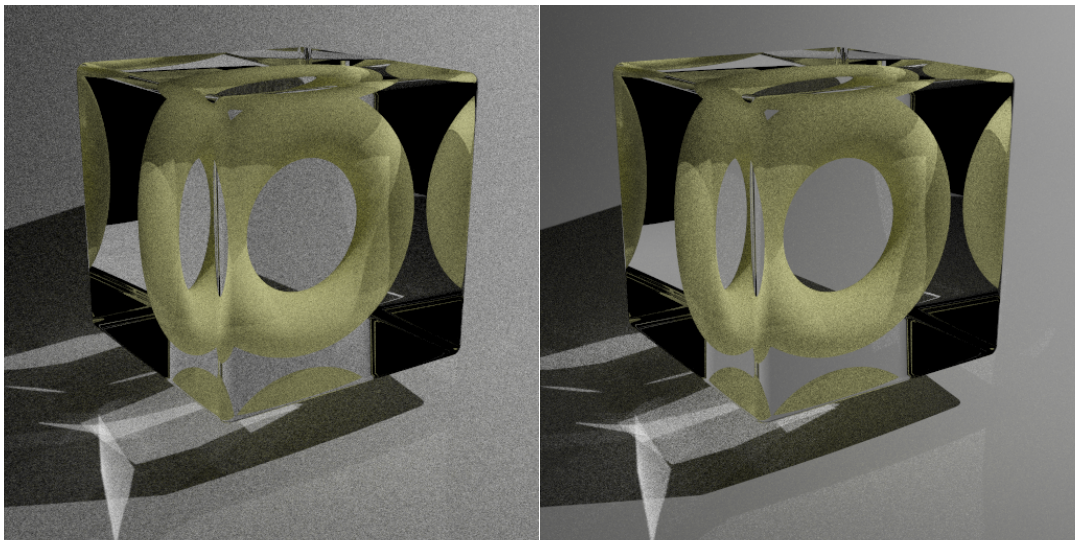


Figure 9.8: Torus scene. Both methods successfully captured all light paths including specular-diffuse-specular paths on the torus, however, the image with stochastic progressive photon mapping contains undesirable noise in the directly illuminated region. Our method significantly reduces noise in such region by adaptively combining the direct illumination solution using shadow rays.

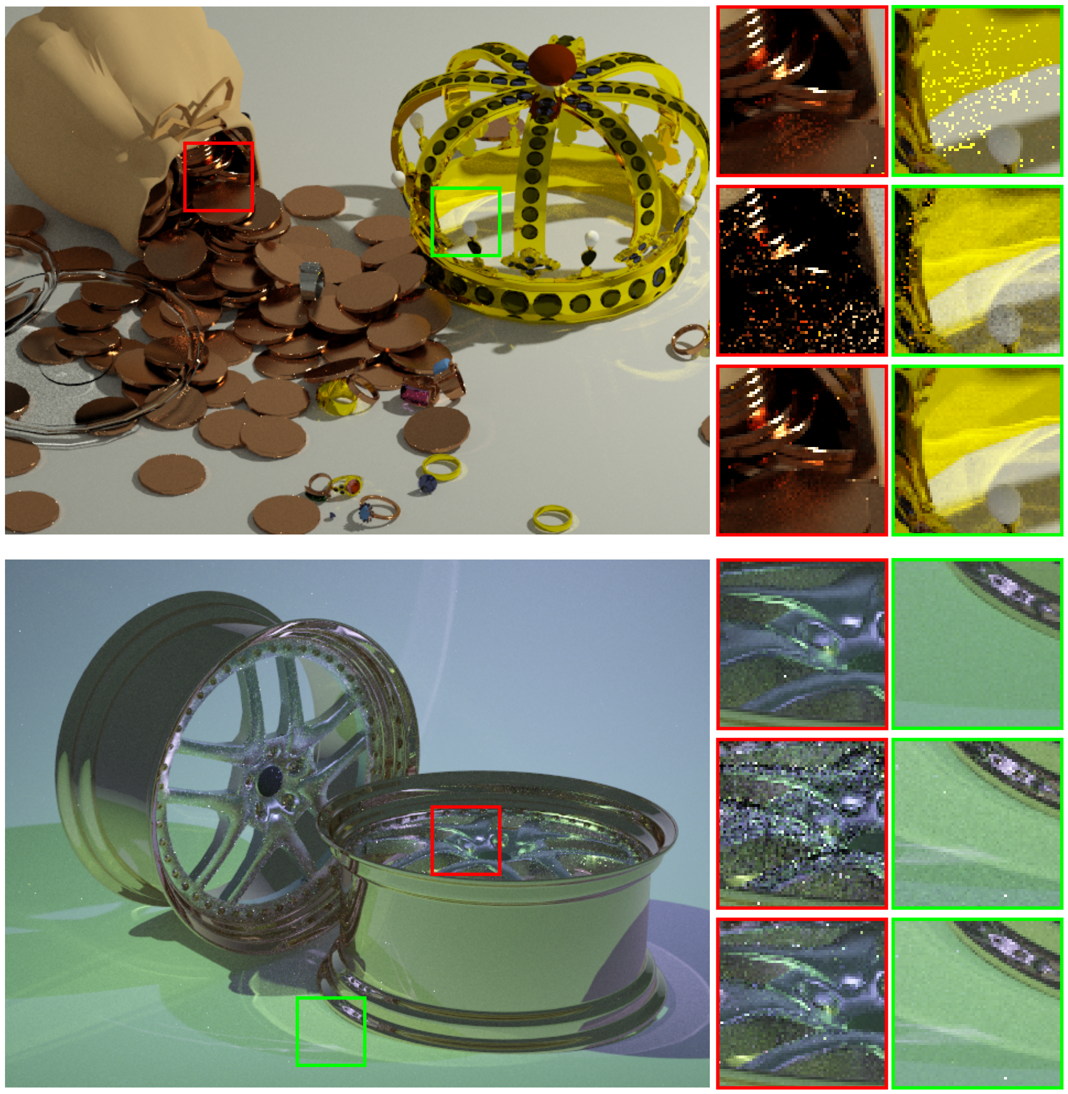


Figure 9.9: Scenes features high geometric complexity and illumination complexity. Treasures scene has glass plates and coins and a crown with glossy metal illuminated by a small diffuse light source. Wheels scene has two wheels with glossy metal illuminated by two spotlights. This figure shows the images rendered by path tracing with the next event estimation (top), progressive photon mapping (middle), and multisampled progressive photon mapping (bottom) using the same rendering time. Path tracing failed to capture caustics while rendering accurate glossy reflections. Progressive photon mapping captures caustics, but produces noisy results for glossy reflections. Multisampled progressive photon mapping takes the best of both approaches and captures the all illumination features efficiently.

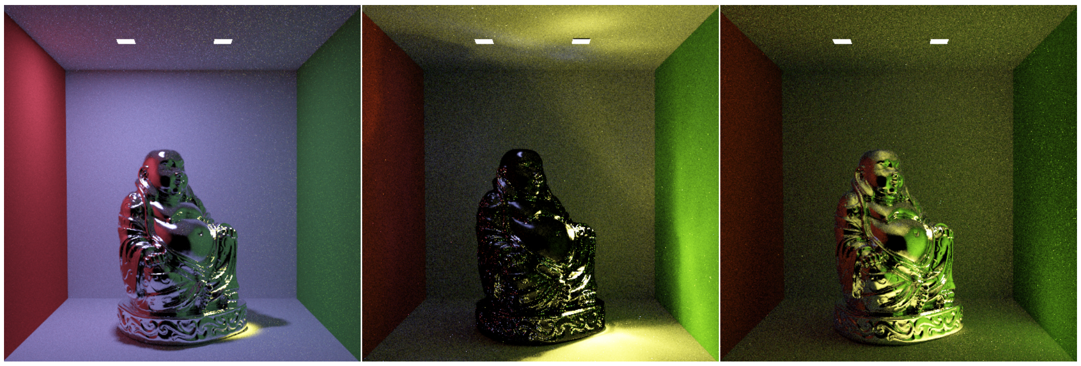


Figure 9.10: Weighted components of multisampled progressive photon mapping. The images show computed illumination due to path tracing (left), progressive photon mapping (center), and the mixture of both methods (right) in the result of our method in Figure 9.6. Each method covers a different component of illumination based on the probability density functions of light paths. Note that mixture sampling on the right combines eye sub-paths with length equal to or longer than two and arbitrary light sub-paths. This mixture sampling includes stochastic progressive photon mapping as a special case of using eye sub-paths of length two.

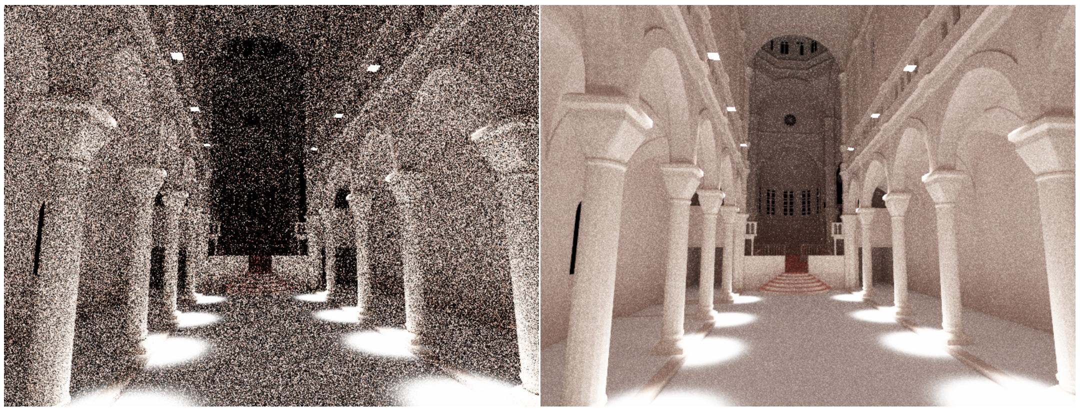


Figure 9.11: Sibenik cathedral with spotlights. The spotlights are the only light sources in this scene. The sharp directional distribution of the spotlights makes rendering with path tracing inefficient as the sampling method does not take this directional distribution into account. Our method adaptively incorporates photon density estimation, which performs importance sampling the directional profile of light sources, and produced a far less noisy result in the same rendering time.

9.7 Discussion

Even though multisampled progressive photon mapping works well across many scene configurations, there are a couple of limitations in its current form. Firstly, while we opted to ignore bias in our method, our derivation shows that bias surely affects a provably good weighting strategy in multiple importance sampling. We found our decision to be reasonable in practice as we have demonstrated, but further investigation might lead to a better alternative.

Secondly, even though our derivation provides a first step in theoretical analysis of the effect of bias in multiple importance sampling, our derivation also makes two assumptions that might be too restrictive in some cases. The first assumption is that we only have one biased Monte Carlo method in a combined solution, and the second assumption is that the contribution to bias from each biased sample is constant. We made these assumptions in order make theoretical analysis of the weighting function tractable, but further research may be able to relax some parts of the assumptions by using our work as a stepping-stone.

Lastly, in our implementation, we have not done full connections of all light vertices and eye vertices using shadow rays as we described in Section 9.5. Such connections could be beneficial for some scenes with strong diffuse indirect illumination. The challenge however is that we need to consider an extremely large number of shadow ray connections since we can potentially have many light paths and eye paths in stochastic progressive photon mapping. Recent work on related problems such as multidimensional lightcuts [122] might be applicable to reduce the necessary number of shadow ray tracings, however, direct applications would not work since none of such existing methods is consistent with a limited number of light paths and eye paths per iteration. Veach [117] described an application of Russian Roulette to reduce the number of shadow rays, which could be applied to our method as well.

A few studies explored applications of multiple importance sampling to regular (not progressive) photon mapping [62]. Bakaert et al. proposed a combination of the regular photon density estimation using multiple importance sampling in the context of their modified photon density estimator [9]. They mentioned that their framework does not handle caustics from specular materials which our method can efficiently handle.

Vorba and Křivánek [120] described how multiple importance sampling can be used in final gathering of regular photon mapping. Contrarily to their approach, our method does not limit density estimation to final gathering paths. Moreover, none of those studies analyzed the effect of bias in multiple importance sampling, which is one of our main theoretical contributions. Our method is also the first to combine progressive photon mapping using multiple importance sampling.

9.8 Conclusion

We presented a new light transport algorithm, multisampled progressive photon mapping, that provides a unified view of progressive photon mapping and unbiased Monte Carlo ray tracing methods based on multiple importance sampling. We provided theoretical analysis on how bias from progressive photon mapping affects a provably good combination of multiple sampling techniques. We described how to evaluate the probability density function of paths in photon density estimation and also delivered the necessary conditions for the practical algorithm based on the analysis. We have demonstrated the improved robustness and efficiency of our method in comparison to the original progressive photon mapping and path tracing. We believe that our framework will find many practical applications for photorealistic image synthesis and also lead to further development of robust light transport simulation methods which can efficiently handle all kinds of illumination. As such, our contribution could be summarized as providing a new basis for more robust future physically-based rendering methods.

9.9 Acknowledgements

The material in this chapter is, in part, a reproduction of the material in Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen, "Multisampled Progressive Photon Mapping" [46]. The dissertation author was the primary investigator and author of this paper. The Buddha statue model is courtesy of Jotero.com, the wheel model is available on turbosquid.com, and the Sibenik model is courtesy of Marko Dabrovic. The dissertation author was supported in part by the 2010 NVIDIA Fellowship Program.

Chapter 10

Stochastic Spatial Hashing

One costly operation in the progressive photon mapping algorithm is a spatial range query of neighboring photons. It is not only that the queries themselves are costly, but also the construction of an acceleration data structure for queries adds non-negligible overhead in progressive photon mapping as the algorithm repeatedly constructs such an data structure many times. In this chapter, we introduce a new data-parallel acceleration data structure for range queries, *stochastic spatial hashing*, which is suitable for processing on a massively parallel processor like Graphics Processing Units (GPUs). Stochastic spatial hashing stochastically selects a single element (photon or measurement point) to be kept, instead of storing a list of elements at each hash entry as in regular spatial hashing. This simple modification makes the construction process data-parallel, and guarantees uniform work distribution of retrieval of photons regardless of the number of collisions, which is preferable for massively parallel processing. The stochastic selection adds no extra bias in results, thus maintains the correctness of progressive photon mapping. Using this new hashing scheme, we demonstrate the implementation of progressive photon mapping on GPUs.

10.1 Related Work

Several previous work attempted to implement range queries on a GPU as a parallel computation platform in the context of photon density estimation. The first implementation was proposed by Purcell et al. [95]. They presented two algorithms that construct a uniform grid data structure for photon maps: bitonic sort and stencil routing. In terms of the algorithmic representation, our method is akin to the stencil routing with a single photon per grid cell. However, stencil routing adds bias due to flux redistribution when hash collision occurs. Our method in contrast adds no extra bias and the theory behind our method is not based on flux redistribution. We also employ a spatial hashing which reduces the number of unused grid cells that needs to be retained on graphics hardware.

Another implementation on GPUs is done by Fabianowski et al [24], which uses a linear BVH for a data structure of photon map [74]. A kd-tree has also been used for a GPU photon mapping [125, 136]. Since those algorithms use tree data structures, the retrieval requires O(logN) (for *N* photons) global data fetches and the work distribution is typically nonuniform. In contrast, our algorithm guarantees a constant and uniform number of global data fetches at the retrieval. Moreover, existing tree construction algorithms have a complex mapping to parallel processing due to dependency between parent nodes and child nodes. Our construction algorithm naturally maps to data-parallel processing on GPUs than tree construction algorithms, and results in significantly faster construction time than those previous work using a tree data structure.

Photon splatting [75, 84] is another algorithm that utilizes graphics hardware to perform photon mapping. Instead of constructing a photon map and gather photons in a separate pass, each photon is directly accumulated as a splat onto the image plane, which can be done by rasterizing photons as a disc (or some proxy geometry [84]). Although photon splatting completely eliminates the construction of a photon map, thus potentially being more efficient, it cannot handle specular reflections or refractions of eye rays (e.g., LD+S+E) due to the limitation of perspective projection in splatting. Our algorithm constructs a photon map and does not rely on perspective projection, thus the algorithm works in general settings.

Ma and McCool [82] proposed a hash-based kNN query algorithm for a parallel platform. The idea is to use a space filling curve to organize a localized set of points such that the neighboring indices of the hashed-index of a query point approximately point to spatially neighboring sets of photons. The hash function approach we are taking in this chapter is motivated by their work. The difference is that our algorithm does not need to construct a list of points, which is an inefficient operation on a massively parallel processor.

Alcantara et al. [4] presented a parallel hashing algorithm on GPUs. Their algorithm first roughly distributes data into each thread using a simple hash function, and performs Cuckoo hashing within each threads in parallel. Cuckoo hashing constructs a perfect hash table which achieves $O(1)$ retrieval. Due to the iterative nature of Cuckoo hashing, the algorithm can fail to terminate, thus it repeats the construction until the construction succeeds within a finite number of iterations. Although our algorithm does not build a perfect hash table, it achieves $O(1)$ retrieval time as well and does not need multiple tries. Our algorithm also has finer granularity of parallelism compared to their algorithm (single item vs 512 items per thread).

10.2 Method

Since progressive photon mapping constructs multiple photon maps to render a single image, we need a fast algorithm to construct a photon map on GPUs in order to implement it on GPUs. Since the original photon mapping uses a kD-tree, it is conceivable to use previous work that build a kD-tree or BVH [136, 24] on GPUs. However, we claim that a tree is not necessarily ideal data structure on GPUs.

First, construction of a tree is an inherently serial process due to dependencies between parent nodes and child nodes. Since the structure of child nodes depends on the structure of parent nodes, each level of the tree should be processed sequentially. Second, range query using a tree structure involves O(logN) data fetches for N photons. Since data fetches are relatively costly compared to mathematical operations on GPUs, we would like to reduce the number of data fetches. Finally, the work distribution of tree traversal is usually not uniform, which results in poor utilization of GPU units [3].

We instead use spatial hashing for data structure for range queries. Spatial hashing is a promising alternative since the construction has less data dependency and the number of data fetches is $O(1)$. At the same time, standard spatial hashing poses some issues that are not desirable for GPUs. In a typical CPU implementation of spatial hashing, each hash entry keeps a list of elements when collision happens. Constructing a list is not particularly efficient on GPUs because chaining of elements is a dependent, serial process. Furthermore, since the number of elements per hash entry typically varies, the amount of work at retrieval is still not uniform (the same observations were made by Alcantara et al [4]).

10.2.1 Stochastic Selection

Our simple solution to these issues is just keeping a single photon per hash entry by selecting the photon in a stochastic way. To be concrete, given n photons that would be mapped to the same hash entry, we pick one photon according to the uniform probability $p(x) = \frac{1}{n}$. In order to keep the results consistent, stored photon flux is now divided by the probability $p(x) = \frac{1}{n}$ (i.e., flux is multiplied by n).

This modification adds no extra bias when we use the resulting hash table to perform progressive radiance estimation. This can be trivially confirmed by that the expected radiance after this modification stays the same. This is the key difference from the redistribution of photon flux in the stencil routing [95], which adds extra bias. We do not redistribute flux, but simply select one photon and scale it according to the selection probability.

Since each hash entry now keeps only one photon, we no longer need to perform chaining of photons to a list. It therefore allows a data-parallel construction of a hash table. Moreover, the amount of work at retrieval is exactly one data fetch regardless of the number of collisions, which results in completely uniform work distribution at data retrieval.

10.2.2 Range Query

Since progressive radiance estimate needs to find photons within some distance to each intersection point, just performing one data retrieval is not enough. In other words, we need to perform range query of photons. Our approach is to consider range query as a sphere-points intersection, which has been well-explored in collision detection [114].

We use a sphere defined by each intersection point and points defined by photon positions. We then simply take the grid cells that overlap with the sphere and look up the hash table [114]. This operation is still done in $O(1)$ in contrast to $O(\log N)$ of previous work using a tree data structure [74, 136]. In order to accelerate range queries, we determine the grid cell size to two times the maximum radius. Range query thus only needs to look up neighboring cells. Since the radius reduction is monotonic (i.e., radius never increases), the number of data fetches in rage query is always 8 regardless of the number of passes. It is also independent from distribution of photons because of the stochastic selection in Section 10.2.1, thus results in uniform work distribution as well.

10.2.3 Eye Ray and Photon Tracing

In addition to progressive radiance estimation, we implemented photon tracing and eye ray tracing using a threaded BVH [115]. We used the combination of ML-CGs [76] as a pseudo random number generator on GPUs. We trace eye rays using distributed ray tracing as in stochastic progressive photon mapping, thus each pixel maintains photon statistics.

A standard CPU implementation of photon tracing keeps a global list of photons and sequentially adds a photon to the list. We can use a similar approach on GPUs by keeping a list of photons per thread. However this approach needs inter-threads commutation to combine lists in order to obtain a complete list of photons.

Our alternative approach traces a single bounce per pass thus stores a single photon per thread. For example, the very first pass traces photons from light sources toward the first intersections and stores the photons. Further bounces are traced in succeeding passes. Each path thus outputs either zero or one photon per thread, not a list of photons, thus simple masking at the hash construction is sufficient.

A new photon path is generated at next pass if the current photon path is killed by Russian Roulette or misses a scene. This process is done independently between threads, thus each thread potentially traces a photon ray at a different number of bounces. This approach keeps all threads busy all the time regardless of path length. A similar implementation is recently used for path tracing [90] and they reported performance improvement. However, our choice of implementation is mainly for simplicity, and photon tracing is not our primary focus.

10.2.4 Implementation

We describe implementation details of our hashing scheme. We used GLSL and OpenGL as the platform, but our algorithm can be implemented with other data-parallel programming languages such as NVIDIA CUDA. The hash table construction consists of the following three steps:

1. Compute hash values.
2. Scatter photons with the stochastic selection.
3. Multiply photon flux by the numbers of collisions.

The all steps are performed in parallel over the all photons. We describe details in the following. Note that the range query is done as in Section 10.2.2, and the updating procedure stays the same as the CPU version.

In order to compute hash values, we define uniform grid over a scene's bounding box and compute the hash value of each photon using the integer grid index [114]. The hash function that we use is a pseudo random number generator [76], which is also used for eye ray/photon tracing. The hash function takes the grid index as the random number seed and outputs a third random number as the hashed index.

Scattered writes of photons into the hash table (i.e., scatter operation) are done with a vertex buffer object in OpenGL that contains one vertex per photon. Each photon is mapped into the hash entry by perturbing the vertex position using a vertex texture fetch of the hash value.

The stochastic selection can be done by two different approaches. The first approach is assigning a uniform random value as a each photon's depth value, and simply renders all the photons with depth buffer enabled. Since probability that one of the photons in the same hash entry has the smallest depth value is constant, this

Table 10.1: Performance of our ray tracing core for incoherent rays and end-to-end performance of our GPU implementation of progressive photon mapping and path tracing. Units are 2^{20}/sec. The numbers shown are the numbers of emitted photons for PPM and the numbers of paths for PT.

Scene	Triangles	Intersection	PPM	PT
Box	4360	20.6	0.94	3.18
Cognac	16456	14.3	3.1	1.76
Pool	122900	17.5	3.2	1.05
Kitchen	44629	13.1	0.35	1.68

effectively implements the stochastic selection described in Section 10.2.1 without making a complete list of photons.

The second approach is a simpler variation of the first approach, which uses the pixel index as the each photon's depth value. The rest of the process is the same as the first approach. This assumes that each pixel generates statistically independent photons. This is true if we use uniform sampling for photon tracing, and under this assumption, the probability that one of the photons mapped to the same hash entry has the smallest depth value is constant. We use this second approach since its implementation is simpler.

We count the number of collisions in each hash entry (i.e., the number of collisions) by doing additive blending of 1.0 using the same vertex buffer. It performs atomic add operations of 1s into hash entries. Since the maximum count needed is equal to the maximum number of hash collisions, which is often very small, we can use an 8 bit buffer to save bandwidth. We then multiply those numbers to stored photon flux, and finish the hash table construction.

Figure 10.1 shows the pseudocode of the stochastic spatial hashing algorithm. Note that the code is very short and simple.

```
procedure CONSTRUCT(Photon)
  HashIndex ← HASH(Photon.Position)
  Table[HashIndex] ← Photon
  ATOMICINC(Count[HashIndex])
procedure RETRIEVE(Query)
  HashIndex ← HASH(Query.Position)
  Photon ← Table[HashIndex]
  if NEIGHBORING(Query.Position, Photon.Position, Query.Radius)
    then { Photon.Flux ← Photon.Flux * Count[HashIndex]
         { return (Photon)
```

Figure 10.1: Stochastic spatial hashing algorithm. CONSTRUCT() adds a photon to the data structure for spatial hashing. RETRIEVE() find a neighboring photon of the given query position. HASH() computes the hash value from the given position. ATOMICINC() increments the value with a guarantee of isolation from concurrent processes working on the same value. NEIGHBORING() returns true if two positions are within the specified distance. The codes run for all the photons/query positions in parallel.

10.3 Results and Discussion

We measured performance of our implementation on AMD Radeon HD 4850. All images are rendered with resolution of 512^2 using 512^2 photons per pass up to 10 bounces. Eye rays are traced after every 10 photon passes. The source code of the example implementation will be available on the author's website.

Table 10.1 shows the performance of our implementation of incoherent ray-triangle intersections, progressive photon mapping (PPM), and path tracing (PT). Incoherent rays are generated by randomly picking two points on a scene's bounding sphere and connecting them. Although our code is not highly optimized, the GPU implementation of PPM is 1-2 orders of magnitude faster than the CPU. Figure 10.2 confirms that PPM is more efficient than path tracing for difficult light paths as it is claimed. Even though the algorithm of PPM is more complex than PT, our implementation is not significantly slower to the extent that it cancels out efficiency of PPM.

We investigated how the stochastic selection in Section 10.2.1 affects image quality in Figure 10.3. A smaller number of hash entires leads to more collisions, thus it could cause much noise in rendered images. We have found that our algorithm is not highly sensitive to the number of hash entries if the number of hash entires is at least equal to the number of photons processed per pass. Even compared with having a full list of photons, the stochastic selection does not add visually significant amount of noise.

Figure 10.4 compares the performance of our spatial hashing scheme with the latest approach by Alcantara et al. on the randomized voxels test in their paper [4]. We measured up to 3M elements because the number of photons per pass in PPM is usually smaller. Our construction is slightly faster and the retrieval is on par with their method. Using tree data structure is also a conceivable option, however, tree construction is slower than hash table construction (9ms for 200K photons [136]). Using previous work on GPU photon mapping to implement PPM will be just slower than our method mainly due to the construction step.

Figure 10.5 shows the breakdown of rendering time. We measured time for the hash table construction, range query with progressive radiance estimation, photon tracing, and eye ray tracing. The photon tracing is the most costly part in our implementation, whereas previous work using BVH photon map [24] reported that the range query with radiance estimation is the most costly part. The difference is due to faster construction and retrieval of our algorithm than BVH (about 150ms for construction/retrieval of 256K photons [24] to 10ms with similar photon tracing performance).

One limitation of our hashing scheme is that usage of a uniform grid is not scalable to a large scene. For example, if we render a city scene where we see both closer buildings and farther buildings, the uniform grid covers a large extent and the resolution of grid might be too coarse for the closer buildings. In fact, the photon tracing is not scalable to such a large scene either, so using uniform grid is not solely responsible to this issue. Another limitation is that our method cannot be used for applications that needs a complete list of hashed elements, such as collision detection.

Concurrently to our work, Yang et al. [135] presented a linked-list construction algorithm on GPUs which might be potentially useful for implementing standard spatial hashing on GPUs. The performance presented in their work looks quite promising, however, standard spatial hashing does not resolve the issue of irregular work distribution at the query time. At the same time, stochastic spatial hashing introduces additional noise in the resulting images. It would be interesting to compare full implementations of progressive photon mapping using our approach and their approach as future work.

10.4 Conclusion

We presented a data-parallel implementation of progressive photon mapping on GPUs. The main contribution is the idea of stochastic selection, which keeps a single photon in each hash entry with some probability, which we named as stochastic spatial hashing. This simple idea results in a data-parallel construction of hash table and uniform work distribution at data retrieval. We show the application to progressive photon mapping, which enables robust and accurate global illumination rendering on GPUs. The main benefit of our algorithm is a simple algorithm comes with sufficient performance.

10.5 Acknowledgements

The material in this chapter is, in part, a reproduction of the material in the presentation of Toshiya Hachisuka and Henrik Wann Jensen, “Parallel Progressive Photon Mapping on GPUs”, ACM SIGGRAPH Asia 2010 Technical Sketches, December 2010 [43]. The dissertation author was the primary investigator and author of this work. The dissertation author was supported by the 2010 NVIDIA Fellowship Program.

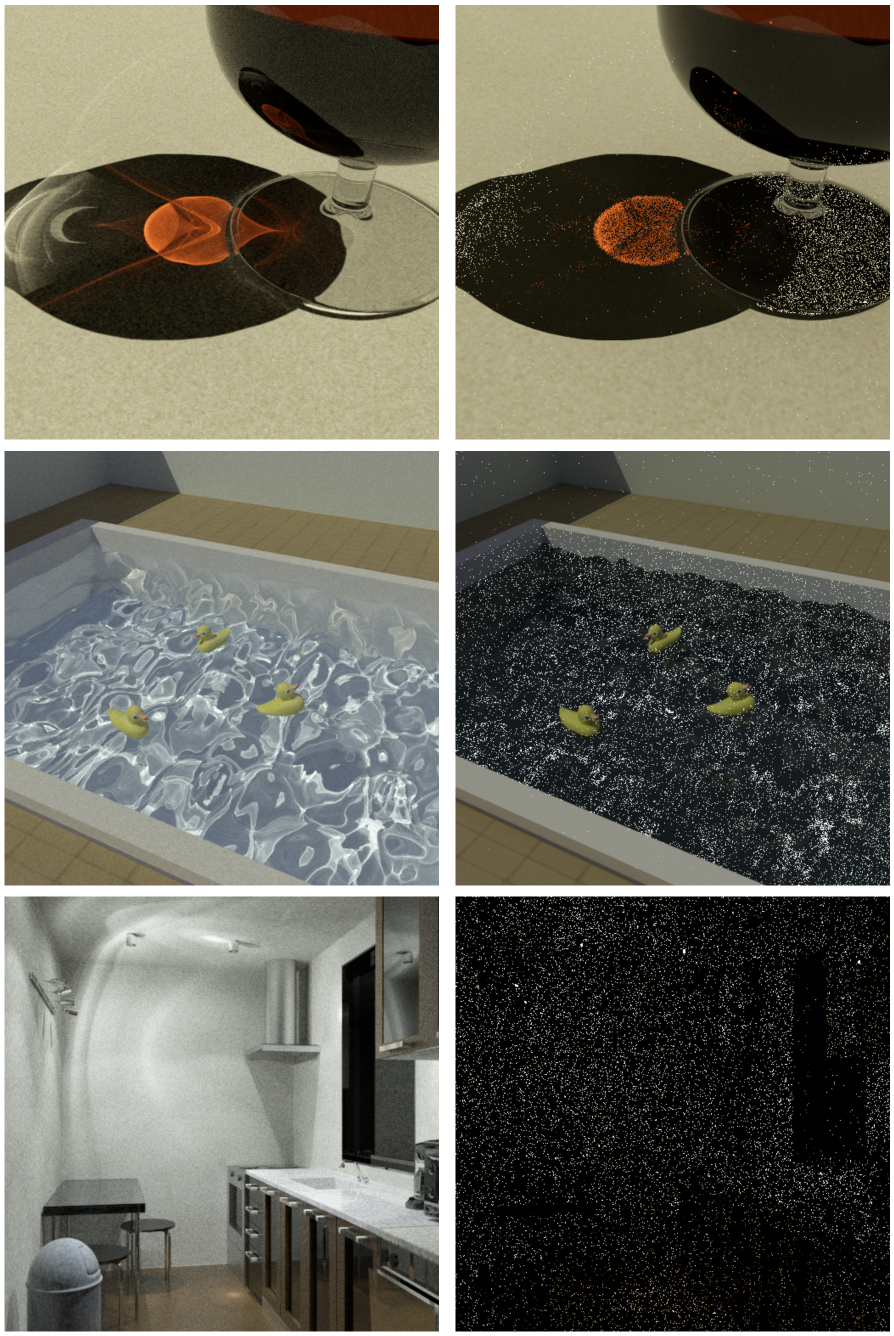


Figure 10.2: Rendered images using our implementations of progressive photon mapping and path tracing with shadow rays both on GPUs. Rendering time is 10 min in all scenes. The kitchen scene is entirely illuminated by caustics from light bulbs, which is common in the real world. A typical implementation of path tracing on GPUs cannot handle those paths efficiently, whereas progressive photon mapping is robust under such common illumination settings.

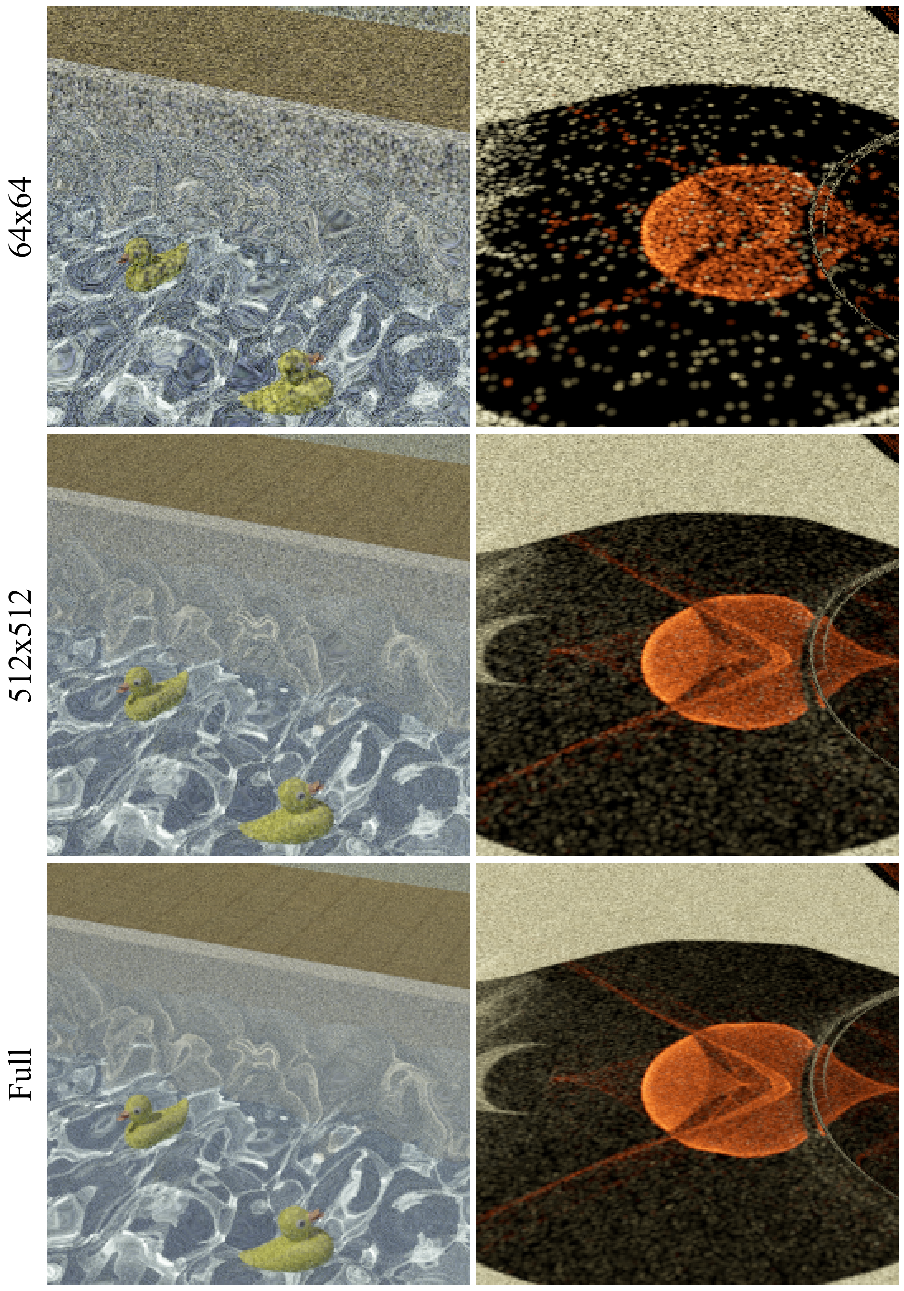


Figure 10.3: Effect of the stochastic selection on rendered images with 100M emitted photons using 512^2 photons per pass. The number of hash entries are 64^2 (top) and 512^2 (center). We also show images with a simulated full photon list (bottom). Fewer hash entries results in noisier images, but having as many hash entries as photons sufficiently suppresses extra noise due to the stochastic selection.

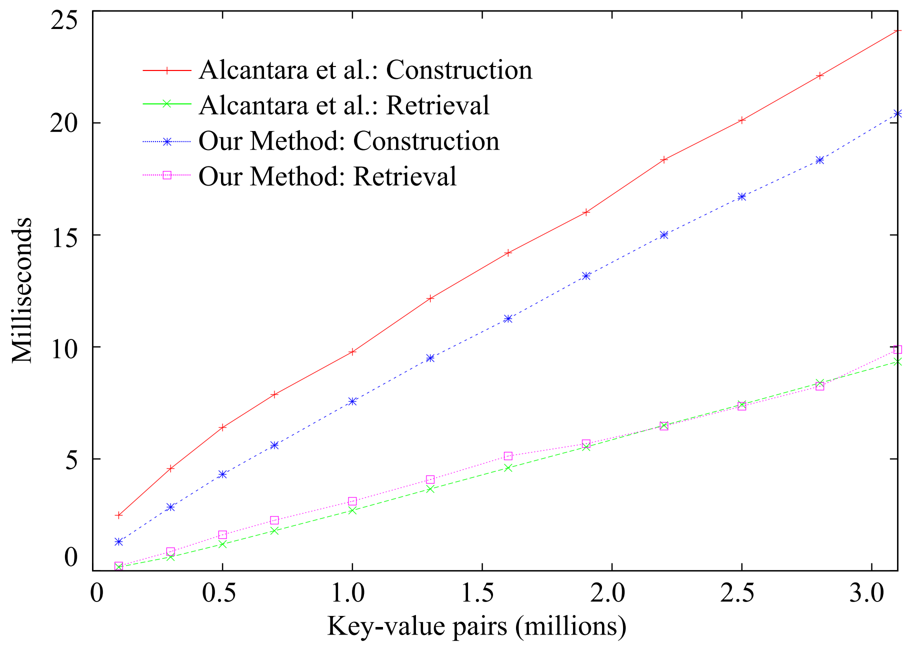


Figure 10.4: Performance comparison of hashing scheme with Alcantara et al. [4]. The measurement for the method by Alcantara et al. is directly taken from their paper, which is measured on GeForce GTX 280 SSC. Note that our measurement is done on Radeon HD 4850. Our construction is slightly faster and retrieval is on par with their method.

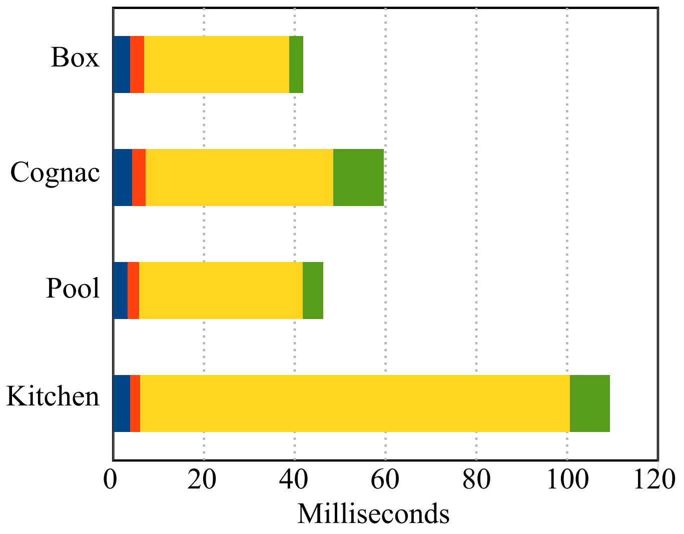


Figure 10.5: Breakdown of average rendering time per pass for the hash table construction (blue), range query with progressive radiance estimation (red), photon tracing (yellow), and eye ray tracing (green). The photon tracing is the most costly part of our implementation of progressive photon mapping.

Chapter 11

Conclusions

This dissertation introduced a new density estimation framework, progressive density estimation, for simulating light transport. This new framework can iteratively take more samples to get a density estimate that converges to the correct solution in the limit. This is the key difference from existing density estimation framework which needs to fix the number of samples in order to perform density estimation. Progressive density estimation utilizes the conditions for convergence in density estimation to build an iterative density estimator with provable convergence. We developed a new light transport simulation algorithm, progressive photon mapping, based on progressive density estimation. We demonstrated that progressive photon mapping is the first practical algorithm that can simulate many important optical configurations such as simulation of light fixtures from the filament level.

We also developed several extensions of this basic framework. Unbiased Monte Carlo ray tracing has been commonly believed to be the only approach that we can simulate light transport with a reasonable error estimator. In Chapter6, we provided a practical error estimation method for progressive density estimation which can be classified as biased Monte Carlo ray tracing. In Chapter 7, we introduced an extension of progressive density estimation, stochastic progressive density estimation, which is a converging estimator of average density over unknown region. This type of problem settings happens very often in computer graphics in the form of lens effects, and stochastic progressive photon mapping is a simple and practical solution for such problems.

Chapter 8 introduced a new adaptive computation method for photon trajectory simulation. The proposed method addresses inherent computational inefficiency of photon trajectory simulation by adaptively focusing computation to the region of interest. To the best of our knowledge, this is the first application of adaptive Markov chain Monte Carlo methods in light transport simulation. In Chapter 9, we provided a unified multiple importance sampling framework that combines progressive photon mapping and unbiased Monte Carlo ray tracing methods. By adaptively combining multiple algorithms, this framework improves the efficiency of the original progressive photon mapping under many scene configurations. We provided theoretical analysis that reveals the consequence of inclusion of progressive density estimation into the original multiple importance sampling framework. In order to accelerate the performance of the proposed framework on a parallel computation platform, we developed a new hashing algorithm in Chapter 10. The key feature of the algorithm is that it is designed to work regardless of the results of contention as opposed to algorithmically coping with contention of multiple processors. We demonstrated that the resulting algorithm efficiently runs on a graphics processing unit, which is a most commonly available parallel processing platform at the moment.

11.1 Future Work

First, we believe that the progressive density estimation framework is widely applicable to many problems that involve density estimation of a large number of samples. Although we used this framework only for light transport simulation in this dissertation work, there is no restricting assumptions made in progressive density estimation in the applications to other problems. For example, suppose that we would like to gather some statistics of transactions on an Internet search engine. Regular density estimation framework requires us to store all the samples taken in order to ensure convergence. This may be impractical if the number of transactions is very large. Progressive density estimation has a provable convergence without storing all the samples, which fundamentally avoid the issue of this storage requirement for many samples of transactions.

Stochastic progressive density estimation will be similarly useful for many general problems. In particular, it will be interesting to investigate whether the ability of estimating correct average density over unknown region is important outside light transport simulation. The error estimation framework should also be applicable to problems other than light transport simulation when a reasonable error estimator is necessary for density estimation. There are many avenues of research around progressive density estimation framework both from theoretical sides and practical sides.

We should note that our adaptive photon simulation method is only using the very basic form adaptive Markov chain Monte Carlo method. Further applications of adaptive Markov chain Monte Carlo methods in computer graphics should be worthwhile. For example, it would be interesting to look at if local adaptation of mutation strategies is possible and efficient in practice for light transport simulation.

The main idea of combining biased estimators and unbiased estimators in the modified multiple importance sampling has to be explored more. In general, we have found that light transport simulation in computer graphics is one of the few areas where biased estimators are investigated as well as unbiased estimators. Such biased estimators may potentially be useful for problems in other areas and the idea of combining biased and unbiased estimators will open up a large body of future work.

The general approach of designing a parallel algorithm that works regardless of the results of contention could be an interesting direction to pursue. Avoiding contention as efficiently as possible is often the main focus of many parallel processing algorithms. However, such an algorithm is generally complex and the requirement to hardware often comes with the algorithm. Our idea of utilizing statistical identity enabled a simple and efficient parallel algorithm for spatial hashing. Investigating this class of algorithms, which might be called as contention-insensitive parallel algorithms, would be interesting.

Finally, it should be noted that this dissertation is just another step toward the ultimate goal of perfect light transport simulation of the real world. The author hopes that this dissertation work serves as the basis for future light transport simulation algorithms.

Bibliography

[1] J. AGUIRREGABIRIA, A. HERNÁNDEZ, AND M. RIVAS, *δ–function converging sequences*, American Journal of Physics, 70 (2002), p. 180.

[2] I. AHMAD AND Y. FAN, *Optimal bandwidths for kernel density estimators of functions of observations*, Statistics & Probability Letters, 51 (2001), pp. 245–251.

[3] T. AILA AND S. LAINE, *Understanding the efficiency of ray traversal on gpus*, in HPG '09: Proceedings of the Conference on High Performance Graphics 2009, New York, NY, USA, 2009, ACM, pp. 145–149.

[4] D. ALCANTARA, A. SHARF, F. ABBASINEJAD, S. SENGUPTA, M. MITZENMACHER, J. OWENS, AND N. AMENTA, *Real-time parallel hashing on the gpu*, in SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers, New York, NY, USA, 2009, ACM, pp. 1–9.

[5] C. ANDRIEU AND C. ROBERT, *Controlled mcmc for optimal sampling*, tech. report, Tech. Rep. 0125, Cahiers de Mathématiques du Ceremade; UniversitéParis-Dauphine, 2001.

[6] C. ANDRIEU AND J. THOMS, *A tutorial on adaptive MCMC*, Statistics and Computing, 18 (2008), pp. 343–373.

[7] A. APPEL, *Some techniques for shading machine renderings of solids*, in Proceedings of the April 30–May 2, 1968, spring joint computer conference, AFIPS '68 (Spring), New York, NY, USA, 1968, ACM, pp. 37–45.

[8] J. ARVO, *Backward ray tracing*, in In ACM SIGGRAPH '86 Course Notes, Developments in Ray Tracing, 1986, pp. 259–263.

[9] P. BEKAERT, P. SLUSSALEK, R. COOLSAND, V. HAVRAN, AND H. SEIDEL, *A custom designed density estimator for light transport*, tech. report, Max-Planck-Institut für Informatik, 2003.

[10] G. BOX AND M. MULLER, *A note on the generation of random normal deviates*, Ann. Math. Statist., 29 (1958), pp. 610–611.

[11] T. CACOULLOS, *Estimation of a multivariate density*, Annals of the Institute of Statistical Mathematics, 18 (1966), pp. 178–189.

[12] M. CAMMARANO AND H. JENSEN, *Time dependent photon mapping*, in Proceedings of the 13th Eurographics workshop on Rendering, P. Debevec and S. Gibson, eds., Pisa, Italy, 2002, Eurographics Association, pp. 135–144.

[13] N. CENCOV, *Evaluation of an unknown distribution density from observations*, Soviet Mathematics, 3 (1962), pp. 1559–1562.

[14] P. CHRISTENSEN AND D. BATALI, *An irradiance atlas for global illumination in complex production scenes*, in Proceedings of Eurographics Symposium on Rendering 2004, June 2004, pp. 133–141.

[15] D. CLINE, J. TALBOT, AND P. EGBERT, *Energy redistribution path tracing*, in SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, New York, NY, USA, 2005, ACM, pp. 1186–1195.

[16] M. COHEN AND D. GREENBERG, *The hemi-cube; a radiosity solution for complex environments*, Computer Graphics (Proceedings of SIGGRAPH 85), 19 (1985), pp. 31–40.

[17] R. COOK, T. PORTER, AND L. CARPENTER, *Distributed ray tracing*, Computer Graphics (Proceedings of SIGGRAPH 84), (1984), pp. 137–145.

[18] C. DACHSBACHER, M. STAMMINGER, G. DRETTAKIS, AND F. DURAND, *Implicit visibility and antiradiance for interactive global illumination*, in ACM SIGGRAPH 2007 papers, SIGGRAPH '07, New York, NY, USA, 2007, ACM.

[19] G. DE MONTRICHER, R. TAPIA, AND J. TOHOMPSON, *Nonparametric maximum likelihood estimation of probability densities by penalty function methods*, The Annals of Statistics, 3 (1975), pp. 1329–1348.

[20] L. DEVROYE, *On the pointwise and integral convergence of recursive kernel estiamtes of probability densities*, Utilitas Mathematica, 15 (1979), pp. 113–128.

[21] S. DUANE, A. KENNEDY, B. PENDLETON, AND D. ROWETH, *Hybrid monte carlo*, Physics Letters B, 195 (1987), pp. 216 – 222.

[22] P. DUTRÉ, P. BEKAERT, AND K. BALA, *Advanced Global Illumination*, AK Peters, Ltd., second ed., 2006.

[23] P. DUTRÉ, E. LAFORTUNE, AND Y. WILLEMS, *Monte carlo light tracing with direct computation of pixel intensities*, in Proceedings of Compugraphics '93, 1993, pp. 128–137.

[24] B. FABIANOWSKI AND J. DINGLIANA, *Interactive global photon mapping*, Computer Graphics Forum, 28 (2009), pp. 1151–1159.

[25] S. FAN, S. CHENNEY, AND Y. LAI, *Metropolis photon sampling with optional user guidance*, in Rendering Techniques '05 (Proceedings of the 16th Eurographics Symposium on Rendering), Eurographics Association, 2005, pp. 127–138.

[26] C. FIORIO, *Confidence intervals for kernel density estimation*, Stata Journal, 4 (2004), pp. 168–179(12).

[27] R. FISHER, *On an absolute criterion for fitting frequency curves*, Messenger of Mathematics, 41 (1912), pp. 155–160.

[28] E. FIX AND J. HODGES, *Discriminatory analsis, nonparametric estimation: Consistency properties*, tech. report, Randolph Field, Texas: USAF School of Aviation Medicine, 1951.

[29] D. FRADIN, D. MENEVEAUX, AND S. HORNA, *Out of core photon-mapping for large buildings*, in Proceedings of Eurographics Symposium on Rendering 2005, Eurographics, June 2005.

[30] E. FREES, *Estimating densities of functions of observations*, Journal of the American Statistical Association, 89 (1994), pp. 517–525.

[31] T. GASSER, H. MULLER, AND V. MAMMITZSCH, *Kernels for nonparametric curve estimation*, Journal of the Royal Statistical Society, 47 (1985), pp. 238–252.

[32] J. GEERTSEMA, *Sequential confidence intervals based on rank tests*, The Annals of Mathematical Statistics, 41 (1970), pp. 1016–1026.

[33] S. GEMAN AND D. GEMAN, *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, pp. 452–472.

[34] A. GLASSNER, *Principles of Digital Image Synthesis*, Morgan Kaufmann, 1995.

[35] J. GONDEK, G. MEYER, AND J. NEWMAN, *Wavelength dependent reflectance functions*, in Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, New York, NY, USA, 1994, ACM, pp. 213–220.

[36] I. GOOD AND R. GASKINS, *Nonparametric roughness penalties for probability densities*, Biometrika, 58 (1971), pp. 255–277.

[37] C. GORAL, K. TORRANCE, D. GREENBERG, AND B. BATTAILE, *Modeling the interaction of light between diffuse surfaces*, in Computer Graphics (Proceedings of ACM SIGGEAPH 84), New York, NY, USA, 1984, ACM Press, pp. 213–222.

[38] N. GORDON, D. SALMOND, AND A. SMITH, *Novel approach to nonlinear/non-gaussian bayesian state estimation*, Radar and Signal Processing, IEE Proceedings F In Radar and Signal Processing, 140 (1993), pp. 107–113.

[39] S. GORTLER, P. SCHRÖDER, M. COHEN, AND P. HANRAHAN, *Wavelet radiosity*, in Computer Graphics (Proceedings of SIGGRAPH 93), New York, NY, USA, 1993, ACM, pp. 221–230.

[40] H. HAARIO, E. SAKSMAN, AND J. TAMMINEN, *An adaptive metropolis algorithm*, Bernoulli, 7 (2001), pp. 223–242.

[41] T. HACHISUKA, W. JAROSZ, AND H. JENSEN, *A progressive error estimation framework for photon density estimation*, ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2010), 29 (2010), pp. 144:1–144:12.

[42] T. HACHISUKA AND H. JENSEN, *Stochastic progressive photon mapping*, in SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers, New York, NY, USA, 2009, ACM, pp. 1–8.

[43] T. HACHISUKA AND H. JENSEN, *Parallel progressive photon mapping on gpus*, in ACM SIGGRAPH ASIA 2010 Sketches, SA '10, New York, NY, USA, 2010, ACM, pp. 54:1–54:1.

[44] T. HACHISUKA AND H. JENSEN, *Robust adaptive photon tracing using photon path visibility*. To appear in ACM Transactions on Graphics (accepted with minor revisions on March 2011), 2011.

[45] T. HACHISUKA, S. OGAKI, AND H. JENSEN, *Progressive photon mapping*, ACM Transactions on Graphics (SIGGRAPH Asia Proceedings), 27 (2008), p. Article 130.

[46] T. HACHISUKA, J. PANTALEONI, AND H. JENSEN, *Multisampled progressive photon mapping*. submitted, 2011.

[47] P. HALL, *Effect of bias estimation on coverage accuracy of bootstrap confidence intervals for a probability density*, The Annals of Statistics, 20 (1992), pp. 675–694.

[48] P. Hanrahan, D. Salzman, and L. Aupperle, *A rapid hierarchical radiosity algorithm*, Computer Graphics (Proceedings of SIGGRAPH 91), 25 (1991), pp. 197–206.

[49] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, Springer Series in Statistics, Springer New York Inc., New York, NY, USA, 2001.

[50] V. Havran, J. Bittner, R. Herzog, and H. Seidel, *Ray maps for global illumination*, in Eurographics Symposium on Rendering, 2005, pp. 43–54.

[51] V. Havran, R. Herzog, and H. Seidel, *Fast final gathering via reverse photon mapping*, Computer Graphics Forum (Proceedings of Eurographics 2005), 24 (2005).

[52] H. Helmholtz, *Handbuch der physiologischen Optik*, 1856.

[53] R. Herzog, V. Havran, S. Kinuwaki, K. Myszkowski, and H. Seidel, *Global illumination using photon ray splatting*, in Eurographics 2007, vol. 26 of Computer Graphics Forum, Blackwell, 2007, pp. 503–513.

[54] H. Hey and W. Purgathofer, *Advanced radiance estimation for photon map global illumination*, Computer Graphics Forum, 21 (2002).

[55] J. Hoberock and J. Hart, *Arbitrary importance functions for metropolis light transport*, Computer Graphics Forum, 29 (2010), pp. 1993–2003.

[56] Y. Iba, *Extended ensemble monte carlo*, International Journal of Modern Physics C, 12 (2001), pp. 623–656.

[57] D. Immel, M. Cohen, and D. Greenberg, *A radiosity method for non-diffuse environments*, Computer Graphics (Proceedings of SIGGRAPH 86), 20 (1986), pp. 133–142.

[58] A. Izenman, *Recent developments in nonparametric density estimation*, Journal of the American Statistical Association, 86 (1991), pp. 205–224.

[59] W. Jakob, C. Regg, and W. Jarosz, *Progressive expectation–maximization for hierarchical volumetric photon mapping*, Computer Graphics Forum (Proceedings of EGSR 2011), 30 (2011).

[60] W. Jarosz, M. Zwicker, and H. Jensen, *The beam radiance estimate for volumetric photon mapping*, Computer Graphics Forum (Proc. Eurographics EG'08), 27 (2008), pp. 557–566.

[61] H. Jensen, *Global illumination using photon maps*, in Rendering Techniques '96, X. Pueyo and P. Schröder, eds., Eurographics, Springer-Verlag Wien New York, 1996, pp. 21–30.

[62] H. JENSEN, *Realistic image synthesis using photon mapping*, A. K. Peters, Ltd., Natick, MA, USA, 2001.

[63] H. JENSEN AND P. CHRISTENSEN, *Efficient simulation of light transport in scences with participating media using photon maps*, in Computer Graphics (Proceedings of SIGGRAPH 98), New York, NY, USA, 1998, ACM Press, pp. 311–320.

[64] H. JENSEN, F. SUYKENS, AND P. CHRISTENSEN, *A practical guide to global illumination using photon mapping*, in SIGGRAPH 2001 Course Notes # 38, ACM, Aug. 2001.

[65] M. JONES, J. MARRON, AND S. SHEATHER, *A brief survey of bandwidth selection for density estimation*, Journal of the American Statistical Association, 91 (1996), pp. 401–407.

[66] J. KAJIYA, *The rendering equation*, in Computer Graphics (Proceedings of SIGGRAPH 86), New York, NY, USA, 1986, ACM Press, pp. 143–150.

[67] C. KELEMEN, L. SZIRMAY-KALOS, G. ANTAL, AND F. CSONKA, *A simple and robust mutation strategy for the metropolis light transport algorithm*, Computer Graphics Forum (Eurographics), 21 (2002), pp. 531–540.

[68] A. KELLER, *A quasi-monte carlo algorithm for the global illumination problem in the radiosity setting*, in Monte-Carlo and Quasi-Monte Carlo Methods in Scientific Computing, Springer, 1995, pp. 239–251.

[69] S. KITAOKA, Y. KITAMURA, AND F. KISHINO, *Replica exchange light transport*, Computer Graphics Forum, 28 (2009), pp. 2330–2342.

[70] C. KNAUS AND M. ZWICKER, *Progressive photon mapping: A probabilistic approach*, ACM Trans. Graph., 30 (2011), pp. 25:1–25:13.

[71] E. LAFORTUNE AND Y. WILLEMS, *Bidirectional path tracing*, in Proceedings of CompuGraphics, 1993, pp. 95–104.

[72] E. LAFORTUNE AND Y. WILLEMS, *Using the modified phong brdf for physically based rendering*, Tech. Report CW197, Katholieke Universiteit Leuven, Nov. 1994.

[73] Y. LAI, S. FAN, S. CHENNEY, AND C. DYER, *Photorealistic image rendering with population monte carlo energy redistribution*, in Proceedings of the Rendering Techniques (EGSR), Grenoble, France, 2007, Eurographics Association, pp. 287–295.

[74] C. Lauterbach, M. Garland, S. Sengupta, D. Luebke, and D. Manocha, *Fast bvh construction on gpus.*, Comput. Graph. Forum, 28 (2009), pp. 375–384.

[75] F. Lavignotte and M. Paulin, *Scalable photon splatting for global illumination*, in GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia, New York, NY, USA, 2003, ACM, pp. 203–ff.

[76] P. L'Ecuyer, *Efficient and portable combined random number generators*, Commun. ACM, 31 (1988), pp. 742–751.

[77] M. Lee, R. Redner, and S. Uselton, *Statistically optimized sampling for distributed ray tracing*, SIGGRAPH Comput. Graph., 19 (1985), pp. 61–68.

[78] J. Lehtinen, M. Zwicker, E. Turquin, J. Kontkanen, F. Durand, F. Sillion, and T. Aila, *A meshless hierarchical representation for light transport*, ACM Trans. Graph., 27 (2008), pp. 37:1–37:9.

[79] G. Lepage, *A new algorithm for adaptive multidimensional integration*, in Journal of Computational Physics, 1978, pp. 27:192–203.

[80] J. Liu, F. Liang, and W. Wong, *The multiple-try method and local optimization in metropolis sampling*, Journal of the American Statistical Association, 95 (2000), pp. 121–134.

[81] L. Lovász and P. Winkler, *Exact mixing in an unknown Markov chain*, Electronic Journal of Combinatorics, 2 (1995). Paper #R15.

[82] V. Ma and M. McCool, *Low latency photon mapping using block hashing*, in Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, HWWS '02, Aire-la-Ville, Switzerland, Switzerland, 2002, Eurographics Association, pp. 89–99.

[83] J. Marron and D. Nolan, *Canonical kernels for density estimation*, tech. report, University of North Carolina, Chapel Hill, 1987.

[84] M. McGuire and D. Luebke, *Hardware-accelerated global illumination by image space photon mapping*, in HPG '09: Proceedings of the Conference on High Performance Graphics 2009, New York, NY, USA, 2009, ACM, pp. 77–89.

[85] N. Metropolis and S. Ulam, *The monte carlo method*, Journal of the American Statistical Association, 44 (1949), pp. 335–341.

[86] K. Myszkowski, *Lighting reconstruction using fast and adaptive density estimation techniques*, in Proceedings of the Eurographics Workshop on Rendering Techniques '97, London, UK, 1997, Springer-Verlag, pp. 251–262.

[87] R. NEAL, *Slice sampling*, Annals of Statistics, 31 (2000), pp. 705–767.

[88] F. NICODEMUS, J. RICHMOND, J. HSIA, I. GINSBERG, AND T. LIMPERIS, *Radiometry*, Jones and Bartlett Publishers, Inc., , USA, 1992, ch. Geometrical considerations and nomenclature for reflectance, pp. 94–145.

[89] H. NIEDERREITER, *Random Number Generation and Quasi-Monte Carlo Methods*, Society for Industrial Mathematics, 1992.

[90] J. NOVAK, V. HAVRAN, AND C. DACHSBACHER, *Path regeneration for interactive path tracing*, in Short Papers, Proceedings of Eurographics 2010, 2010, p. to appear.

[91] A. OWEN, *Quasi-Monte Carlo sampling*, in Monte Carlo Ray Tracing: Siggraph 2003 Course 44, H. W. Jensen, ed., SIGGRAPH, 2003, pp. 69–88.

[92] A. OWEN AND Y. ZHOU, *Safe and effective importance sampling*, Journal of the American Statistical Association, 95 (1998), pp. 135–143.

[93] E. PARZEN, *On estimation of a probability density function and mode*, The Annals of Mathematical Statistics, 33 (1962), pp. 1065–1076.

[94] K. PERLIN, *Improving noise*, in SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, New York, NY, USA, 2002, ACM, pp. 681–682.

[95] T. PURCELL, C. DONNER, M. CAMMARANO, H. JENSEN, AND P. HANRAHAN, *Photon mapping on programmable graphics hardware*, in SIGGRAPH '05: ACM SIGGRAPH 2005 Courses, New York, NY, USA, 2005, ACM, p. 258.

[96] W. PURGATHOFER, *A statistical method for adaptive stochastic sampling*, Computer & Graphics, 11 (1987). Pergamon Press, New York.

[97] R. RAMAMOORTHI, D. MAHAJAN, AND P. BELHUMEUR, *A first-order analysis of lighting, shading, and shadows*, ACM Trans. Graph., 26 (2007), p. 2.

[98] G. ROBERTS, A. GELMAN, AND W. GILKS, *Weak convergence and optimal scaling of random walk metropolis algorithms*, Ann. Appl. Probab., 7 (1997), pp. 110–120.

[99] J. ROSENTHAL, S. BROOKS, A. GELMAN, G. JONES, AND X. MENG, *Optimal proposal distributions and adaptive MCMC*. Chapter For MCMC Handbook, 2008.

[100] B. SALEH AND M. TEICH, *Fundamentals of Photonics*, Wiley-Interscience, 2 ed., March 2007.

[101] R. Schregle, *Bias compensation for photon maps*, in Computer Graphics Forum 22, 4 (2003), C792-C742, 2003.

[102] D. Scott, *On optimal and data-based histograms*, Biometrika, 66 (1979), pp. 605–610.

[103] B. Segovia, J. Iehl, and B. Peroche, *Coherent metropolis light transport with multiple-try mutations*, Tech. Report RR-LIRIS-2007-015, Apr. 2007.

[104] P. Shirley, B. Wade, P. Hubbard, D. Zareski, B. Walter, and D. Greenberg, *Global illumination via density estimation*, Rendering Techniques 95 (Proceedings of Eurographics Workshop on Rendering 95), (1995), pp. 219–230.

[105] B. Silverman, *Density Estimation for Statistics and Data Analysis*, Mongraphs on Statistics and Applied Probability, Chapman and Hall, New York, NY, 1986.

[106] B. Smits, J. Arvo, and D. Salesin, *An importance-driven radiosity algorithm*, Computer Graphics (Proceedings of SIGGRAPH 92), 26 (1992), pp. 273–282.

[107] B. Smits and H. Jensen, *Global illumination test scenes*, Tech. Report UUCS-00-013, University of Utah, 2000.

[108] I. Stephenson, *Production Rendering*, SpringerVerlag, 2004.

[109] G. Stokes, *On the perfect blackness of the central spot in newton's rings, and on the verification of fresnel's formulae for the intensities of reflected and refracted rays*, Cambridge and Dublin Mathematical Journal, 4 (1849), pp. 1–14.

[110] F. Suykens and Y. Willems, *Adaptive filtering for progressive monte carlo image rendering*, in Eighth International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG 2000), Plzen, Czech Republic, 2000.

[111] R. Swendsen and J. Wang, *Replica monte carlo simulation of spin-glasses*, Physical Review Letters, 57 (1986), pp. 2607+.

[112] R. Tamstorf and H. Jensen, *Adaptive smpling and bias estimation in path tracing*, in Proceedings of the Eurographics Workshop on Rendering Techniques '97, London, UK, 1997, Springer-Verlag, pp. 285–296.

[113] D. Tannenbaum, P. Tannenbaum, and M. Wozny, *Polarization and birefringency considerations in rendering*, in Proceedings of the 21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94, New York, NY, USA, 1994, ACM, pp. 221–222.

[114] M. TESCHNER, B. HEIDELBERGER, M. MUELLER, D. POMERANETS, AND M. GROSS, *Optimized spatial hashing for collision detection of deformable objects*, in Proceedings of VMV'03, 2003, pp. 47–54.

[115] N. THRANE AND L. SIMONSEN, *A comparison of acceleration structures for gpu assisted ray tracing*, master's thesis, University of Aarhus, 2005.

[116] G. TORRIE AND J. VALEAU, *Nonphysical sampling distributions in Monte-Carlo free energy estimation - Umbrella Sampling*, J. Comput. Phys., 23 (1977), pp. 183–199.

[117] E. VEACH, *Robust Monte Carlo Methods for Light Transport Simulation*, PhD thesis, Stanford University, Dec. 1997.

[118] E. VEACH AND L. GUIBAS, *Optimally combining sampling techniques for Monte Carlo rendering*, in Computer Graphics (Proceedings of SIGGRAPH 95), New York, NY, USA, 1995, ACM Press, pp. 419–428.

[119] E. VEACH AND L. GUIBAS, *Metropolis light transport*, in Proceedings of the 24th annual conference on Computer graphics and interactive techniques, SIGGRAPH '97, New York, NY, USA, 1997, ACM Press/Addison-Wesley Publishing Co., pp. 65–76.

[120] J. VORBA AND J. KŘIVÁNEK, *Bidirectional photon mapping*, in Proceedings of CESCG 2011, The 15th Central European Seminar on Computer Graphics (non-peer-reviewed), 2011.

[121] B. WALTER, *Density estimation techniques for global illumination*, PhD thesis, Cornell University, Ithaca, NY, USA, 1998. Chair-Donald P. Greenberg.

[122] B. WALTER, A. ARBREE, K. BALA, AND D. GREENBERG, *Multidimensional lightcuts*, ACM Trans. Graph., 25 (2006), pp. 1081–1088.

[123] B. WALTER, S. FERNANDEZ, A. ARBREE, K. BALA, M. DONIKIAN, AND D. GREENBERG, *Lightcuts: a scalable approach to illumination*, ACM Trans. Graph., 24 (2005), pp. 1098–1107.

[124] B. WALTER, P. HUBBARD, P. SHIRLEY, AND D. GREENBERG, *Global illumination using local linear density estimation*, ACM Transactions on Graphics, 16 (1997), pp. 217–259.

[125] R. WANG, R. WANG, K. ZHOU, M. PAN, AND H. BAO, *An efficient gpu-based approach for interactive global illumination*, ACM Trans. Graph., 28 (2009), pp. 1–8.

[126] G. WARD AND P. HECKBERT, *Irradiance gradients*, in Rendering Techniques '92, A. Chalmers, D. Paddon, and F. Sillion, eds., Eurographics, Bristol, UK, 1992, Consolidation Express Bristol, pp. 85–98.

[127] G. WARD, F. RUBINSTEIN, AND R. CLEAR, *A ray tracing solution for diffuse interreflection*, in SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques, New York, NY, USA, 1988, ACM, pp. 85–92.

[128] L. WASSERMAN, *All of Nonparametric Statistics (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[129] E. WEGMAN AND H. DAVIES, *Remarks on some recursive estimators of a probability density*, The Annals of Mathematical Statistics, 7 (1979), pp. 316–327.

[130] T. WHITTED, *An improved illumination model for shaded display*, Commun. ACM, 23 (1980), pp. 343–349.

[131] P. WHITTLE, *On the smoothing of probability density functions*, Journal of the Royal Statistical Society, 20 (1958), pp. 334–343.

[132] J. WIEBELT, *Engineering radiation heat transfer.*, New York, Holt, Rinehart and Winston, 1966.

[133] C. WOLVERTON AND T. WAGNER, *Recursive estimates of probability densities*, IEEE Transactions on Systems, Science and Cybernetics, 5 (1969), p. 307.

[134] H. YAMATO, *Sequential estimation of a continuous probability density function*, Bulletin of Mathematical Statistics, 14 (1971), pp. 1–12.

[135] J. YANG, J. HENSLEY, H. GRÜN, AND N. THIBIEROZ, *Real-time concurrent linked list construction on the gpu*, Computer Graphics Forum, 29 (2010), p. 1297Ű1304.

[136] K. ZHOU, Q. HOU, R. WANG, AND B. GUO, *Real-time kd-tree construction on graphics hardware*, ACM Trans. Graph., 27 (2008), pp. 1–11.