SEQUENTIAL MONTE CARLO METHODS FOR PHYSICALLY BASED RENDERING

by

Shaohua Fan

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2006

To my family

ACKNOWLEDGMENTS

I am most indebted to my advisor Stephen Chenney for his inspiration and valuable guidance on my PhD work over the last few years. I feel so fortunate to have had the opportunity to work with Stephen because he is always very supportive, patient and encouraging. Stephen taught me a lot not only in computer graphics, but also in many aspects in life. He is very sharp as a researcher and extremely nice as a person. His stimulating insights and discussions help the research tremendously. His kindness and encouragement make the experience of working with him very enjoyable.

I also owe a great deal to my co-advisor, Chuck Dyer. When I approached Chuck about five years ago for a research position to work with him, all the computer science courses I have ever taken were Computer Vision and Data Structure and I was not even a graduate student in computer science. Maybe because that I did well in his class, or maybe because I have a mathematics background, or maybe simply because he was too nice to turn me down, Chuck took a chance and put his faith in me. He brought me into the computer science Ph.D. program one year later and has taught me so much over the years from doing research to writing papers. Chuck is a great mentor and has always been there when I needed advice. I really appreciate everything he has done for me.

I would like to thank my other committee members, Mike Gleicher, Kam-Wah Tsui, and Jerry Zhu. Mike is a strong believer of lab environment and makes the lab a pleasant place to work, from which I benefit so much. Mike's constructive comments and feedbacks on my work presented in the group meetings always stimulate new thoughts. Thank Kam and his graduate student Bo Hu in the Statistics Department for the regular discussions on sampling theory and the pleasant

collaboration. Lastly I thank Jerry for serving on my committee and taking the time to review my thesis.

I can not express sufficiently for how much I have enjoyed our computer graphics/vision group environment and how much I have learned from all our group members. I really liked the movie nights at the lab organized by Michael Wallick and Rachel Heck. I would like to mention Lucus Kovar and Mankyu Sung for the discussions and sharing of their experience on prelim and oral exams. I also appreciate Yu-chi Lai and Feng Liu for the “lunch meetings” and all the fun, relaxing chats we had. Special thank goes to Yu-chi for the discussion and cooperation on research.

Thank Matt Pharr and Gerg Humphreys for their wonderful book *Physically Based Rendering* and well-designed system PBRT. I especially thank Matt for his time and kindness in answering my questions and giving feedbacks to our paper drafts.

I am extremely grateful to Julia Foster, our “American Granny” and house-mate, who provides us a home full of love. She teaches us English with much patience and creativity. Over the years, I have read articles in the newspapers to her, whereupon she would correct my pronunciation along the way. We have so many laughs hearing some strange words out of my mouth. I do not remember how many papers and letters she has helped to proofread. Her whole family has been very friendly and generous to us. They invite us to every family party and treat us exactly the same as the rest of the family members. They have provided us not only an unique opportunity of learning about American culture, but also a feeling of family that we deeply appreciate and will never forget.

Finally, my deepest gratitude goes to Dandan, for everything.

DISCARD THIS PAGE

TABLE OF CONTENTS

Page

Page

DISCARD THIS PAGE

LIST OF TABLES

DISCARD THIS PAGE

LIST OF FIGURES

Figure Page

Figure Page

DISCARD THIS PAGE

NOMENCLATURE

Φ	Radiant Power or Flux
E	Irradiance
I	Intensity
L	Radiance
f_r	Bidirectional Reflectance Distribution Function
$\vec{p}$	A surface point
$\bar{X}$	A light transport path
$f(\bar{X})$	Path contribution function
$\pi(f)$	$\int_\Omega f(x)\pi(x)dx$
β	Bias
$\epsilon[F]$	Efficiency for estimator F
$\hat{I}_N$	Estimator for I with N samples
BRDF	Bidirectional Reflectance Distribution Function
DIS	Defensive Importance Sampling
DDIS	Deterministic Defensive Importance Sampling

DMS	Deterministic Mixture Sampling
ERPT	Energy Redistribution Path Tracing
MCMC	Markov chain Monte Carlo
MIS	Multiple Importance Sampling
MPS	Metropolis Photon Sampling
OCV	Optimizing Control Variate
PDF	Probability Distribution Function
P-Eff	Perceptually-based Mean Squared Efficiency
PMC	Population Monte Carlo
PMCR	Population Monte Carlo Rendering
SIR	Sampling Importance Resampling
SPP	Samples Per Pixel
SMC	Sequential Monte Carlo

SEQUENTIAL MONTE CARLO METHODS FOR PHYSICALLY BASED RENDERING

Shaohua Fan

Under the supervision of Assistant Professor Stephen J. Chenney and Professor Charles R. Dyer
At the University of Wisconsin-Madison

The goal of global illumination is to generate photo-realistic images by taking into account all the light interactions in the scene. It does so by simulating light transport behaviors based on physical principles. The main challenge of global illumination is that simulating the complex light interreflections is very expensive. In this dissertation, a novel statistical framework for physically based rendering in computer graphics is presented based on sequential Monte Carlo (SMC) methods. This framework can substantially improve the efficiency of physically based rendering by adapting and reusing the light path samples without introducing bias. Applications of the framework to a variety of problems in global illumination are demonstrated.

For the task of photo-realistic rendering, only light paths that reach the image plane are important because only those paths contribute to the final image. A visual importance-driven algorithm is proposed to generate visually important paths. The photons along those paths are also cached in photon maps for further reuse. To handle difficult paths in the path space, a technique is presented for including user-selected paths in the sampling process. Then, a more general statistical method for light path sample adaptation and reuse is studied in the context of sequential Monte Carlo. Based on the population Monte Carlo method, an unbiased adaptive sampling method is presented that works on a population of samples. The samples are sampled and resampled through distributions that are modified over time. Information found at one iteration can be used to guide subsequent iterations without introducing bias in the final result. After obtaining samples from multiple distributions, an optimal control variate algorithm is developed that allows samples from multiple distribution functions to be combined optimally.

Stephen J. Chenney Charles R. Dyer

ABSTRACT

The goal of global illumination is to generate photo-realistic images by taking into account all the light interactions in the scene. It does so by simulating light transport behaviors based on physical principles. The main challenge of global illumination is that simulating the complex light inter-reflections is very expensive. In this dissertation, a novel statistical framework for physically based rendering in computer graphics is presented based on sequential Monte Carlo (SMC) methods. This framework can substantially improve the efficiency of physically based rendering by adapting and reusing the light path samples without introducing bias. Applications of the framework to a variety of problems in global illumination are demonstrated.

For the task of photo-realistic rendering, only light paths that reach the image plane are important because only those paths contribute to the final image. A visual importance-driven algorithm is proposed to generate visually important paths. The photons along those paths are also cached in photon maps for further reuse. This approach samples light transport paths that connect a light to the eye, which accounts for the viewer in the sampling process and provides information to improve photon storage. Paths are sampled with a Metropolis-Hastings algorithm that exploits coherence among important light paths. To handle difficult paths in the path space, a technique is presented for including user-selected paths in the sampling process. This allows a user to provide hints about important paths to reduce variance in specific parts of the image.

A more general statistical method for light path sample adaptation and reuse is studied in the context of sequential Monte Carlo. Based on the population Monte Carlo method, an unbiased adaptive sampling method is presented that works on a population of samples. The samples are sampled and resampled through distributions that are modified over time. Information found at

one iteration can be used to guide subsequent iterations without introducing bias in the final result. This is the first application of the population Monte Carlo method to computer graphics.

After getting samples from multiple distributions, how the estimator is constructed for Monte Carlo integration has a big impact on the variance in the rendered images. Combining the idea of importance sampling and control variate, an optimal control variate algorithm is developed that allows samples from multiple distribution functions to be combined optimally. Its optimizing nature addresses a major limitation with control variate estimators for rendering: users supply a generic correlated function that is optimized for each estimate rather than a single highly-tuned one that must work well everywhere.

The population Monte Carlo rendering framework and optimized unbiased estimator result in more efficient and robust algorithms for global illumination. Significant improvements in results are demonstrated for various commonly existing environments such as scenes with non-uniform variance on the image planes and scenes with difficult but important paths.

Chapter 1

Introduction

Applications from special effects to product design demand realistic, physically based renderings of complex scenes. Images from physically based rendering algorithms not only look plausible but also can accurately predict the appearance of the real world or a virtual environment. Because of their realism and predictiveness, those images can be used to answer questions such as "what would this theater look like if we arrange lights at certain locations?" without actually installing those lights or even building the theater. As a result, physically based rendered images find applications in many fields such as light design, movies, architectural design, pilot training, and fine art. Even for less rigorous applications such as computer games and virtual reality walk-throughs, physically based rendering greatly enhances the visual appeal.

To see the differences between a non-physically based rendered image and a physically based rendered image, Figure 1.1 shows that while the image with only direct lighting and ambient lighting on the left looks realistic, it misses some important lighting effects such as color bleeding, soft shadows, and caustic that show up in the physically based rendered image on the right.

The goal of this thesis is to develop efficient algorithms for off-line physically based rendering. While there are many interesting applications for non-physically based rendering (e.g., non-photorealistic rendering [61] and interactive rendering [99]), they sacrifice realism for either artistic expression or interactive speed. Using principles of physics to simulate light transport, physically based rendering aids our understanding of the fundamental principles of image rendering, makes it possible to evaluate rendered images, and provides the basis for non-physically based applications.

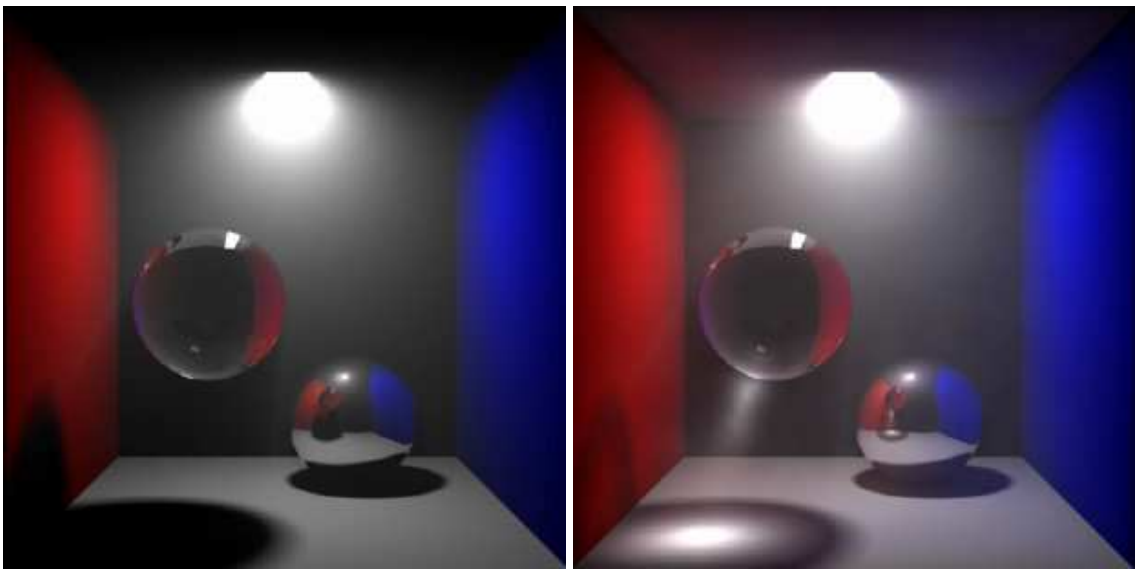

Figure 1.1: Image with direct lighting only vs. image with global illumination.

The reminder of this chapter presents the statement of problems, a brief summary of contributions, and an overview of the methods that will be developed in subsequent chapters.

1.1 The Global Illumination Problem

The input to a physically based rendering system is a full description of the 3D scene including the light sources, scene geometry, surface materials and camera information. In order to generate physically accurate images, we need to simulate all inter-reflections between the lights and objects in the scene model; this is called the global illumination problem [4, 92].

The physical foundation for image synthesis in global illumination is the rendering equation [47, 43] to which both finite element and Monte Carlo methods have been applied. Finite element methods, or radiosity algorithms [33], are most effective in purely diffuse scenes. Generalizations for non-diffuse surfaces and complex geometries turn out to be very difficult practically and theoretically. On the other hand, Monte Carlo methods for physically based rendering handle general reflectance functions and geometric representations. Kajiya [47] proposed the first unbiased Monte Carlo *path tracing* algorithm and introduced a range of sampling strategies to improve the method.

While demonstrated to be the most general and robust method, the main disadvantage of Monte Carlo for global illumination is that it is very expensive to compute if applied naively; if not enough samples are taken, noise appears in the resulting images and noise is reduced only slowly with increasing samples. This has limited the use of global illumination in production environments. Industry reports [93, 11] that the average computation time for rendering a single complex scene is still multiple hours using modern computers. Over the years, researchers have continued to develop new Monte Carlo algorithms with various trade-offs in speed, accuracy and generality [47, 107, 54, 102, 44, 103, 27, 13].

The efficiency of rendering can be improved significantly if more samples can be concentrated in the part of the sample space which matters the most and if the high-contribution samples can be reused. In computer rendered images, not all areas have the same impact on human perception. Some regions are more complex and detailed than others; some regions have high contrast; and some regions capture more human attention. Figure 1.2 shows how noises vary across the image plane in a global illumination image. It can be seen on the left image that the noise level on the shadowed ground regions is perceptually much higher than on the un-shadowed ground. The noise level difference can be several orders of magnitude. In this example, most of the noise is due to variation in incoming illumination: around the shadow boundary, those occluded light samples have zero contribution while the others have high contribution. This causes high sample variance on the rendered image. The variance can be reduced by either putting more image rays on those shadow boundary regions (i.e., adapt image plane) or by casting more shadow rays towards the visible light than the occluded light (i.e., adapt hemispheric direction).

The goal of this thesis is to develop robust and efficient unbiased Monte Carlo methods for the global illumination problem, which allows adaptively generating samples and reusing important samples. For these demands, we present a novel statistical framework based on sequential Monte Carlo (SMC) methods [21, 59] for physically based rendering. It is demonstrated that sequential Monte Carlo methods can be used to efficiently generate and reuse path samples for physically based rendering.

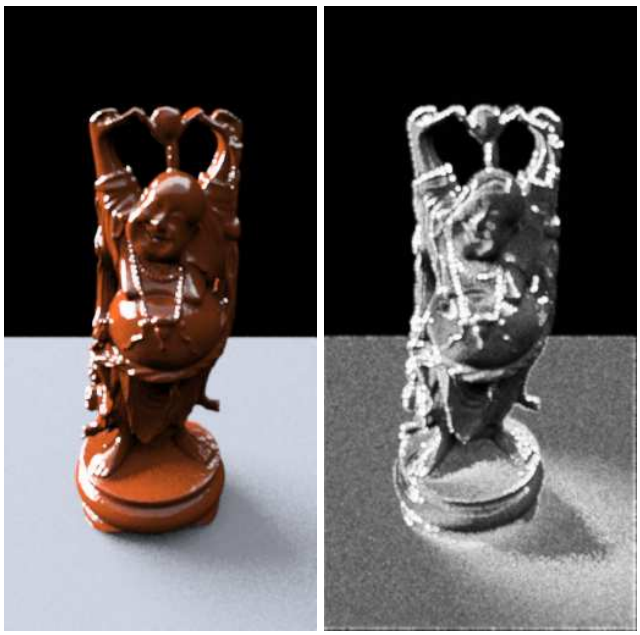

Figure 1.2: A global illumination image and its noise distribution. Whiter regions on the right image indicate higher noise on the left image.

Mathematically, the problem can be stated as follows: Given a target probability distribution $\pi(x)$ defined on a common measurable space Ω, and a measurement function, $f(x)$, where $x \in \Omega$, introduce a sequence of intermediate proposal distributions, $\pi_t(x), t = 1, \cdots, N$, which are calibrated to converge to $\pi(x)$ along N iterations so that the Monte Carlo estimator for $\int_\Omega f(x)\pi(x)dx$ based on all the samples from those distributions will converge quickly with low variance and be unbiased.

Figure 1.3 shows how sequential Monte Carlo methods can help sampling a target distribution. In the target distribution, $\pi(x)$, there are two modes that we assume are a combination of two underlying distributions, $p_1(x)$ and $p_2(x)$, but the weight functions, $w_1(x)$ and $w_2(x)$, are unknown (Sequence 0). Note that the weighting functions could be nonlinear and their values depend on x. Since no *a priori* information about the weighting functions is available in the initial step, we set them uniformly and generate samples from $0.5p_1(x) + 0.5p_2(x)$ (Sequence 1). Based on the samples from the previous iteration, the importance function can be adjusted to get closer to the target distribution, $\pi(x)$ (Sequence 2). Repeating this process results in a sequence of intermediate distributions that converges towards $\pi(x)$.

To address this problem in the context of global illumination, we subdivide it into the following three sub-problems.

- **How to design and evolve the sequence of distributions so that the later distributions can be improved based on the performance of previous distributions?**

  In Monte Carlo algorithms for the global illumination problem, it is essential to use good sampling techniques so that noise in the rendered image will be reduced quickly. However, the optimal sample technique is often difficult to choose beforehand because it depends on parameters whose values are only known during the sampling process; for example, the material and geometry of the surface point that the sampled ray intersects.

  In the Checker scene (Figure 1.4), there are two area lights of different sizes, and three surface materials – diffuse, specular and glossy. If we pre-set a fixed sampling strategy such as BRDF sampling or light sampling, it would be good for one region, but very bad for other regions. For example, light sampling works very well for diffuse surfaces but does a poor job for the specular regions in front of the big area light.

  The problem is how to detect lighting conditions and crate the best importance function without introducing bias. In order to efficiently estimate the direct lighting for all surface points, a sensible sampling strategy should take into account of a combination of factors that affect the sampling, and adjust the sampling technique on the fly based on the performance of the samples. This thesis shows that population Monte Carlo method can be used to solve this problem. In the scene above, we would like to detected that the light sampling technique for the specular regions in front of the big area light generate high variance samples, and then use BRDF sampling instead.

- **How to generate samples and reuse the high-contribution but difficult samples for one single target distribution?**

  Due to the geometric setting and material properties, some light path samples may be more difficult to detect in the sample space. For example, the caustic path in the Cornell Box scene

and light paths starting from the back room and passing through the ajar door in the Room scene (Figure 1.5). After those samples are generated, they should be reused to locally explore nearby important paths. We present an algorithm based Metropolis sampling and photon caching for samples reuse. Population Monte Carlo can be used for it too.

- **How to optimally combine the samples from a sequence of distributions to minimize the estimation variance without introducing bias?**

  For example, in computing the direct lighting for the Checker scene (Figure 1.4), the estimator that combines the samples from BRDF sampling and light sampling makes a big difference on the image variance. A naive linear combination of the samples from different sampling techniques does not work well because if any of those sampling techniques has high variance, then the estimator produced by the linear combination will have high variance as well. A OCV estimator is proposed to address that problem.

Sequential Monte Carlo methods provide a mechanism for meeting these requirements. The key idea is that samples from the previous distribution in the sequence can provide information to improve the proposal distributions and be reused to discover other high-contribution samples.

SMC techniques offer four major advantages over existing methods: (1) they reduce estimation variance by choosing samples correlated across the distributions without introducing bias; (2) they make it easier to find important, rare light paths by sharing information among the distributions; (3) they provide a natural way to discard low contribution samples and retain high contribution samples based on the sample weight; and (4) this framework unifies many existing rendering algorithms such as path tracing, metropolis light transport, energy redistribution path tracing, multiple importance sampling, and adaptive importance sampling.

1.2 Summary of Contributions

We introduce novel applications of the sequential Monte Carlo method to computer graphics that lead to new adaptive sampling algorithms for physically based rendering. Our main contributions are the following:

- **Metropolis Photon Sampling (MPS):** As a way of generating and reusing important path samples, we propose a visual importance-driven algorithm for populating photon maps. Our approach samples light transport paths that join a light to the eye, which accounts for the viewer in the sampling process and provides information to improve photon storage. Paths are sampled with a Metropolis-Hastings algorithm that exploits coherence among important light paths. We also present a technique for including user selected paths in the sampling process without introducing bias. This allows a user to provide hints about important paths or reduce variance in specific parts of the image.

- **Population Monte Carlo Rendering (PMCR):** Based on population Monte Carlo, we develop an unbiased adaptive sampling method that works on a population of samples. The sample population is iterated through distributions that are modified over time. Information found in one iteration can be used to guide subsequent iterations, without introducing bias in the final result.

- **Optimizing Control Variate (OCV):** Combining the idea of importance sampling and control variate, OCV allows samples from multiple distribution functions to be combined in one algorithm. Its optimizing nature addresses a major limitation with control variate estimators for rendering: users supply a generic correlated function which is optimized for each estimate rather than a single highly-tuned one that must work well everywhere.

1.3 Thesis Outline

Chapter 2 of the thesis gives an overview of Monte Carlo methods. After a brief history of Monte Carlo methods, the principle of Monte Carlo integration, which uses Monte Carlo simulation to estimate an integration, is described. Next, some variance reduction techniques such as importance sampling and control variates are introduced. We further introduce the concept of MCMC – metropolis sampling. For sequential Monte Carlo methods, two approaches most applicable to computer graphics are discussed: Sampling Importance Resampling and population Monte Carlo.

Chapter 3 introduces the basic concepts related to global illumination and physically based rendering. After providing the definition of the four most commonly used terms in radiometry, surface BRDF and the rendering equation are presented. A summary of existing representative rendering algorithms involving Monte Carlo methods to solve the global illumination problem is given.

Chapter 4 presents *Metropolis Photon Sampling* (MPS), a visual importance-driven algorithm for populating photon maps. A technique for including user-selected paths in the sampling process without introducing bias is presented.

Chapter 5 presents a novel statistical framework for image rendering called *Population Monte Carlo Rendering* (PMCR). PMCR works on a population of samples that is iterated through distributions that are modified over time. We show its application to a number of problems in realistic rendering.

Chapter 6 discusses the *Optimizing Control Variate* estimator, a new estimator for Monte Carlo rendering that combines the samples from different distributions in a provably good way.

Chapter 7 concludes with a summary and the original contributions in the thesis, and identifies some future research directions.

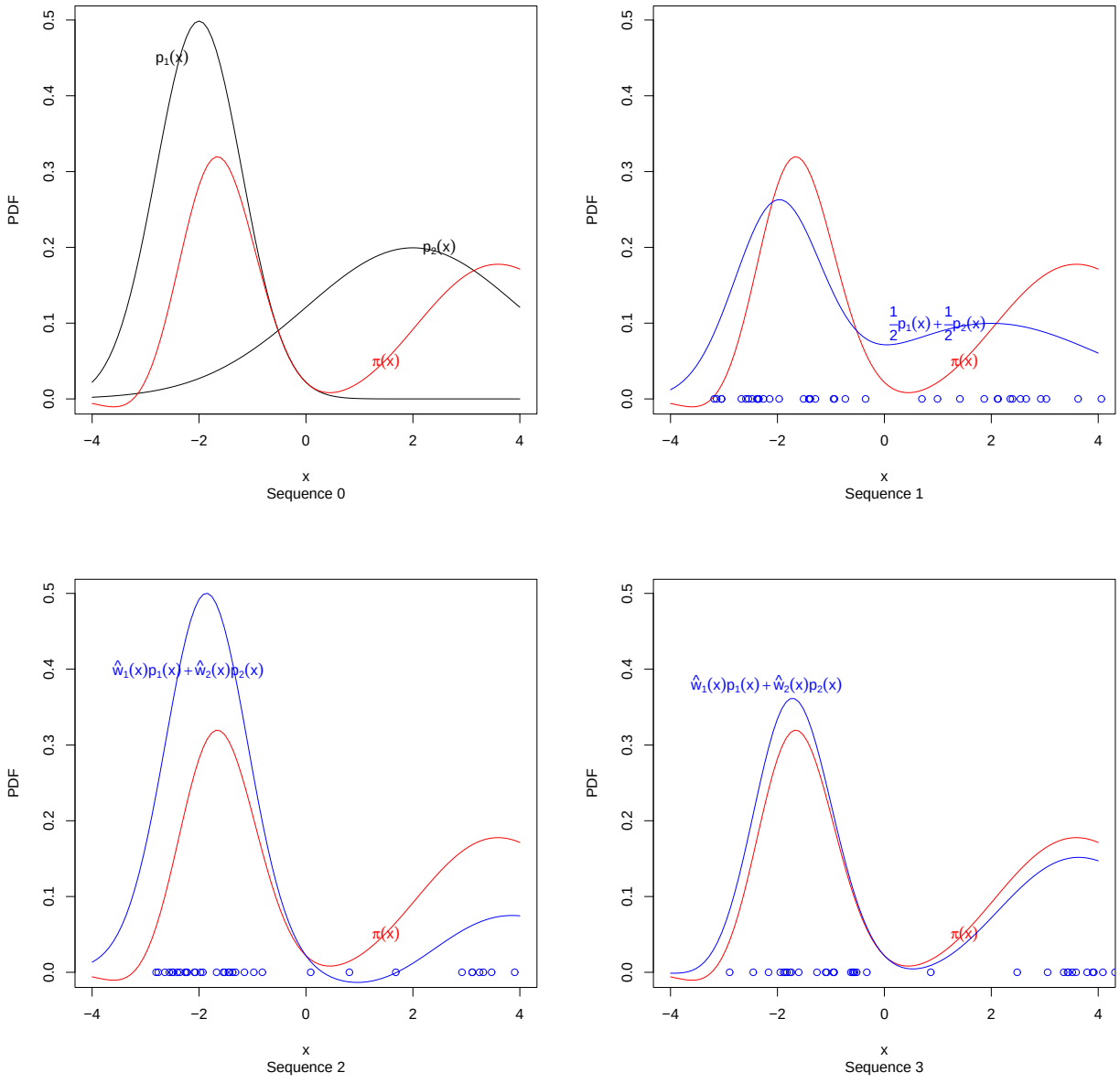

Figure 1.3: SMC distributions. Sequence 0 shows the target distribution $\pi(x)$ and the two underlying basis distribution $p_1(x)$ and $p_2(x)$. Sequence 1 uses $\frac{1}{2}p_1(x) + \frac{1}{2}p_2(x)$ as an importance distribution to generate samples. Based on the samples in sequence 1, the weighting functions $\hat{w}_1(x)$ and $\hat{w}_2(x)$ are adjusted so that the importance distribution is closer to $\pi(x)$. New samples are generated from the updated density $\hat{w}_1(x)p_1(x) + \hat{w}_2(x)p_2(x)$. Repeating the process hopefully leads to an importance distribution that is very close to the target distribution $\pi(x)$ so that the sample variance is low.

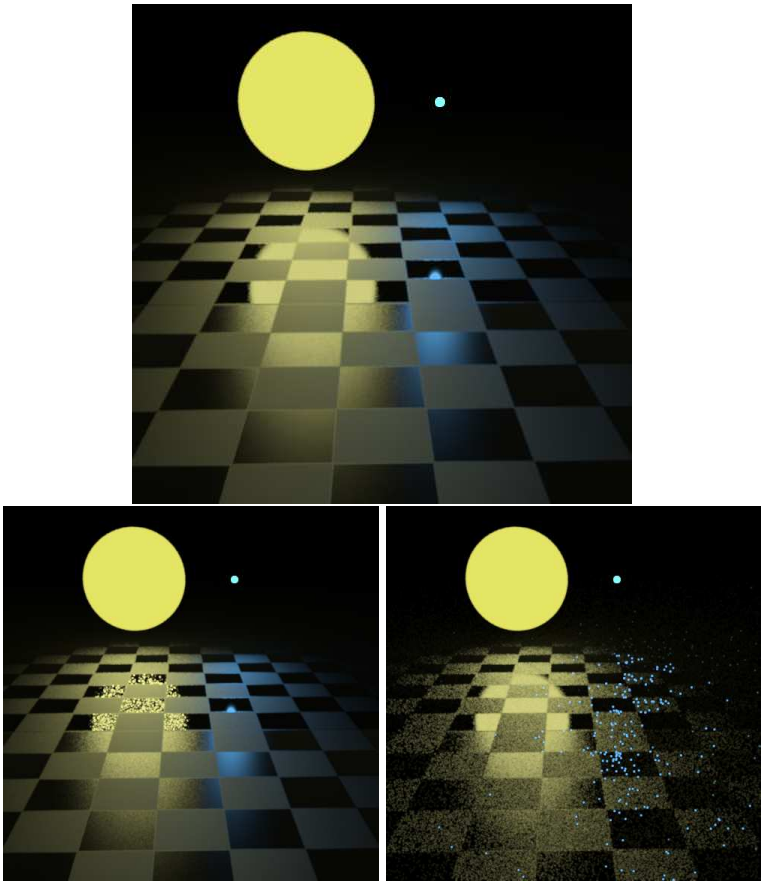

Figure 1.4: Checker scene (top) consists of two area lights of different sizes, and three different surface materials. Light sampling (bottom-left) does very poorly in the specular region in front of the big area light, while the BRDF image (bottom-right) appears very noisy on the diffuse surface. Both do poorly in glossy regions.

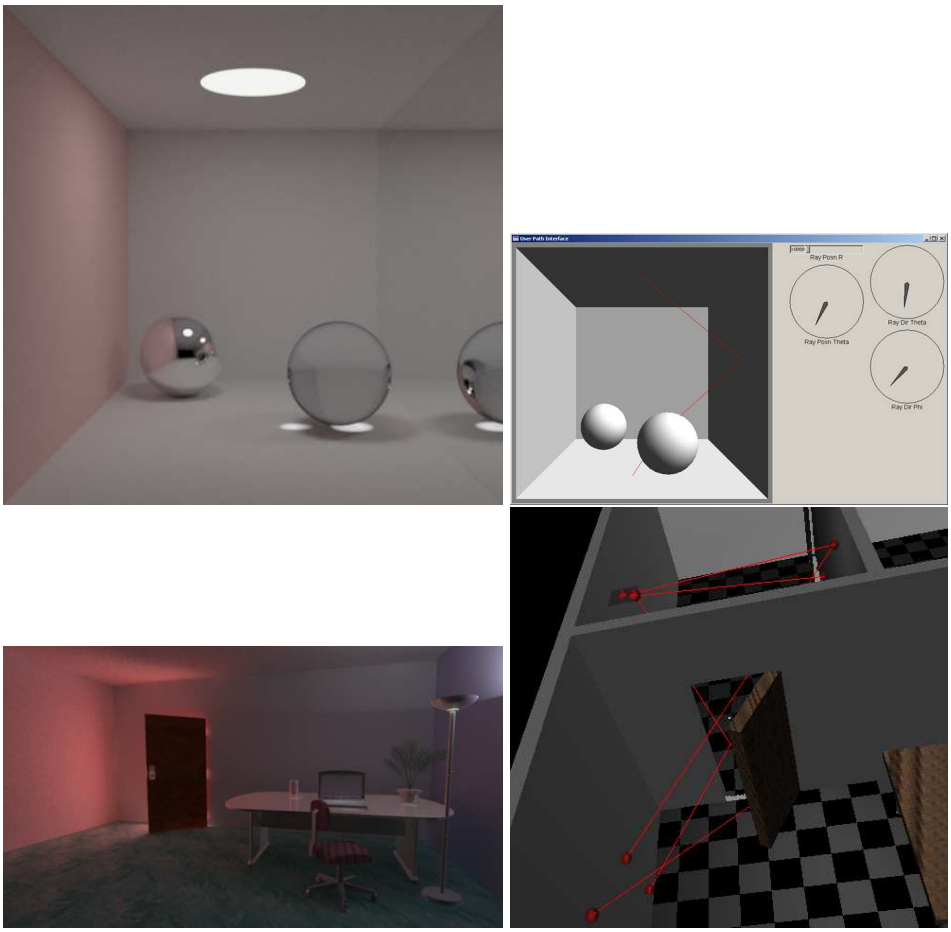

Figure 1.5: Difficult paths. In the Cornell Box scene, the light paths shown in the top-right image are very difficult to sample because they have to bounce from the mirror to a certain region on the glass ball to form caustic. The light paths marked for the Room scene (bottom-right) are difficult because they have to pass through a narrow door way after bouncing from a wall or the door.

Chapter 2

Monte Carlo Methods

This chapter introduces some basic statistical concepts and Monte Carlo methods. After a brief overview of Monte Carlo methods, the principle of Monte Carlo integration is introduced. Then, a number of variance reduction techniques such as importance sampling, control variate, and deterministic mixture sampling are described. We also present Metropolis-Hastings sampling, which is a Monte Carlo method using dependent samples and is the basic building block for Markov chain Monte Carlo (MCMC). Sequential Monte Carlo (SMC) methods extend the idea of generating samples from one single distribution to a sequence of distributions. Good references on basic Monte Carlo methods include Kalos and Whitlock [48], and Hammersley and Handscomb [38]. Spanier and Gelbard's book [86] is an authoritative source for Monte Carlo applications to nuclear transport problems. More advanced topics on Monte Carlo methods can be found in [80]. Gilks et al. [31] is an excellent starting point for MCMC. More details on sequential Monte Carlo methods are available in [21, 59, 80]. A useful website *http://www-sigproc.eng.cam.ac.uk/smc/* is maintained at Cambridge University for SMC related research.

2.1 Monte Carlo Methods: A Brief History

The generic term "Monte Carlo method" refers to all numeric methods involving statistical sampling processes for approximate solutions to quantitative problems. It can be used not only for probabilistic problems, but also for non-probabilistic problems such as optimization and numerical integration. Application domains range from economics to nuclear physics to computer sciences.

The earliest documented example of Monte Carlo computation is an experiment done by Comte de Buffon in 1777. He performed an experiment by throwing a needle of length l at random onto a board marked with parallel lines a distance d apart to infer the probability p that the needle will intersect one of those lines.

Later, Laplace pointed out that the experiment by Comte De Buffon can be used to estimate the value of π. Suppose the needle is throw n times and M is a random variable for the number of times the needle intersects a line. Then

$$p = E(M)/n \tag{2.1}$$

where $E(M)$ is the expectation of M. It is not difficult to analytically obtain the probability p as

$$p = \frac{2l}{d\pi} \tag{2.2}$$

Connecting the above two equations and rearranging, we get a Monte Carlo estimator for π:

$$\hat{\pi} = \frac{n}{M}\frac{2l}{d} \tag{2.3}$$

In 1864, Captain O. C. Fox did three such experiments to estimate π. Interestingly, Fox improved his second experiment by rotating the ruled board between drops to eliminate the bias due to the position of dropping the needle. In his third experiment, he adjusted the values of l and d so that the needle could cross multiple lines in a single toss. In this way, he reduced the sample variance in the estimation and improved his estimation of π from 3.178 in his first attempt to 3.1416 in the third experiment with similar numbers of drops, n.

In 1873, A. Hall [36] published a paper on the experimental determination of π. Other isolated examples of using Monte Carlo methods include that Lord Kelvin used random sampling to estimate time integrals of kinetic energy in 1901, and, Student (W. S. Gosset) used random sampling to help him discover the t-distribution in 1908.

The modern history of Monte Carlo methods starts in the early 1940's when scientists at Los Alamos systematically used them as a research tool in their work on developing nuclear weapons. One of the key figures was Stanislaw Ulam [62], a Polish mathematician who worked for John von

Neumann during World War II. Ulam did not invent the statistical sampling method, which had been used to solve quantitative problems long before. However, with the development of the first electronic computer, ENIAC, Ulam was the first one to realize the potential of using computers to automate the statistical sampling process. Together with John von Neuman and Nicolas Metropolis, he developed algorithms and explored the means to convert non-random problems into random forms so that statistical sampling can be used for their solution. One of the first published papers on this topic was by Metropolis and Ulam [63] in 1949. The name "Monte Carlo" was suggested by Metropolis after the famous Monaco casino.

2.2 Estimators and their Properties

A function F of random variables is called an *estimator* for an unknown population quantity θ if its mean $E(F)$ is a usable approximation of θ. A particular numerical value of F, after instantiating the random variables with the known sample data, is called an *estimate*.

For any given quantity, there exist many possible estimators. Generally, we would like to use Monte Carlo estimators that provide good estimates in a reasonable amount of computational time. In order to choose one estimator over another, some criteria are needed. Those criteria are usually based on the following properties of an estimator: mean squared error, bias, consistency and efficiency. However, it is worth pointing out that in many cases there may not exist a clear choice among estimators, even though in some cases some estimators can be clearly better than others.

- Mean Squared Error

  The quality of an estimator is generally judged by its mean squared error. The mean squared error (MSE) of an estimator F of a quantity θ is defined as the expected value of the square of the difference between F and θ:

  $$MSE(F) = E[(F - \theta)^2] \qquad [2.4]$$

- Bias

F is called an unbiased estimator of θ if its expected value is exactly the same as the true value of θ. If not, the difference between them is the bias:

$$\beta = E[F] - \theta \quad [2.5]$$

One advantage of using an unbiased estimator is that it is guaranteed to get the correct value of θ if enough samples are used. Also, the expected value of an unbiased estimator will be the correct value after any number of samples, which makes it much easier to analyze the error of the estimator. Rewriting Equation 2.4, we have

$$\begin{aligned} MSE(F) &= E[(F-\theta)^2] \\ &= E[((F-E[F]) + (E(F)-\theta))^2] \\ &= E[(F-E[F])^2] + 2E[F-E[F]](E(F)-\theta) + (E[F]-\theta)^2 \\ &= Var[F] + 2(E[F]-E[F])(E(F)-\theta) + \beta^2 \\ &= Var[F] + \beta^2 \end{aligned} \quad [2.6]$$

If the estimator F is unbiased, then β is 0. This means that the MSE for the estimator is the same as its variance. So, in order to estimate the error for an unbiased estimator, we just need to compute the sample variance of the estimator.

On the other hand, a biased estimator may still not give a correct estimate for θ even with an infinite number of samples. The error for a biased estimator is generally more difficult to estimate than an unbiased estimator. However, in some cases, a biased estimator may have some desirable properties, such as smaller variance, over any unbiased estimator. For that and other reasons, it is sometimes preferable not to limit oneself to unbiased estimators. Generally, we seek the estimator minimizing the MSE that is a combination of both bias and variance.

- Consistency

An estimator F is called consistent for the quantity θ if and only if F converges to θ with probability 1 as the number of samples goes to infinity. That is,

$$prob\{\lim_{N\to\infty} F(X_1, \cdots, F_N) = \theta\} = 1$$

Note that the condition for consistency is stronger than simply requiring the bias, β, go to 0 as the number of samples approaches infinity. One sufficient condition for an estimator to be consistent is that both its variance and bias go to 0 as N increases. There exist situations where an unbiased estimator is not consistent, for example when its variance is infinite. For a biased estimator with finite variance, the estimator is consistent if its bias diminishes to 0 as N increases.

- Efficiency

  For any estimator, increasing computation time almost always decreases the variance, so the tradeoff is whether a decrease in $V[F]$ will more than compensate for the increase in time, $T[F]$. The efficiency of a Monte Carlo estimator is defined as the inverse of the product of the variance and the running time to reach that variance [38]:

$$\epsilon[F] = \frac{1}{V[F]T[F]}$$

2.3 Monte Carlo Integration

One important class of applications where Monte Carlo methods can help greatly is to evaluate the integration of functions or, equivalently, the expectations of functions. It is usually not difficult to formulate a quantity as an expectation and to propose a naive Monte Carlo estimator. Actually, at least in a trivial sense, every application of the Monte Carlo method can be somehow represented as a definite integral.

Suppose we want to evaluate the integral

$$I = \int_\Omega f(x)dx \qquad [2.7]$$

where domain Ω is a region in multiple-dimensional space and $f(x)$ is the integrand. The idea of Monte Carlo integration is to estimate the integral with an expected value using random samples. I can be interpreted as the expected value of random variable $f(X)$, where X is a random variable uniformly distributed in Ω. If we draw a set of samples, $X_1, \cdots, X_N$, uniformly in Ω, then an approximation to I can be obtained by its arithmetic mean:

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} f(X_i) \qquad [2.8]$$

Based on the *law of large numbers*, $\hat{I}_N$ is an unbiased estimator for I. We call $\hat{I}_N$ in Equation 2.8 the crude Monte Carlo estimator. The variance of the crude Monte Carlo estimator is

$$Var(\hat{I}_N) = Var(\frac{1}{N} \sum_{i=1}^{N} f(X_i)) = \frac{1}{N} Var(f(X)) \qquad [2.9]$$

So the standard error of $\hat{I}_N$ is $\sigma/\sqrt{N}$, where $\sigma^2 = Var(f(X))$.

Two conclusions can be drawn from the variance in Equation 2.9: (1) the standard error of the crude Monte Carlo estimator decreases with the square root of the sample size N, and (2) it does not suffer from the curse of dimensionality, i.e., the computation does not increase exponentially with the dimensionality of the integral (methods such as the Newton-Cotes rules or Simpson's method suffer from the curse of dimensionality). The statistical error is independent of the dimensionality of the integral.

While the statistical error of the crude Monte Carlo estimator remains constant in high-dimensional problems, there are two potential difficulties: (1) it may not be possible to uniformly sample an arbitrary space Ω, and (2) for a high-dimensional space, the function of interest, $f(x)$, may be 0 in most regions while having high values in some very small regions. Uniformly sampling Ω may cause the variance σ to be extremely large.

With a trivial manipulation, we can rewrite Equation 2.7 as

$$\begin{aligned} I &= \int_{\Omega} f(x)dx \\ &= \int_{\Omega} \frac{f(x)}{p(x)} p(x)dx \end{aligned} \qquad [2.10]$$

where $p(x)$ is a PDF in Ω. Instead of uniformly sampling Ω, we can generate N samples $X_1, \cdots, X_N$ from $p(x)$ and compute the following estimator

$$\hat{I}_p = \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i)}{p(X_i)} \qquad [2.11]$$

It is easy to see that $\hat{I}_p$ is an unbiased estimation of I and the variance of $\hat{I}_p$ is σ_p^2/N, where

$$\begin{aligned} \sigma_p^2 &= \int (\frac{f(x)}{p(x)} - I)^2 p(x) dx \\ &= \int \frac{f^2(x)}{p(x)} dx - I^2 \end{aligned} \qquad [2.12]$$

The crude Monte Carlo estimator is a special case of the estimator $\hat{I}_p$ if $p(x)$ is chosen to be a uniform distribution function in Ω. The estimator $\hat{I}_p$ has the same two properties as the crude Monte Carlo estimator: the statistical error decreases with $\sqrt{N}$ and it is not affected by the dimension of the sample space Ω.

One major advantage of Monte Carlo methods for integration is that it is easy to understand and simple to use. The only thing needed is a density function, $p(x)$, from which we can generate samples, and the ability to evaluate the sample weights $\frac{f(X_i)}{p(X_i)}, i = 1, \cdots, N$. Another advantage of Monte Carlo methods is flexibility – they can be applied to a wide range of problems. In situations like high-dimensional integration, Monte Carlo methods may be the only feasible solution. For example, the problem of global illumination in computer graphics must evaluate the integral over the space of all light paths. Consequently, its domain has infinite dimension, but Monte Carlo methods provide a natural way of handling it.

2.4 Variance Reduction Techniques

The major disadvantage of Monte Carlo methods for integration is its RMS error converges at a relatively slow rate of $O(N^{-1/2})$, which means that we need to quadruple the number of samples in order to reduce the standard deviation by half.

In order to speed up Monte Carlo simulation, users need to use techniques for variance reduction. Even in early applications of Monte Carlo at Los Alamos, von Neumann and Ulam refined their simulations with some variance reduction techniques such as Russian Roulette and splitting.

The variance reduction methods commonly used include importance sampling, control variates, and stratified sampling. These and other more advanced variance reduction techniques are discussed in the next sections.

2.4.1 Importance Sampling

Obviously, the variance, $Var(\hat{I}_p)$, of the Monte Carlo estimator $\hat{I}_p$ in Equation 2.12 depends on the choice of the density function $p(x)$ from which we draw the samples. Intelligently choosing $p(x)$ to reduce the variance of the Monte Carlo estimator is called *importance sampling*. $p(x)$ is called the *importance density*. For each sample $X^{(k)}$ from $p(x)$, $f(X^{(k)})/p(X^{(k)})$ is called the *importance weight*. The only two constraints for using importance sampling are: (1) it is possible to generate samples from $p(x)$, and (2) we must be able to evaluate the importance weights.

Equation 2.12 suggests that more samples should be put in the "important" regions in the sample space, where $f(x)$ has relatively high values. This is very important especially for high-dimensional problems since the target function, $f(x)$, could have nonzero values in only a very small portion of the whole sample space. Uniformly sampling the whole sample space is doomed to fail in these simulations.

The optimal density function $p^*(x)$ that minimizes the asymptotic variance is $c|f(x)|$, where the c is the constant term $c = 1/\int f(x)dx$. $p^*(x)$ leads to zero variance. Unfortunately, using an optimal density function is not practical because it requires knowledge of the normalization constant, c, which involves the value of the desired integral, I. However, it suggests a good importance sampling density should have a shape similar to $|f(x)|$. Typically, a function $g(x)$ may be obtained by using some factors of $f(x)$ or approximating $f(x)$ with the major components in its Taylor expansion. If $g(x)$ obtained that way is possible to generate samples from, then we can set $p(x) \propto g(x)$.

2.4.2 Control Variates

Another important technique for variance reduction is *control variates* [48]. The basic idea of control variates is to replace the evaluation of an unknown expectation with the evaluation of the

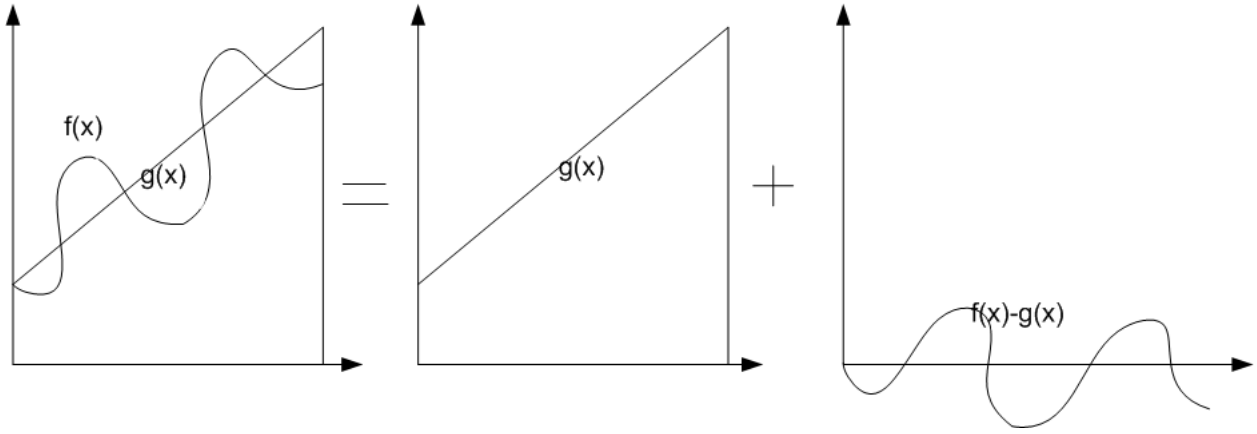

Figure 2.1: Control variates.

difference between the unknown quantity and another expectation whose value can be integrated analytically.

Assuming we are interested in the integral in Equation 2.11 and we can find a function $g(x)$ that can be integrated analytically and has the following property:

$$Var(f(x) - g(x)) \leq Var(f(x)) \qquad [2.13]$$

then an estimator of the form

$$F = \int g(x)dx + \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i) - g(X_i)}{p(X_i)} \qquad [2.14]$$

will have a lower variance than $\hat{I}_p$ in Equation 2.11.

Generally, a good choice of control variate for a function, $f(x)$, is the sum of the first several terms of its Taylor series. For example, Kalos and Whitlock [48, pg. 108] showed that by using the first two terms of the Taylor series of $\exp(x)$, $1 + x$, as the control variate, the Monte Carlo variance was reduced from 0.242 to 0.043 with the same uniform samples in (0,1).

If we have a function $g(x)$ which is an approximation of $f(x)$, $g(x)$ may be good as either a control variate or importance sampling density. In general, if $f(x) - g(x)$ is approximately a constant (absolutely uniform), using $g(x)$ as a control variate in correlated sampling is more efficient than using importance sampling. On the other hand, if $f(x)/g(x)$ is nearly a constant (relatively uniform), it would be more appropriate to use $g(x)$ as importance density in importance

sampling [37]. Furthermore, $g(x)$ should be integrable analytically to be used as a control variate, while $g(x)$ has to be easy to sample to be used as an importance sampling density.

2.4.3 Defensive Importance Sampling

A common pitfall of importance sampling is that importance sampling can fail if the target function, $f(x)$, has a heavier tail than the importance sampling density function, $p(x)$, even though $p(x)$ might have roughly the same shape as $f(x)$. In that situation, when a sample is generated from the far tails of $p(x)$, the importance weight for that sample may be orders of magnitude larger than the typical values for the samples at modes. This will cause very high variance in the estimate. In the extreme case that $p(x)$ decreases towards 0 faster than $f^2(x)$ as x moves towards its tails, the variance will be ∞.

Defensive importance sampling (DIS) [41, 42] is a technique that fixes the above problem with importance sampling. Assume we want to compute the integral

$$I = \int_{\Omega} f(x)q(x)dx \qquad [2.15]$$

where $q(x)$ is a target density function on Ω. Let $p(x)$ to be a probability density function that is close to the optimal importance sampling density, $|f(x)|q(x)/I$. Instead of using $p(x)$ alone as the importance density function, defensive importance sampling uses a *defensive mixture distribution* that has the form

$$p_\alpha(x) = \alpha q(x) + (1-\alpha)p(x) \qquad [2.16]$$

where $0 < \alpha < 1.0$.

Using a defensive mixture distribution makes the sample weight $q(x)/p_\alpha(x)$ bounded by $1/\alpha$. It also guarantees the variance of defensive importance sampling is less than or equal to $1/\alpha$ times the variance of the simple Monte Carlo estimate using a uniform distribution.

If we can not use the target distribution, $q(x)$, in a defensive mixture because either it is unknown or it is difficult to sample from, then a mixture distribution with more than two components can be used so that all the important regions in the sample space will be represented. For example, if $q(x)$ can be decomposed into a product of several density functions, $q_1(x), \cdots, q_n(x)$, and each

PDF is easy to sample from, then we can use a mixture distribution of the general form

$$p_\alpha(x) = \sum_{k=1}^{n} \alpha_k q_k(x) + \alpha_0 p(x) \qquad [2.17]$$

where $\sum_{k=0}^{n} \alpha_k = 1.0$ and $\alpha_k > 0$.

2.4.4 Mixture Sampling

Mixture sampling as defined by Owen and Zhou [69] combines importance sampling with control variates in a way that uses a mixture density for importance sampling while employing the mixture components as control variates.

Suppose we have m different PDFs, $p_1(x), \cdots, p_m(x)$, and we can construct a mixture density, $p_\alpha(x) = \sum_{i=1}^{m} \alpha_i p_i(x)$, where $\alpha_i > 0$ and $\sum_{i=1}^{m} \alpha_i = 1$. As described by Owen and Zhou, the mixture components $p_i(x)$ can also be used as control variates. If we generate n samples, $X_1, \cdots, X_n$, from $p_\alpha(x)$, the estimator that results for the integral $I = \int f(x)dx$ using mixture sampling is

$$\tilde{I}_{\alpha,\beta} = \frac{1}{n} \sum_{j=1}^{n} \frac{f(X_j) - \sum_{i=1}^{m} \beta_i p_i(X_j)}{p_\alpha(X_j)} + \sum_{i=1}^{m} \beta_i \qquad [2.18]$$

where the β_i are a set of real-valued variables. This estimator is unbiased, and its variance is

$$\sigma^2_{\alpha,\beta} = \int \left(\frac{f(x) - \sum_{i=1}^{m} \beta_i p_i(x)}{p_\alpha(x)} - I + \sum_{i=1}^{m} \beta_i \right)^2 p_\alpha(x)dx \qquad [2.19]$$

If β^*, the optimal set of β_i which minimizes $\sigma^2_{\alpha,\beta}$, is used, then Owen and Zhou showed that $\sigma^2_{\alpha,\beta^*} \leq \min_{i=1}^{m} \alpha_i^{-1} \sigma^2_{p_i}$. In other words, using n samples from the mixture with the control variate estimate is no worse than drawing $n\alpha_j$ samples from the *best* component of the mixture.

We do not know β^*, but we can obtain an estimate, $\hat{\beta}$, by multiple regression of $f(X_j)/p_\alpha(X_j)$ on predictors $p_i(X_j)/p_\alpha(X_j)$. With this method, $\hat{\beta}_i = \beta^*_i + O_{p_\alpha}(n^{-1/2})$ for $i = 1, \cdots, m$, and $\tilde{I}_{\alpha,\hat{\beta}} = \tilde{I}_{\alpha,\beta^*} + O_{p_\alpha}(n^{-1})$.

In practice, deterministic mixture sampling (DMS) is preferred over ordinary mixture sampling because DMS has provably smaller variance. In DMS, the number of samples from each component, $p_i(x)$, is allocated deterministically as $n_i = n\alpha_i$, where n is the total number of samples. From $p_i(x)$, we generate n_i independent samples, X_{ij}, $i = 1, \cdots, m$ and $j = 1, \cdots, n_i$. Then the

estimator is

$$\hat{I}_{\alpha,\beta} = \frac{1}{n}\left(\sum_{i=1}^{m}\sum_{j=1}^{n_i}\frac{f(X_{ij}) - \sum_{i=1}^{m}\beta_i p_i(X_{ij})}{p_\alpha(X_{ij})}\right) + \sum_{i=1}^{m}\beta_i \quad [2.20]$$

2.4.5 Multiple Importance Sampling

For a target distribution with multiple modes, sampling according to a single importance density may not be able to capture all the important regions of the integrand. Instead, several PDF's may be constructed and each of them can generate samples for some specific, important regions.

For the estimation of the integral in Equation 2.7, suppose we have n different PDFs, $p_1(x), \cdots, p_n(x)$, and generate n_i samples $\{X_{i,1}, \cdots, X_{i,n_i}\}$ from $p_i(x)$. The question is how to combine those samples in a manner that minimizes the estimation variance without introducing bias. Simply averaging those samples generally will not produce an optimal result.

Veach and Guibas [102] introduced multiple importance sampling (MIS) in the context of global illumination and studied the above problem. To use all the samples, $\{X_{i,j}\}, 1 \leq i \leq n, 1 \leq j \leq n_i$, to estimate the desired integral, a *multiple-sample estimator* is defined as

$$F = \sum_{i=1}^{n}\frac{1}{n_i}\sum_{j=1}^{n_i} w_i(X_{i,j})\frac{f(X_{i,j})}{p_x(X_{i,j})} \quad [2.21]$$

where the weighting functions, $w_1, \cdots, w_n$, give the weight, $w_i(x)$, for each sample x drawn from p_i. In order for the multiple-sample estimator to be unbiased, the weighting functions should satisfy $\sum_{i=1}^{n} w_i(x) = 1$ and $w_i(x) \geq 0$.

One obvious choice for the weighting functions is to use

$$\hat{w}_i(x) = c_i\frac{p_i(x)}{q(x)} \quad [2.22]$$

where

$$q(x) = c_1 p_1(x) + \cdots + c_k p_k(x) \quad [2.23]$$

$c_i \geq 0$ and $\sum_i c_i = 1$. It is "obvious" in the sense that if we sample according to the mixture PDF in Equation 2.23, a classical importance sampling estimator will give the same estimation as the multiple importance sampling estimator with the above weighting functions.

If c_i is set in proportional to the number of samples from each PDF, $c_i = n_i / \sum_i n_i$, it leads to

$$\hat{w}_i(x) = \frac{n_i p_i(x)}{\sum_k n_k p_k(x)} \quad [2.24]$$

This weighting strategy is called the balance heuristic. Balance heuristic weighting is nearly optimal, which means no other combination is much better. In Appendix A, we show that MIS with balance heuristic weighting can be treated as a special case of defensive importance sampling.

2.4.6 Stratified Sampling

The basic idea of stratified sampling is to divide the full integration space into mutually exclusive subspaces (strata), and then perform Monte Carlo integration in each subspace. Suppose we are interested in estimating $I = \int_\Omega f(x)dx$, and we have m disjoint subspaces $\Omega_1, \cdots, \Omega_m$, whose union is Ω. If we generate n_i samples, $X_{i,1}, \cdots, X_{i,n_i}$, from subspace Ω_i, where $i = 1, \cdots, m$, then the estimator from stratified sampling

$$\hat{I} = \sum_{i=1}^{m} \frac{1}{n_i} \sum_{j=1}^{n_i} f(X_{ij}) \quad [2.25]$$

is an unbiased estimator for I with variance

$$var(\hat{I}) = \sum_{i=1}^{m} \frac{\sigma_i}{n_i}$$

where σ_i is the variance of $f(x)$ in subspace Ω_i.

It can be shown that stratified sampling will never have higher variance than plain unstratified sampling [100]. Stratified sampling techniques are very useful when the population is heterogeneous but certain homogeneous sub-populations can be separated into subgroups. However, stratified sampling does not scale well to high-dimensional integration because there are too many dimensions to refine.

2.4.7 Adaptive Sampling

Adaptive sampling strategies allow for adjusting sampling pattern depending upon observations made during the sampling process [96]. Similar to importance sampling, adaptive sampling

puts more samples in the more important regions for the integral. However, one major difference between adaptive sampling and importance sampling is that the distribution for adaptive sampling is modified "on the fly" to learn from the performance of previous samples, while the distribution for importance sampling is set *a priori* before sampling starts.

There are number of applications of adaptive sampling in computer graphics for photo-realistic rendering [65, 70, 75, 95, 6, 29]. The goal of those algorithms is to concentrate samples where they will affect the quality of the rendered image most. There are three central issues for adaptive sampling algorithms: refinement criteria, how to avoid bias, and how to generate samples from the refinement distribution so that they reach the areas most in need. Much of the work in the rendering literature has been done on developing different refinement criteria.

The main disadvantage of adaptive sampling is that it can introduce bias if not used with care [51]. Bias can be avoided using two-stage sampling. A better solution is to put adaptive sampling into an importance sampling framework [8, 80]. Much of this thesis work is to develop unbiased adaptive rendering algorithms that reuse the samples to account for spatial and temporal coherence in the rendering.

2.5 MCMC and Metropolis-Hastings Sampling

Markov chain Monte Carlo (MCMC) methods use Markov chain simulation to sample a specified target distribution [31]. Given the state space Ω and a target distribution $\pi(x)$, where $x \in \Omega$, the MCMC algorithm generates a random walk, $X_0, X_1, X_2, \cdots$, from a distribution $K(X_{t+1}|X_t)$ which depends on the current state of the chain, X_t. The conditional probability density $K(\cdot|\cdot)$ is called the *transition kernel* of the chain. If the chain has $\pi(x)$ as a stationary distribution, then after a large number of iterations (*burn-in* phase), the chain will be a sequence of dependent samples approximately from $\pi(x)$. From a Monte Carlo point of view, $\pi(x)$ can be represented by those samples after burn-in, which means that any computation of expectations (or integrals) using π can be estimated to an acceptable degree of accuracy by using those dependent samples in the Markov chain.

Now the problem is how to construct a Markov chain such that its stationary distribution is exactly the target distribution, $\pi(x)$. This turns out to be surprisingly easy. The Metropolis-Hastings algorithm [64, 39] provides a way of generating such a Markov chain. This is done by first proposing a *candidate* state, X'_{t+1}, using information from X_t. The algorithm then either accepts the candidate, X'_{t+1}, or rejects it and retains X_t. In pseudo-code:

Initialize X_0; set $t = 0$.
for $t = 1$ to N
  $X'_{t+1} \leftarrow T(\cdot|X_t)$
  generate a random number $r \in [0, 1]$
  if($r < \alpha(X'_{t+1}|X_t)$) then
    $X_{t+1} = X'_{t+1}$
  else
    $X_{t+1} = X_t$

Figure 2.2: The Metropolis sampling algorithm.

The function $\alpha(X'_{t+1}|X_t)$ computes the *acceptance probability* for X'_{t+1} given the current sample X_t. It is computed as

$$\alpha(X'_{t+1}|X_t) = min\{1, \frac{\pi(X'_{t+1})T(X_t|X'_{t+1})}{\pi(X_t)T(X'_{t+1}|X_t)}\} \qquad [2.26]$$

where $T(X'_{t+1}|X_t)$ is the *proposal distribution* (or tentative transitional function), denoting the probability density of going to state X'_{t+1} given that the current state is X_t. Remarkably, $T(X'_{t+1}|X_t)$ can have almost any form and the chain generated by the Metropolis-Hastings algorithm will still have stationary distribution $\pi(x)$. This can be seen from the following argument. The transition kernel for the Metropolis-Hastings algorithm is

$$\begin{aligned} K(X_{t+1}|X_t) &= T(X_{t+1}|X_t)\alpha(X_t|X_{t+1}) \\ &+ \delta(X_{t+1} = X_t)\left(1 - \int T(Y|X_t)\alpha(X_t|Y)dY\right) \end{aligned} \qquad [2.27]$$

where $\delta(\cdot)$ is the indicator function, so $K(X_{t+1}|X_t)$ and $K(X_t|X_{t+1})$ will have the same second part no matter whether $X_{t+1} = X_t$ or not. Based on how we compute the acceptance probability, we have

$$\pi(X_t)T(X_{t+1}|X_t)\alpha(X_t|X_{t+1}) = \pi(X_{t+1})T(X_t|X_{t+1})\alpha(X_{t+1}|X_t) \quad [2.28]$$

From Eqs. [2.27] and [2.28], we obtain

$$\pi(X_t)K(X_{t+1}|X_t) = \pi(X_{t+1})K(X_t|X_{t+1}) \quad [2.29]$$

which is called the *detailed balance* equation. Detailed balance is a sufficient condition for guaranteeing that $\pi(\cdot)$ is the stationary distribution of $K(\cdot|\cdot)$.

Metropolis-Hastings sampling is very general. It can be used to sample an arbitrary, complex probability distribution function, $\pi(x)$, known up to a normalizing constant, as long as $\pi(x)$ can be evaluated. The proposal distribution, $T(X_{t+1}|X_t)$, can have almost any form and the chain will still eventually converge to $\pi(x)$. However, the relationship between $T(X_{t+1}|X_t)$ and $\pi(x)$ has a significant impact on the convergence rate of the chain, and hence the number of samples required to get a good result for integral estimation. The key to designing a good MCMC sampler is designing good proposal distributions.

2.6 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods are a set of sampling techniques that generate samples from a sequence of probability distribution functions [22]. SMC methods are very flexible, easy to implement, parallelizable, and applicable in general settings.

There are a variety of ways to do SMC sampling, with two approaches being most applicable to graphics [66, 59, 60]: in an importance sampling context, the sample can be re-used and re-weighted, resulting in sampling importance resampling (SIR); or the procedure can be framed in both an importance sampling and a Markov chain Monte Carlo context, which leads a population Monte Carlo framework [8].

2.6.1 Sampling Importance Resampling (SIR)

Assume we want to estimate the integral

$$\pi(f) = \int_{\Omega} f(x)\pi(x)dx \qquad [2.30]$$

The sampling importance resampling method [81, 58] extends importance sampling to achieve simulation from the target distribution by resampling. SIR proceeds in two stages. The first stage draws some independent samples from a proposal distribution, $p(x)$. For each sample, the importance weight is computed. The second stage generates samples by resampling those samples in the first stage based on their importance weights. The algorithm is outlined below:

---

1 Generate N independent samples $\{\tilde{x}_1, \cdots, \tilde{x}_M\}$ from a proposal distribution $p(x)$

2 Compute importance weights $w_i = \pi(x_i)/p(x_i), i = 1, \cdots, M$

3 Generate M samples $\{x_1, \cdots, x_N\}$ by resampling $\{\tilde{x}_1, \cdots, \tilde{x}_M\}$ with replacement according to probability proportional to their weights

---

Figure 2.3: SIR algorithm.

The SIR estimator of $\pi(f)$ is constructed as

$$\hat{\pi}(f) = \frac{1}{N}\sum_{i=1}^{N} f(x_i) \qquad [2.31]$$

which converges to $\pi(f)$ since each x_i is approximately distributed from $\pi(x)$. As with importance sampling, the efficiency of SIR strongly depends on the choice of the proposal distribution, $p(x)$.

The resampling scheme used above is multinomial resampling [34]. Other resampling algorithms are also available such as stratified resampling, which is optimal in terms of variance [52], and minimum entropy resampling [15].

2.6.2 Population Monte Carlo (PMC)

The population Monte Carlo algorithm [8] is an iterated importance sampling scheme. In this scheme, a sample population approximately distributed according to a target distribution is

generated at each iteration. Then the samples from all the iterations can be used to form unbiased estimates of integrals under that distribution. It is an adaptive algorithm that calibrates the proposal distribution to the target distribution at each iteration by learning from the performance of the previous proposal distributions.

Assume we are interested in estimating the integral $\pi(f) = \int_\Omega f(x)\pi(x)dx$. We wish to sample according to the target distribution, $\pi(x)$. The generic PMC sampling algorithm is given in Figure 2.4.

---

1 generate the initial population, $t = 0$

2 **for** $t = 1, \cdots, T$

3 adapt $K^{(t)}(x^{(t)}|x^{(t-1)})$

4 **for** $i = 1, \cdots, n$

5 generate $\hat{X}_i^{(t)} \sim K^{(t)}(x|X_i^{(t-1)})$

6 $w_i^{(t)} = \pi(\hat{X}_i^{(t)})/K^{(t)}(\hat{X}_i^{(t)}|X_i^{(t-1)})$

7 resample according to $w_i^{(t)}$ for the new population $X_i^{(t)}$

Figure 2.4: The generic population Monte Carlo algorithm.

Line 1 creates the initial population to jump-start the algorithm. Any method can be used to generate these samples provided that any sample with non-zero probability under f can be generated, and the probability of doing so is known.

The outer loop is over iterations. In each iteration of the algorithm, a *kernel function*, $K^{(t)}(x^{(t)}|x^{(t-1)})$, is determined (line 3) using information from the previous iterations. The kernel function is responsible for generating the new population, given the current one. It takes an existing sample, $X_i^{(t-1)}$, as input and produces a candidate new sample, $\hat{X}_i^{(t)}$, as output (line 5). The distinguishing feature of PMC is that the kernel functions are modified after each step based on information gathered from prior iterations. The kernels adapt to approximate the ideal importance function based on the samples seen so far. While this dependent sampling may appear to introduce bias, it can be proven that the result is either unbiased or consistent, depending on whether

certain normalizing constants for π and the kernels are known. In our applications to the rendering problem, π is always 1 and kernels are always PDFs.

The weight computed for each sample, $w_i^{(t)}$, is essentially its importance weight. The resampling step in line 7 is designed to cull candidate samples with low weights and promote high-weight samples. It takes the candidate population, $\left\{\hat{X}_1^{(t)}, \ldots, \hat{X}_n^{(t)}\right\}$, and produces a new population ready for the next iteration. Resampling is not always necessary, particularly if the kernel is not really a conditional distribution. Even when used, resampling and kernel adaptation (lines 3 and 7) need not be done on every iteration. Our examples demonstrate such cases. Figure 2.5 shows the sampling and resampling steps in SMC algorithm for the example illustrated in Figure 1.3.

At any given iteration, t, a PMC estimator given by

$$\tilde{\pi}(f) = \frac{1}{n}\sum_{i=1}^{n} w_i^{(t)} f(X_i^{(t)}) \qquad [2.32]$$

is unbiased for $\pi(f)$. To see that, we have

$$\begin{aligned} E[w_i^{(t)} f(X_i^{(t)})] &= \int\int \frac{\pi(x)}{K_{it}(x|\zeta)} f(x) K_{it}(x) dx\, g(\zeta) d\zeta \\ &= \int\int f(x)\pi(x) dx\, g(\zeta) d\zeta \\ &= \int f(x)\pi(x) dx = \pi(f) \end{aligned} \qquad [2.33]$$

where ζ is the vector of past random variates which contribute to K_{it} and $g(\zeta)$ is an arbitrary distribution. It concludes that $\tilde{\pi}(f)$ is an unbiased estimator of $\pi(f)$.

However, in most settings, $\pi(x)$ may be known only up to a constant scale factor. Then, an estimator with a self-normalized term has to be used:

$$\hat{\pi}(f) = (\sum_{i=1}^{n} w_i^{(t)})^{-1} \sum_{i=1}^{n} w_i^{(t)} f(X_i^{(t)}) \qquad [2.34]$$

In this case, the unbiasedness property of the estimator is lost, but $\hat{\pi}(f)$ is consistent.

In practice, we can average over all iterations to improve the estimate. A cumulative self-normalized PMC estimator over all T iterations can be defined as

$$\hat{\pi}_\beta(f) = \sum_{t=0}^{T} \beta_t \left((\sum_{i=1}^{n} w_i^{(t)})^{-1} \sum_{i=1}^{n} w_i^{(t)} f(X_i^{(t)}) \right) \qquad [2.35]$$

where $\beta_t, t = 0, \cdots, T$, are the weights to combine the estimates from different iterations. The optimal choices of β_t, which minimize the variance of $\hat{\pi}_\beta(f)$, are given by [20]:

$$\beta_t^{min} = \sigma_t^{-2} / (\sum_{t=0}^{T} \sigma_t^{-2})$$

where σ_t is the variance of the estimator $\hat{\pi}(f)$ at iteration t.

In our work, we introduce the population Monte Carlo method to computer graphics and apply it to adapting and reusing samples in the global illumination context.

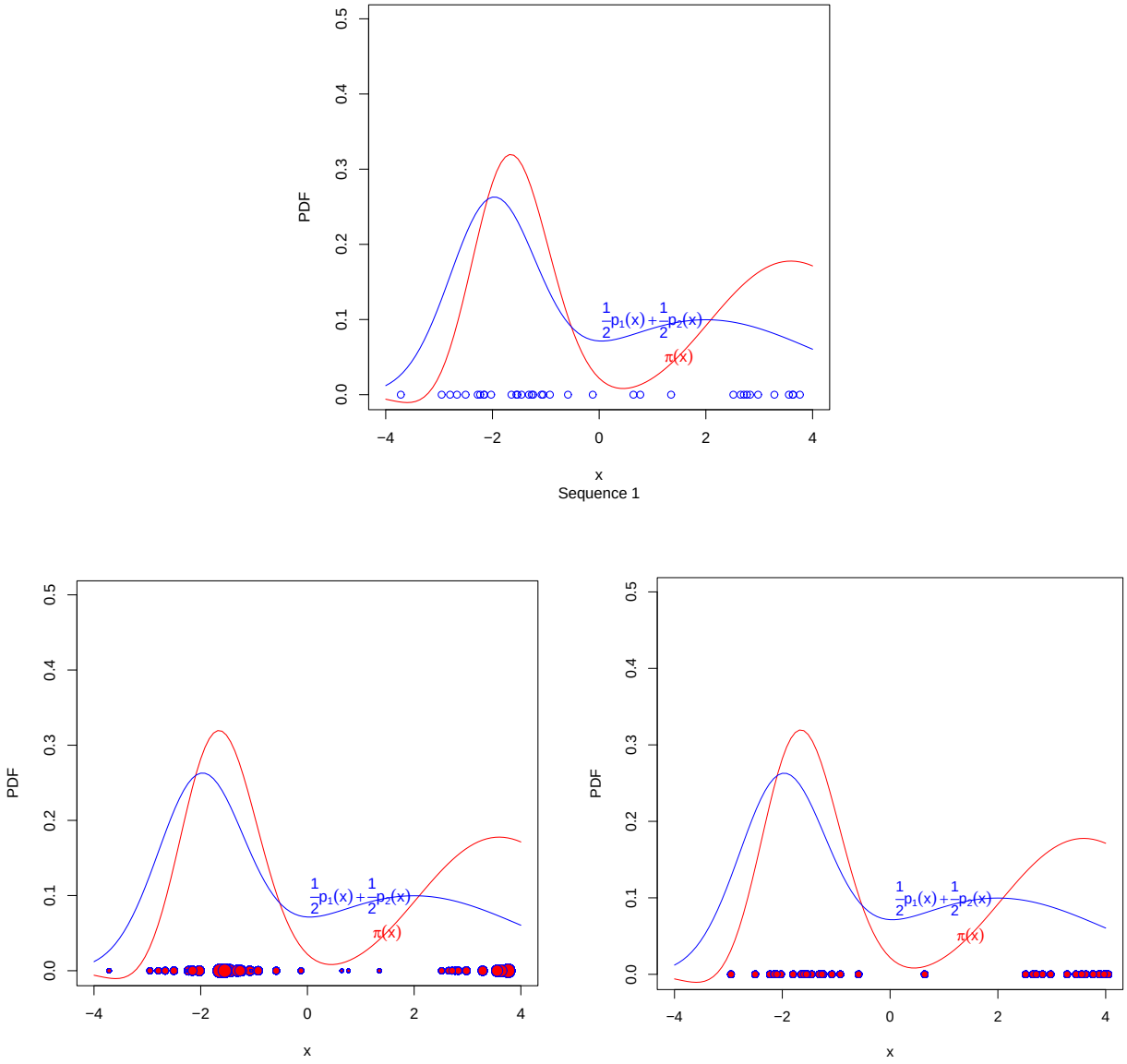

Figure 2.5: PMC sampling and resampling steps. The top diagram shows the initial samples from the proposal distribution $\frac{1}{2}p_1(x) + \frac{1}{2}p_2(x)$. The bottom-left shows the weights for the initial samples. The bottom-right shows the samples after resampling based on the weights.

Chapter 3

Global Illumination

This chapter provides a background for global illumination. To render photo-realistic images, we must simulate the transport of light, starting from light sources, interacting with surfaces in the scene, and finally reaching the camera. In this chapter the physical quantities and equations used for light transport and global illumination computation are described. The rendering equation provides the mathematical foundation for the global illumination problem. The path integral formation for the rendering equation makes it convenient for applying Monte Carlo methods to solve this equation and generate photo-realistic images. Some representative Monte Carlo based algorithms for global illumination in the literature are summarized. The strengths and weaknesses of those algorithms are discussed.

3.1 Radiometry

Flux Φ: The total energy passing through a surface per second. Flux is also known as power, and is measured in Watts (joules/second). A light source is, by definition, something that emits power in the visible spectrum. So, for example, we can say a light bulb is 60 Watts.

Irradiance E: Flux per unit surface area. Its unit is (W/m^2).

$$E = \frac{d\Phi}{dA} \qquad [3.1]$$

For an un-occluded point light source, the irradiance at a surface point, $\vec{p}$, that is distance r away from the point light, is $\Phi/(4\pi r^2)$. This fact explains why surfaces far from a point light source are darker: the irradiance falls off with the squared distance from the light.

Intensity I: Flux per solid angle. Its unit is (W/sr).

$$I = \frac{d\Phi}{d\omega} \quad [3.2]$$

Radiance L: The flux per unit projected area per unit solid angle. It has unit of $(W/m^2/sterd)$.

$$L = \frac{d\Phi}{dA^{\perp}d\omega} \quad [3.3]$$

Radiance does not attenuate with distance due to its "per unit solid angle" definition. For example, we have the same impression of the brightness of a wall, regardless of whether we are near or far from it.

Radiance is the most important quantity to be measured in radiometry. In particular, it is the quantity required for quantitatively analyzing directional effects such as bidirectional reflectance. Radiance is also the most frequently used term in computer graphics. There are two major reasons for this. First, all the other terms can be derived from radiance. Integrating radiance over solid angle gives irradiance; integrating radiance over area gives intensity; and integrating radiance over both solid angle and area gives flux. Second, radiance remains constant along a ray in free space, so it is very convenient to use in rendering algorithms such as ray tracing.

3.2 BRDF Function

In photo-realistic rendering, how light interacts with surfaces in the scene is essential for simulating light transport. The Bidirectional Reflectance Distribution Function (BRDF) describes how much light is reflected when it hits a material.

The BRDF is defined as the ratio of the outgoing radiance in the direction ω_o to the incoming differential irradiance from the direction ω_i [67]. It is a function of incoming direction, outgoing direction and surface point $\vec{p}$:

$$f_r(\vec{p}, \omega_o, \omega_i) = \frac{dL_o(\vec{p}, \omega_o)}{dE_i(\vec{p}, \omega_i)} = \frac{dL_o(\vec{p}, \omega_o)}{L_i(\vec{p}, \omega_i)\cos\theta_i d\omega_i} \quad [3.4]$$

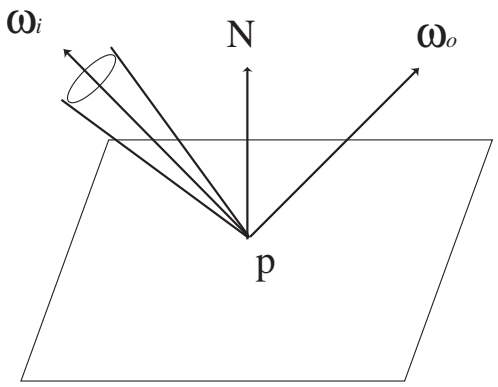

Figure 3.1: Bidirectional Reflectance Distribution Function.

Note that even though the BRDF is defined as a ratio, it is not unitless. The units of the BRDF are inverse solid-angle sr^{-1}. To understand why the BRDF is defined this way, let us look at what the BRDF is used for: to compute the radiance leaving from a surface point $\vec{p}$ along the outgoing direction ω_o, $L(\vec{p}, \omega_o)$, which is the sum of reflected radiance of the incoming radiance from all the directions in the hemisphere. Denote the BRDF term as $ratio(\vec{p}, \omega_o, \omega_i)$ for a moment. Then,

$$L_o(\vec{p}, \omega_o) = \sum_{Hemishpere} L_i(\vec{p}, \omega_i) * ratio(\vec{p}, \omega_o, \omega_i)\Delta\omega_i \qquad [3.5]$$

One obvious choice is to define $ratio(\vec{p}, \omega_o, \omega_i)$ as a ratio of radiances $\frac{\Delta L_o(\vec{p},\omega_o)}{\Delta L_i(\vec{p},\omega_i)}$, however, then $L_i(\vec{p}, \omega_i) * ratio(\vec{p}, \omega_o, \omega_i)$ is a radiance, which leads the right side of Equation 3.5 to be an irradiance due to the sum while the left side is a radiance. So, in the BRDF definition, we have to cancel out the $\Delta\omega_i$ term by using $\frac{\Delta L_o(\vec{p},\omega_o)}{\Delta L_i(p,\omega_i)\cos\theta\Delta\omega_i}$.

There are two major properties of the BRDF. Models that have these properties are considered to be physically plausible.

- Helmholtz Reciprocity Rule

  For any incoming and outgoing direction pair, ω_i and ω_o, the BRDF is symmetric to the directions:

  $$f_r(\vec{p}, \omega_o, \omega_i) = f_r(\vec{p}, \omega_i, \omega_o) \qquad [3.6]$$

- Energy Conservation Law

The energy conservation law says that the quantity of light reflected must be less than or equal to the quantity of incident light. For any direction ω_o,

$$\int_{2\pi} f_r(\vec{p}, \omega_o, \omega') \cos\theta' d\omega' \leq 1 \quad [3.7]$$

3.3 The Rendering Equation

The goal of a global illumination algorithm is to generate photo-realistic images by taking into account all the light interactions in the scene. It does so by simulating light transport behaviors based on physical principles. Mathematically, the solution to a global illumination problem is the same as a solution to the rendering equation [47].

To understand the rendering equation, rewrite the definition of the BRDF given in Equation 3.4 as

$$dL_o(\vec{p}, \omega_o) = f_r(\vec{p}, \omega_o, \omega_i) L_i(\vec{p}, \omega_i) \cos\theta_i d\omega_i \quad [3.8]$$

If we integrate the incoming radiance over the hemisphere of incoming directions centered at $\vec{p}$, the outgoing reflected radiance is given by the *reflection equation*:

$$L_r(\vec{p}, \omega_o) = \int_{\Omega_i} f_r(\vec{p}, \omega_o, \omega_i) L_i(\vec{p}, \omega_i) \cos\theta_i d\omega_i \quad [3.9]$$

From the law of energy conservation, the exitant radiance at a surface point along an outgoing direction must be equal to the sum of the emitted and reflected radiances. This gives the *energy balance equation*:

$$L_o(\vec{p}, \omega_o) = L_e(\vec{p}, \omega_o) + L_r(\vec{p}, \omega_o) \quad [3.10]$$

Plugging the reflection equation into the energy balance equation results in

$$L_o(\vec{p}, \omega_o) = L_e(\vec{p}, \omega_o) + \int_{\Omega_i} f_r(\vec{p}, \omega_o, \omega_i) L_i(\vec{p}, \omega_i) \cos\theta_i d\omega_i \quad [3.11]$$

In free space, radiance along a ray is constant. If we define a ray-casting function $\vec{p}' = t(\vec{p}, \omega)$, where $\vec{x}'$ is the first surface point visible from $\vec{p}$ along the direction ω, then the incident radiance and outgoing radiance can be connected by

$$L_i(\vec{p}, \omega_i) = L_o(t(\vec{p}, \omega_i), -\omega_i) \quad [3.12]$$

Rewriting Equation 3.11 and dropping the subscript o for brevity, we obtain the *rendering equation*:

$$L(\vec{p},\omega_o) = L_e(\vec{p},\omega_o) + \int_{\Omega_i} f_r(\vec{p},\omega_o,\omega_i) L(t(\vec{p},\omega_i),-\omega_i)\cos\theta_i d\omega_i \qquad [3.13]$$

3.4 Monte Carlo Methods for the Rendering Equation

The rendering equation is a form of the Fredholm equation of the second kind in which we need to solve the unknown quantity L appearing on the both sides of the equation. The use of Monte Carlo methods to solve this kind of problem can be traced back decades in the statistical literature [86, 82].

3.4.1 Path Integral Formulation for the Rendering Equation

To apply Monte Carlo to solve the rendering equation for global illumination, it is more convenient to convert the integral over the hemisphere into an integral over surfaces. As a result, the rendering equation can be expressed as an integral in path space in which each path is a sequence of surface vertices of any possible length. We follow the path integral framework introduced by Veach [100]. The rendering equation in path integral form is

$$I = \int_{\Omega} f(\bar{x}) d\mu(\bar{x}) \qquad [3.14]$$

The integral is over Ω, the set of light transport paths that begin at a light source and end at the eye, where $\mu(\overline{\mathbf{x}})$ is the surface area measure for the path $\overline{\mathbf{x}}$, and $f(\overline{\mathbf{x}})$ is defined as

$$\begin{aligned} f(\overline{\mathbf{x}}) &= W(\overline{\mathbf{x}}) L_e(\mathbf{x}_0,\mathbf{x}_1) G(\mathbf{x}_0,\mathbf{x}_1) \\ &\quad \cdot \prod_{i=1}^{m-1} f_r(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1}) G(\mathbf{x}_i,\mathbf{x}_{i+1}) \end{aligned} \qquad [3.15]$$

in which the function $W(\overline{\mathbf{x}})$ takes the value 1 if the path passes through the image plane, and 0 otherwise. $\mathbf{x}_i$ is a point on the path $\overline{\mathbf{x}}$, $L_e(\mathbf{x}_0,\mathbf{x}_1)$ is the radiance emitted by a light point $\mathbf{x}_0$ toward $\mathbf{x}_1$, $f_r(\mathbf{x}_{i-1},\mathbf{x}_i,\mathbf{x}_{i+1})$ is the BRDF for surface point $\mathbf{x}_i$, and $G(\mathbf{x}_i,\mathbf{x}_{i+1})$ is the geometry

term between points $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$:

$$G(\mathbf{x}_i, \mathbf{x}_{i+1}) = V(\mathbf{x}_i, \mathbf{x}_{i+1}) \frac{|\cos(\theta_i)\cos(\theta_i')|}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|^2} \quad [3.16]$$

θ_i and θ_i' are the angles between $\mathbf{x}_i \rightarrow \mathbf{x}_{i+1}$ and the surface normals at $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ respectively. The visibility term $V(\mathbf{x}_i, \mathbf{x}_{i+1})$ has value 1 if $\mathbf{x}_i$ can see $\mathbf{x}_{i+1}$ and 0 otherwise.

In the context of computing the rendering equation integral using a Monte Carlo method, we want to draw samples that are random light transport paths, $\bar{X}_k, k = 1, \ldots, n$, according to some chosen density function p, and then compute the estimate:

$$\hat{I} = \frac{1}{n} \sum_{k=1}^{n} \frac{f(\bar{X}_k)}{p(\bar{X}_k)} \quad [3.17]$$

3.4.2 Monte Carlo Algorithms for Global Illumination

Kajiya [47] introduced the first unbiased Monte Carlo based solution called *path tracing* for solving the rendering equation. Over the years, many other algorithms have been developed for solving the rendering equation. Here, we briefly summarize some important algorithms and provide an historical timeline.

3.4.2.1 Path Tracing

Introduced by James Kajiya in the paper in which he first described the rendering equation [47], path tracing was the first general light transport algorithm to compute a complete global illumination solution. Path tracing builds random ray trees rooted at the eye and considers each valid transport path as a sample.

Path tracing generates a path by starting a ray from the camera, recursively tracing the ray in the scene, and ending at light sources. At each bounce, a direction is sampled according to a distribution, for example a BRDF function or a cosine function. The contribution of the path to the image plane is evaluated by the radiance the path carries weighted by the probability of this path being generated.

A variation of this algorithm is to trace rays from light sources to the camera. This is called light tracing (also known as particle tracing, or backward ray tracing). Light tracing is a dual

algorithm of path tracing because the physics of light transport do not change when a path is reversed. They both have advantages and disadvantages. Furthermore these two algorithms can be coupled to improve the image results. The algorithm combing path tracing and light tracing is called bidirectional path tracing, which is discussed next.

3.4.2.2 Bidirectional Path Tracing

Bidirectional path tracing was developed independently by Lafortune [54] and Veach [101]. They formulated their algorithms based on different statistical frameworks, however. Veach introduced multiple importance sampling as the basis for his bidirectional importance sampling algorithm, while Lafortune formulated his as a recursive evaluation of the global reflectance distribution function (GRDF).

Bidirectional path tracing is a generalization of the standard path tracing algorithm. The main observation in bidirectional path tracing is that some sub-paths are more efficiently sampled starting from the light "backward" while others are more efficiently sampled starting from eye "forward." The backward sub-paths can provide important information for forward sub-paths, and vice verse. Each pair of "backward" and "forward" sub-paths can be connected at different vertices to form multiple full paths. Those full paths are then combined with appropriate weights to form the estimator. The choice of weights has great impact on the variance of the estimator. Multiple importance sampling [102] provides a theoretical basis and a near-optimal way for setting the weights.

As with path tracing, bidirectional path tracing is unbiased and can handle arbitrary geometry and lighting. It combines the advantages of path tracing and light tracing. Bidirectional path tracing can dramatically reduce the variance for indirect lighting estimation compared to path tracing. However, an image created using bidirectional path tracing is still noisy and needs many samples to converge. Because subpaths have to be connected to form valid full paths, bidirectional path tracing is not suitable for scenes where most "forward" and "backward" subpaths are not visible to each other.

This has the advantage of combining both visual importance and the lights' power, but the disadvantage that each path is independent; while a difficult path may be located by random chance,

3.4.2.3 Irradiance Caching

Irradiance caching is a technique that exploits the fact that indirect lighting often changes slowly over diffuse surfaces [107]. So, if the indirect lighting is computed accurately at a sparse set of scene locations and cached into a data structure, the indirect lighting at new locations can be approximated with an interpolation of those cached values. It works this way: when extant radiance at a diffuse point is being computed, the irradiance cache is looked up to see whether one or more acceptable nearby samples exist. If so, an interpolated irradiance value from those samples is used to compute the new radiance; otherwise, the accurate irradiance at that point is estimated and stored in the irradiance cache.

To make the method work, there are three questions to be answered: (1) When is it acceptable to use nearby cached values to approximate the irradiance at the new location? (2) How are estimates interpolated? and (3) What data structure should be used to store the computed irradiance values so that the look up is fast?

The gradients of the irradiance [106] are used to determine when the irradiance at a new location can be approximated with reasonable accuracy as an interpolated value of the nearby cached values. This approach takes account of not only the distances to the nearest surfaces, but also the irradiance gradient due to a change in position as well as orientation. This approach does not require any further samples, but simply uses a sophisticated analysis of the samples in the irradiance estimate.

Since only the irradiance is cached, the information on the directional distribution of the incoming radiance is lost, and so this technique can only be used for diffuse surfaces.

3.4.2.4 Metropolis Light Transport

Metropolis Light Transport (MLT) is a robust global illumination algorithm that applies Metropolis sampling to photo-realistic rendering [103]. Metropolis sampling is a Markov chain Monte

Carlo (MCMC) technique that can generate a sequence of dependent samples from a non-negative function f, with f as the stationary distribution of that chain. It only requires that f is known up to a constant scale and can be evaluated at each point in the domain. In other words, no analytical form for f is necessary.

Veach and Guibas showed that Metroplis sampling can be applied to infinite dimensional path space for global illumination. The algorithm starts by generating a set of path samples using bidirectional path tracing. These paths are modified using different mutation strategies to obtain tentative new paths. A tentative path is accepted as a new path according to the acceptance probability computed as in the Metropolis sampling algorithm.

The mutation strategies in MLT correspond to the *proposal distribution*, which greatly affects the convergence of the Markov chain. To make MLT efficient, mutation strategies have to be designed so that the path space is efficiently explored through the path random walk. The mutation strategies proposed in the MLT paper included bidirectional mutation, perturbations, and lens subpath mutation. Bidirectional mutations are used to make big changes to the path and guarantee the whole path space can be visited (to ensure ergodicity of the Markov chain).

The key advantage of MLT is that various coherent structures in the path space are explored and, as a result, once a difficult sample path is found, this path will be reused and exploited. MLT is very efficient in handling traditionally difficult scenes such as light going through an ajar door. Another advantage of MLT is that the Metropolis sampling framework ensures its unbiasedness. MLT is also competitive with previous unbiased algorithms for relatively simple scenes.

3.4.2.5 Photon Mapping

Photon mapping [44] is a two-pass global illumination algorithm. The first pass uses standard light tracing to shoot photons from light sources. Whenever a photon intersects a non-specular surface (diffuse or glossy), the intersection point, incoming direction, and flux of the photon are stored in a cache called the *photon map*. The second pass renders the image by taking advantage of the photon maps built in the first pass which significantly speeds up the rendering process.

The photon mapping algorithm divides the integrand into four components: direct lighting, specular reflection, caustic, and indirect lighting (multiple diffuse reflection). Direct lighting and specular reflection are accurately evaluated using standard Monte Carlo ray tracing. The caustics are evaluated via a caustic map. Indirect lighting is computed through a *final gathering*, which uses the global photon map to estimate the incoming radiances. The radiance estimate from the photon map is based on nearest neighbor density estimation, which is a well-studied discipline in statistics. Note that the radiance estimate using a photon map introduces bias.

Photon mapping can handle all illumination phenomena, including caustics, color bleeding and specular reflection, in a reasonably efficient manner. Another advantage is that the photon map does not depend on the underlying scene geometry, which means it scales well with scene complexity.

To make the final gathering step efficient, irradiance caching can be used to compute the indirect lighting for diffuse surfaces. When a final gather ray hits a diffuse surface, the irradiance cache is searched for a single nearby good sample. If found, its irradiance can be used to estimate the outgoing radiance by multiplying the BRDF value. Otherwise, computing the irradiance is done by using photon density estimation and adding it to the irradiance cache. Using the irradiance cache avoids repeating some density estimation.

3.4.2.6 Sampling Importance Resampling for Direct Lighting

Recently, two algorithms for direct lighting were proposed based on sampling importance resampling method (SIR): Bidirectional importance sampling (BIS) [7] and Resampling importance sampling (RIS) [94]. In these algorithms, for the outgoing direction ω_o along which the radiance is to be estimated, first M incoming direction samples $\tilde{\omega}_i^{(1)}, \cdots, \tilde{\omega}_i^{(M)}$ are generated from an importance distribution $p(x)$, which is usually either BRDF sampling or light sampling, and the importance weights for those samples are computed. Then N samples $\omega_i^{(1)}, \cdots, \omega_i^{(N)}$ are generated by resampling the M initial samples based on their importance weights. The estimator for the

direct lighting along ω_o is

$$\hat{L}(\vec{p},\omega_o) = \left(\frac{1}{M}\sum_{m=1}^{M}\frac{f_r(\vec{p},\omega_o,\tilde{\omega}_i^{(m)})L_i(\vec{p},\tilde{\omega}_i^{(m)})\cos\theta_i^{(m)}}{p(\tilde{\omega}_i^{(m)})}\right)\left(\frac{1}{N}\sum_{n=1}^{N}V(\omega_i^{(n)})\right) \qquad [3.18]$$

where $V(\omega_i^{(n)})$ is the light visibility test for surface point $\vec{p}$ along the direction $\omega_i^{(n)}$. The estimator can be interpreted as computing reflected radiance from M direction samples without visibility testing and scaling it by the average result of N visibility tests of those samples having big contributions to the radiance.

M is usually one to two orders of magnitude larger than N. The key observation used in the algorithms is that generally it is much cheaper to generate direction samples than to do visibility testing. The algorithm gains by postponing visibility testing until the resampling step so that only N tests are needed and visibility tests are only performed for high-contribution directions. Talbot et al. [94] further showed how to choose M and N to achieve near optimal variance reduction.

These algorithms are good for generating samples from a PDF that can be decomposed into two factors: one is cheap to compute and incorporates most of the variance, and another that is expensive to compute and has low variance. However, for the application in direct lighting, due to not considering the visibility test in the initial sampling, the algorithm does not work well for partially occluded regions. Actually, the algorithm will fail in the following scenario: a ball on a floor is lighted by two lights. One light is much brighter than the other. For the shadow region that is occluded from the bright light but visible to the dim light, the direct light computation using the BIS or RIS algorithms will be very poor because almost all of the N resampled samples will be from the bright light and turn out to be useless in the estimation due to the occlusion.

Additionally, the choice of $p(x)$ makes a big difference of the efficiency in the algorithm as well. If $p(x)$ is far away from the target distribution, most samples will end up with very low importance weights, which means low contribution to the light estimation. Designing a good $p(x)$ for this algorithm is not a trivial task, however.

3.4.2.7 Energy Redistribution Path Tracing

Cline et al. [13] introduced an energy redistribution (ER) sampling method for estimate integrals that are correlated. As an application of ER sampling to the global illumination problem, Energy Redistribution Path Tracing (ERPT) is a hybrid global illumination algorithm that combines the ideas of metropolis light transport and path tracing.

In some sense, ERPT algorithm can be understood as a parallel MLT: the algorithm starts a set of initial paths using path tracing, and then uses each initial path as a seed for a Markov chain. As in MLT, the current path is mutated to get a tentative path and the tentative path is accepted with a probability to maintain the *detailed balance* for the chain. Several path mutation strategies are designed to redistribute the energy of the samples over the image plane to reduce variance. Instead of using bidirectional mutation in MLT to guarantee the ergodicity in the Markov chain, ERPT just re-generates a totally new path using path tracing with a non-zero probability. Two other mutation strategies include lens perturbation and caustic perturbation.

This algorithm uses post-processing noise filters to reduce image noise; however, this introduces bias.

Year	Algorithm	Pros	Cons
1986	Path Tracing [47]	Unbiased; First general solution to the rendering equation	Very slow to converge for indirect lighting
1992	Irradiance Caching[107, 106]	Fast for diffuse scenes	Biased; Fails for caustics and shadow boundaries; Only works for diffuse surfaces
1994	Bidirectional Path Tracing[101, 54]	Unbiased; Much more efficient in indirect lighting than path tracing	Not efficient to compute slowly varying diffuse components
1996	Photon Mapping [44, 45]	Works well in practice; Industry standard.	Biased; Wrong density estimation can lead to light leaking; Inefficient if most lights can not reach the image plane
1997	MLT [103]	Unbiased; Reuses path samples; Handles difficult paths well	Difficult to implement
2005	SIR for Direct Lighting [7, 94]	Unbiased; Good for scenes without much occlusion	Only works for direct lighting; Bad for partially occluded regions
2005	ERPT [13]	Easier to understand and implement than MLT; keeps most MLT features	Biased after using filter

Table 3.1: Monte Carlo algorithms for global illumination.

Chapter 4

Metropolis Photon Sampling

Photon Mapping [45] is the current choice of industry for scenes with general surface primitives and reflectance functions [23]. It uses an initial pass to populate photon maps with samples of the power arriving at points in the scene. A *final gather* pass then uses the maps to estimate the contribution of indirect illumination to visible pixels. It is essential that the initial pass populate the maps with photons useful to the final gather, but the standard technique fails to do so in some common scenes. This chapter introduces *Metropolis Photon Sampling* (MPS), a Monte Carlo sampling algorithm for constructing photon maps that produces high-quality results in situations where standard photon map construction fails. MPS also gives users a technique to control variance over the image.

Standard Photon Mapping traces particles from the lights distributed according to the lights' power distribution, and deposits photons when the particles interact with surfaces. It performs poorly when little of the lights' total power arrives at locations important to the final gather. This situation is not uncommon in practice: indoor environments may have many lights that contribute unevenly to the image (Figure 4.1); in some scenes most light paths are occluded (Figure 4.5); and local views of outdoor scenes may see little of the sun's power (e.g., under a forest canopy or in downtown city streets). Poor sampling results in excess noise in the indirect illumination estimates derived from the map. Furthermore, low photon density leads to larger search radii in accessing photons, which causes inappropriate samples to be included and hence severe energy bleeding.

Both effects are evidenced in the left image of Figure 4.1, based on the photon distribution on the left in Figure 4.2.

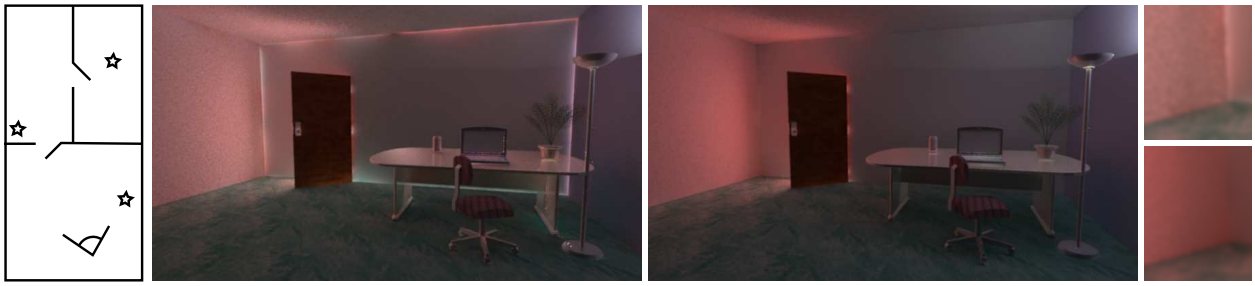

Figure 4.1: Leftmost is the plan of a scene in which only a small portion of the lights' total power contributes to the image. The left image was produced using standard Photon Mapping, which under-samples some regions and over-samples others, resulting in image noise and severe energy bleeding from the adjacent room (the cause of the incorrect illumination around the edges of the rear wall). To the right is our result. Paths joining the eye to a light were sampled and photons were stored only in important locations. The insets on the far right show zoomed sections taken from the center-left of the images, and demonstrate how our method (lower) both reduces noise and avoids energy bleeding.

One underlying cause of a poor sample distribution is the lack of visual importance information; sampling from the light does not consider the camera location. Our first contribution is a technique, *Metropolis Photon Sampling* (MPS), that builds photon maps using complete light paths that join a light to the eye. By linking to the eye we account for visual importance and can identify photon storage locations that will be useful to the final gather (Figure 4.2). This reduces image noise and energy bleeding artifacts in scenes where most paths traced only from the lights are irrelevant to the image (Figure 4.1). MPS uses a Metropolis-Hastings algorithm [64, 39, 31] to sample over paths, but the general framework supports other sampling methods.

Regardless of the sampling strategy used, light paths that are difficult to find randomly lead to image artifacts in Monte Carlo rendered images. In Photon Mapping this tends to manifest itself as smooth but incorrect results, while in a pure Monte Carlo framework the result is noise. Frequently the difficult paths are obvious to a user: light may have to pass through a small opening or be focused by a particular scene element. Our second contribution enables a user to provide a small set of important light transport paths that the sampling process uses to reduce variance. No

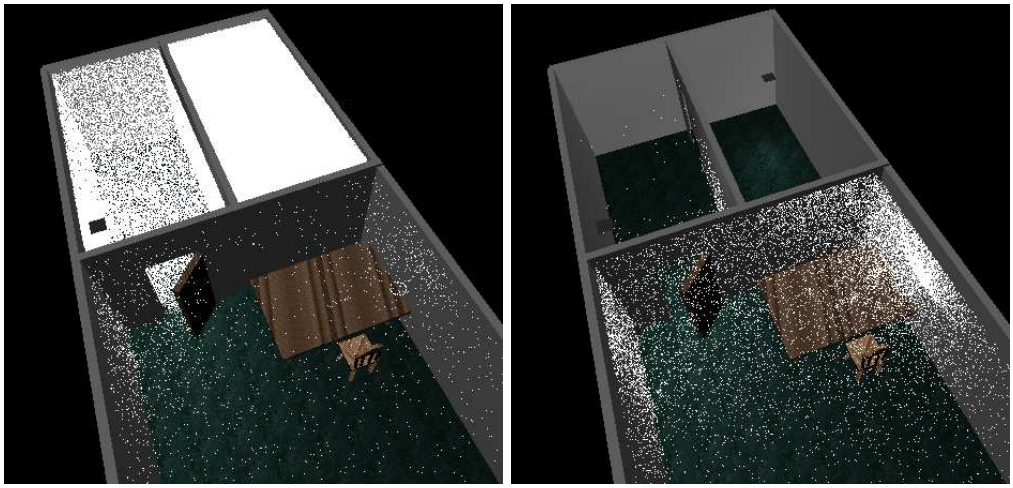

Figure 4.2: Photon distributions for Figure 4.1. While standard Photon Mapping generates many photons in a short period of time (left), they are almost all located in places not relevant to the final image. Right is our result for identical computation time, with all the samples in locations useful to a final gather operation.

bias is introduced to the result. User defined paths help when sampling from difficult geometric arrangements, and also give a user local control over variance in the image. For instance, in Figure 4.1 the user suggested 10 paths that carry light through the doorway from the neighboring room. This is the first technique in the rendering literature for including specific user-defined sample paths in a Monte Carlo framework.

4.1 Related Work

The rendering equation [47, 73] is the physical foundation for image synthesis. Many Monte Carlo based algorithms have been proposed to solve the equation, such as path tracing and bidirectional path tracing (see chapter 3 for a summary). Kollig and Keller [53] addressed this problem with quasi-Monte Carlo methods, which can exploit coherence in random number space under the assumption that paths generated with similar random choices are similar paths, which is not necessarily the case in even mildly complex scenes.

Veach [103] presented *Metropolis Light Transport* (MLT), which is a Markov chain Monte Carlo (MCMC) algorithm designed to exploit coherence in path space. MCMC views sampling as

a Markov process, and hence a good sample found in one step can improve subsequent samples. MCMC also allows multiple sampling strategies to be combined without introducing bias, which enables us to incorporate user-guided sampling. Veach's *Multiple Importance Sampling* [102] also combined different strategies, and it could also support user input of the form we propose. An alternate MCMC approach to rendering was proposed by Kelemen et al. [49]. Rather than sampling in path space, they sampled on a high-dimensional unit cube. MLT was extended to support participating media by Pauly et al. [71].

A single sample may be representative of illumination over a large region if radiance varies slowly, as is often the case in scenes with significant indirect diffuse illumination. *Particle tracing* algorithms, of which Photon Mapping is one, exploit this to re-use light paths. Arvo [3], Heckbert [40] and Collins [14] proposed algorithms that use *illumination-maps* to store irradiance arriving along sampled paths. Like Photon Mapping, particles are traced from the lights, but they require parameterized geometry for the maps. The method of Shirley et al. [84] traces particles and builds a polygonal mesh representation that can be rendered in real time for varying viewpoints. Chen et al. [9] also worked with 2D maps but, in addition, offered a progressive refinement solution. Our sampling method could be used with any of these existing techniques, with some modification to particle storage. Ward's *RADIANCE* system [107, 105] traced rays from the eye and cached diffuse contributions for use in subsequent estimates. The *irradiance caching* technique [106] was used to determine if the cached samples provide an adequate estimate.

Many rendering algorithms have been developed to exploit visual importance; see Christensen [10] for a survey. Specific to particle tracing, *importon* techniques trace particles from the eye to construct an *importon map* that is used to estimate visual importance. Peter and Pietrek [72] used the importon map to construct importance sampling distributions for each scattering event of the particle tracing phase. The algorithm is expensive due to the cost of computing distributions at every particle bounce, its local decisions may not produce a globally important path, and the importance sampling produces photons with highly variable power. Keller and Wald [50] used importon maps to avoid photon storage in areas that contribute little to the final image. Their technique reduces

memory usage and maintains roughly uniform photon power, but gives no control over the generation of the samples in the first place. Suykens and Willems' [89] algorithm considers the current sample density in the photon map when storing a new sample and redistributes its power if it would result in excess density (without modifying photon generation). Unlike existing methods, our algorithm samples from complete paths joining the light to the eye and thus efficiently accounts for visual importance without using importons. Complete paths also provide information about important photon storage locations and hence reduce redundant photons.

Variance is typically controlled by using more samples, or designing new algorithms (not a natural tool for most end-users). Ward [105] allows users to specify surfaces as important secondary light sources, and the system builds their outgoing irradiance functions for use in indirect illumination. The technique is targeted at large secondary sources, such as windows, but fails if the secondary source itself is not easy to reach from the light or no one surface is significant enough to warrant the attention. Our approach allows a user to specify paths through multiple reflections, and places no restrictions on the surfaces or pieces of surface affected. A related idea to user input is sampling based on pilot paths that are found in a random initial pass (or in the previous frame of an animation). Dmitriev et al. [18] discuss this approach in the animation context, but it relies on similarity in random number space to compute path perturbations. With user input, there are no random variables associated with the paths, so this approach cannot be applied.

4.2 Light Paths to Photons

We incorporate visual importance into photon map construction by extracting photons from *complete* light paths that join a point on a light source to the eye via some number of scattering (reflection or transmission) events. Complete paths also allow us to identify the point on the path at which a photon should be stored. Assume for the moment that we can produce sample light paths. In the next section we address the way photons are extracted from the paths.

4.2.1 Photon Locations

Given a light path, we wish to identify the point or points along it that will be accessed during a photon map lookup. This clearly depends on how the final gather is performed. We use a standard Photon Mapping final gather as described by Jensen [45], to whom we refer the reader for motivation and details. Estimation of radiance from the global photon map takes place at points that lie at the second diffuse bounce on paths traced from the eye (possibly with intervening specular bounces). Hence, we store a photon at the second diffuse point for each path that our sampler produces. Estimation from caustic photons occurs at the first diffuse bounce, so along caustic paths we store a photon in both the global and caustic map at the first diffuse point. In any case, we refer to the photon storage location on a path as the *storage point*.

The nearest neighbors around a point of interest, $\mathbf{p}$, are used when estimating radiance from the maps. The neighbors are assumed to be representative of the incoming radiance at $\mathbf{p}$, which requires that radiance vary slowly in the region from which they come. This assumption is more likely to be true, and hence the estimate better, as the density of photons around $\mathbf{p}$ increases and the neighbors fall within a smaller region. Our algorithm ensures that most stored photons lie around points where final gather estimates are formed, and hence improves the quality of the estimate for a given map-building effort and memory footprint.

The use of a kd-tree for photon storage removes the need for a surface parameterization (allowing for a wider range of surfaces and fast neighbor lookup) but this also decouples photons from surface properties. Severe light bleeding can occur due to the breakdown of the slowly varying radiance assumption, which is hard to detect without surface information. This is a major problem in scenes where a light is on the back side of a thin divider, as in Figure 4.1.

A common practical solution is to store a normal vector with each photon and require that it be similar to the normal at the point where the estimate is being taken. This reduces bleeding in concave corners, but fails in our scenes. For instance, the floor is oriented the same on both sides of the wall in Figure 4.1. Importon based methods (Section 6.2) fail to address the energy bleeding through walls problem because importance can leak just as energy does, allowing photons to be stored in unimportant regions. However, points on the back side of a wall are almost never the

second diffuse bounce on a path from the eye, so our method automatically avoids storing them and hence significantly reduces energy bleeding in from unimportant areas of the scene.

4.2.2 Photon Storage

As with standard Photon Mapping, for each photon j, we store the location, $\mathbf{x}^{(j)}$, incoming ray direction, $\theta^{(j)}$, and radiant flux (power), $\Phi^{(j)}$. In this section we describe how $\Phi^{(j)}$ is computed for a sampled path. Our discussion is based on the particle tracing framework introduced by Veach [100, §4.A] and applied to Photon Mapping by Pharr and Humphreys [73]. Expressed in terms of sampling from path space, we require that each photon j have power such that

$$E\left[\sum_{\mathcal{R}} \Phi^{(j)}\right] = \int_{\Omega_{\mathcal{R}}} f_{map}(\overline{\mathbf{x}}) \mathrm{d}\mu(\overline{\mathbf{x}}) \qquad [4.1]$$

where the sum is over the set of photons within a region of area and solid angle, $\mathcal{R}$. The integral is over $\Omega_{\mathcal{R}}$, the set of light transport paths that begin on a light and end within the region, $\mu(\overline{\mathbf{x}})$ is the surface area measure for the path $\overline{\mathbf{x}}$, and $f_{map}(\overline{\mathbf{x}})$ is defined as

$$\begin{aligned} f_{map}(\overline{\mathbf{x}}) &= L_e(\mathbf{x}_0, \mathbf{x}_1) G(\mathbf{x}_0, \mathbf{x}_1) \\ &\quad \cdot \prod_{i=1}^{m-1} f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1}) G(\mathbf{x}_i, \mathbf{x}_{i+1}) \end{aligned}$$

in which $\mathbf{x}_i$ is a point on the path $\overline{\mathbf{x}}$, $L_e(\mathbf{x}_0, \mathbf{x}_1)$ is the radiance emitted by a light point $\mathbf{x}_0$ toward $\mathbf{x}_1$, $f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ is the bidirectional scattering distribution function for surface point $\mathbf{x}_i$, and $G(\mathbf{x}_i, \mathbf{x}_{i+1})$ is the geometry term between points $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$:

$$G(\mathbf{x}_i, \mathbf{x}_{i+1}) = V(\mathbf{x}_i, \mathbf{x}_{i+1}) \frac{|\cos(\theta_i) \cos(\theta_i')|}{\|\mathbf{x}_i - \mathbf{x}_{i+1}\|^2}$$

where θ_i and θ_i' are the angles between $\mathbf{x}_i$—$\mathbf{x}_{i+1}$ and the surface normals at $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ respectively. The visibility term $V(\mathbf{x}_i, \mathbf{x}_{i+1})$ has value 1 if $\mathbf{x}_i$ can see $\mathbf{x}_{i+1}$ and 0 otherwise.

If we consider the region of interest, $\mathcal{R}$, to be all the points accessed during the final gather, Equation 4.1 takes the form of a Monte Carlo estimate of an integral. The sum on the left is over all the photons in the map, and the integral on the right evaluates to the total power arriving in the map, B_{map}. If we sample paths according to the distribution $p_{map} = f_{map}(\overline{\mathbf{x}})/B_{map}$, each one of the N photons should have the same power: $\Phi = B_{map}/N$.

We only store photons at points relevant to the final gather, so the above discussion assumes we are sampling over paths terminating at such points. However, the designation of storage points relies on having the complete path to the eye, in order to count the number of diffuse bounces on the sub-path from the eye. To obtain this information, MPS samples from the space of all paths that join the light to the eye and stores photons only for the desired sub-paths. We sample according to the probability distribution function (PDF) given by $p_{eye}(\overline{\mathbf{x}}) = f_{eye}(\overline{\mathbf{x}})/B_{eye}$, where

$$\begin{aligned} f_{eye}(\overline{\mathbf{x}}) &= W(\overline{\mathbf{x}})L_e(\mathbf{x}_0, \mathbf{x}_1)G(\mathbf{x}_0, \mathbf{x}_1) \\ &\quad \cdot \prod_{i=1}^{m-1} f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})G(\mathbf{x}_i, \mathbf{x}_{i+1}) \end{aligned} \quad [4.2]$$

The function $W(\overline{\mathbf{x}})$ takes the value 1 if the path passes through the image plane, and 0 otherwise. B_{eye} is the normalizing constant, in this case the total power arriving at the image, and should satisfy

$$B_{eye} = \int_{\Omega_{eye}} W(\overline{\mathbf{x}}) f_{eye}(\overline{\mathbf{x}}) d\mu(\overline{\mathbf{x}})$$

where Ω_{eye} is the space of all paths that join a light to the eye. Following Veach [100], path tracing is used to estimate this integral. Not many path tracing samples are required because we are averaging over all pixels.

When we use p_{eye} as the target distribution the resulting samples will no longer be distributed according to p_{map} as required for correct photon map estimation (Equation 4.1). This is accounted for using standard importance sampling re-weighting:

$$\Phi^{(j)} = \frac{1}{N} \frac{f_{map}(\overline{\mathbf{x}}^{(j)}_{map})}{p_{eye}(\overline{\mathbf{x}}^{(j)})} = \frac{B_{eye}}{N} \frac{f_{map}(\overline{\mathbf{x}}^{(j)}_{map})}{f_{eye}(\overline{\mathbf{x}}^{(j)})}$$

where $\overline{\mathbf{x}}_{map}$ is the sub-path $L(D|S)^*D$ from a sampled path of the form $L(D|S)^*DS^*DS^*E$ for which a photon is stored in the global map, or the sub-path LS^*D of an LS^*DS^*E path for caustic photon storage. Note that we no longer require B_{map}. Furthermore, when sampling according to $p_{eye}(\overline{\mathbf{x}})$ we may generate paths that do not result in photon storage (i.e. not of the form $L(D|S)^*DS^*DS^*E$ or LS^*DS^*E). In this case, $f_{map} = 0$ and no photon is stored.

The Metropolis-Hastings sampler we use may provide many paths with the same storage point, $\mathbf{x}^{(j)}$, and incoming ray direction, $\theta^{(j)}$. This is due either to rejection of candidate paths, in which

case the entire path is repeated, or a path mutation that retains the storage point while changing some other part of the path (see Section 4.3). Instead of generating a new photon in such cases, we accumulate the power in a single photon and hence reduce photon storage cost and look-up time. In practice, few paths contribute to any one photon and the resulting per-photon power variation does not create artifacts.

The scattering function $f_s(\mathbf{x}_{i-1}, \mathbf{x}_i, \mathbf{x}_{i+1})$ is wavelength dependent. We evaluate f_s for the standard RGB channels, and use them to compute $f_{map,R}$, $f_{eye,R}$, etc. For the sampling process we must attach a single probability to each path. We use the luminance channel, $f_{eye,Y}$, computed by the RGB to XYZ color conversion. With this path probability, the red power for the stored photon (green and blue are similar) is

$$\Phi_R^{(j)} = \frac{B_{eye,Y}}{N} \frac{f_{map,R}(\overline{\mathbf{x}}_{map}^{(j)})}{f_{eye,Y}(\overline{\mathbf{x}}^{(j)})}$$

The framework developed to this point does not depend on the method for finding sample paths, or even on their PDF, p_{eye}. Any sampling technique capable of generating paths from the light to the eye, such as bidirectional path tracing, could be used. We chose a Metropolis-Hastings sampler because it can both exploit coherence in path space and support user input.

4.3 Sampling Paths

Metropolis-Hastings algorithms use a Markov process designed to obtain a sequence of samples whose distribution converges to a target PDF. Following Veach [100], to estimate radiometric quantities we want each sample path, $\overline{\mathbf{x}}$, to come from the space of all transport paths joining the light to the eye, Ω_{eye}. The target PDF is $p_{eye}(\overline{\mathbf{x}})$. Each path $\overline{\mathbf{x}}$ with m segments is parameterized by the surface intersection points at which a scattering event occurs, $\mathbf{x}_i$, $i \in [1, \ldots, m-1]$, along with the final point, $\mathbf{x}_m$, and the point on the light source from which the particle is emitted, $\mathbf{x}_0$.

The Markov process generates each sample in the sequence, X_t, by proposing a candidate, X'_t, based on the previous sample X_{t-1}, and either accepting this candidate as X_t or rejecting it and repeating X_{t-1}. In pseudo-code:

$X_0 \leftarrow initialSample()$

for $t = 1$ to N

$X'_t \leftarrow propose(X_{t-1})$

$r \leftarrow uniformRandom[0, 1)$

if ($r < \alpha(X'_t|X_{t-1})$) then

$X_t = X'_t$

else

$X_t = X_{t-1}$

The procedure $initialSample$ chooses one of the paths generated by the path tracing computation for B_{eye}, according to the distribution p_{eye}. The initial sample chosen in this way is unbiased, so there will be no start-up bias in the Markov chain [31, 100]. The proposal function, $propose(X_{t-1})$, produces a new light path by applying a random modification to the current sample. While the correctness conditions placed on the modifications are not difficult to satisfy, the strategies employed are the primary factor in determining the efficiency of the algorithm (the number of samples required for a good estimate). We describe our mutation strategies below.

The function $\alpha(X'_t|X_{t-1})$ computes the *acceptance probability* for X'_t given the current sample.

$$\alpha(X'_t|X_{t-1}) = \min\left\{1, \frac{f_{eye,Y}(X'_t)T(X_{t-1}|X'_t)}{f_{eye,Y}(X_{t-1})T(X'_t|X_{t-1})}\right\} \qquad [4.3]$$

The function $f_{eye,Y}(X'_t)$ is proportional to the target PDF $p_{eye}(\overline{\mathbf{x}})$ (and the normalization constant cancels out).

$T(X'_t|X_{t-1})$ is the transition function (or proposal distribution) which gives the probability of choosing, by any means, X'_t given X_{t-1}. Note that the reverse transition function, $T(X_{t-1}|X'_t)$, is also required, and in a Metropolis-Hastings sampler it need not equal $T(X'_t|X_{t-1})$.

4.3.1 Proposal Strategies

The techniques used in the $propose(X_{t-1})$ procedure of the MCMC algorithm are the key to its efficient and correct operation. There are two conflicting goals in designing a good proposal. The candidate path, X'_t, should be as different as possible from the current path, X_{t-1}, to rapidly move around the sample state space. At the same time it should be sufficiently similar to X_{t-1} to exploit

coherence in high-power paths. The technical conditions on $propose(X_{t-1})$ ensure that there is some non-zero probability way to move between any two non-zero probability paths (see Gilks et al. [31]). The acceptance probability, $\alpha(X'_t|X_{t-1})$, is specifically designed to take *any* proposal strategy that meets the conditions, properly encoded in the transition functions $T(X_{t-1}|X'_t)$ and $T(X'_t|X_{t-1})$, and create an unbiased sampler.

We introduce two novel mutation strategies. **User Path (UP)** proposals make use of user hints about which paths are likely to be important to the final result (Section 4.4). The variance of any estimate is reduced around the given paths. **Photon Map (PM)** proposals explore paths that will contribute to the global photon map (Section 4.4.3). They change the sample path while retaining the DS^*E sub-path to the eye.

In addition, four other proposal types previously described for MLT are suitable for use here [103]. **Bidirectional (BD)** proposals modify sub-paths of the current path, with the aim of rapidly exploring the sampling space. **Caustic Perturbation (CP)** and **Lens Perturbation (LP)** proposals also modify sub-paths, but this time with the aim of exploiting coherence in high-power, localized features. Finally, **Lens Sub-path (LS)** proposals stratify samples across the image, which ensures that enough samples are captured in darker regions of the scene. We implement each of these strategies in the same manner as MLT.

Each time the $propose(X_{t-1})$ procedure is called we choose one of the above strategies at random according to a fixed distribution. That is, $propose_{type}(X_{t-1})$ is selected with probability P_{type} where $type$ is one of the above options and $\sum_{type} P_{type} = 1$. In computing the transition function, $T(X'_t|X_{t-1})$, all possible proposals that might generate X'_t from X_{t-1} should be considered:

$$T(X'_t|X_{t-1}) = \sum_{type} P_{type} T_{type}(X'_t|X_{t-1}) \qquad [4.4]$$

However, it is also acceptable to consider only the function derived from the proposal strategy chosen to generate X'_t [97, 2]:

$$T(X'_t|X_{t-1}) = T_{chosen}(X'_t|X_{t-1}) \qquad [4.5]$$

We use a combination of both strategies: Equation 4.5 avoids the computation of unnecessary transition functions, but Equation 4.4 is required for user path proposals (Section 4.4.2).

4.4 User Path Proposals

The user path proposal strategy increases the proportion of candidate paths around those supplied by the user. This results in variance reduction for any estimate based on the paths, such as photon map evaluation. There are several applications:

Difficult Paths: Transport paths that are particularly hard to find randomly lead to large variance because they may be found and give a high contribution, or are not found and give no contribution. Among our images, the caustic caused by light bouncing off the mirror and through the glass ball in the Box scene of Figure 4.6 best fits this description. Light shining through a keyhole is perhaps the most commonly thought of example, if not the most common in practice. A user can supply paths that meet the geometric constraints and thus ensure the feature is adequately sampled.

User Control of Variance: Some regions of an image may be more important than others, such as those toward the center or in some other perceptually important region. A user can supply paths leading to the region of interest and it will be sampled with lower variance than other regions (Figure 4.3, page 67).

Resampling: Rather than a user defining paths, they could be taken from some previous sampling operation. Our earliest experiments used paths taken from the initial path tracing pass to estimate B_{eye}. Alternatively, a user could identify paths from a coarse run of the algorithm and re-use them in a final render. Resampling should also enable adaptive, unbiased Monte Carlo rendering and provide a handle on low-variance, physically-accurate animation rendering, but we leave these topics for future work.

Figure 4.3 compares images rendered with the Metropolis Light Transport algorithm: one with user paths and one without. Each image used 3 million iterations, producing a variance measurement of $VAR(E) = 1.04$ (Section 6.5) for the image with user input. It requires 4.8 million samples, or about 60% more time, to achieve similar results without user input.

Reducing variance in one area of the image may lead to increased variance elsewhere, but it is not a zero-sum game. User paths can lead to a global reduction in variance if they increase the average acceptance probability, and hence the number of different paths sampled. This was the case in Figure 4.3, where the acceptance rate rose from 58% to 65% with the introduction of user paths. In any event, users can choose to make a trade-off based on their own situation.

The user path proposal is not essential to achieving good results with Metropolis-Hastings sampling. It is a way to enhance control of the algorithm. The image in Figure 4.5 did not use the proposal, and the result in Figure 4.1 is almost as good without user paths.

4.4.1 Candidates from User Paths

Each path provided by the user must start at a light and end at a diffuse surface. To obtain paths, we built a simple interface for the Box scene which allowed a user to interactively vary the origins and directions of rays from the light which were then traced through the scene and extracted as user paths. Tools like this could readily be included in modeling packages. For Figure 4.3 we specified paths by hand based on knowledge of the geometry.

Each path is input to the system as a sequence of surface points at which scattering occurs. These are stored as a set, $\{\overline{\mathbf{u}}_1, \ldots, \overline{\mathbf{u}}_{N_{UP}}\}$, containing N_{UP} paths. The first step of a proposal is to choose, uniformly at random, one of the input paths, $\overline{\mathbf{u}} = \langle \mathbf{x}_0, \ldots, \mathbf{x}_m \rangle$. This path forms a skeleton that we perturb to form the candidate path. The perturbation explores the space around the user path while avoiding the accumulation of large power at a single photon.

The candidate path, $\langle \mathbf{x}'_0, \ldots, \mathbf{x}'_m \rangle$, is built starting at the light: $\mathbf{x}'_0 = \mathbf{x}_0$. We randomly generate a direction within a cone about axis $\mathbf{x}_0 \rightarrow \mathbf{x}_1$ by sampling θ, the angle between the axis and the direction, uniform in $[0, \beta)$ and ϕ, the azimuthal angle, uniform in $[0, 2\pi)$. The surface point struck in this direction, $\mathbf{x}'_1$, is the next point on the candidate path. We repeat the process m times, using the direction $\mathbf{x}'_{i-1} \rightarrow \mathbf{x}_i$ as the axis of the sample cone. To form a complete path to the eye, the sub-path of sample X_{t-1} joining the eye to the first diffuse point hit is appended to the candidate. The candidate is rejected if there is no such diffuse point. When setting β, lower values are good for exploring tightly constrained paths while higher values give more variation around the user

path and hence reduce variance over a larger area. The user can also specify a different β for each path segment.

The candidate path may pass through an opaque surface, in which case a visibility term in f_{eye} is zero and the path will be rejected. If the user path contains specular interactions, a specular surface must be found at the same index on the perturbed path. If it is, we follow the specular bounce rather than sampling a perturbed direction. If the user path specularity is not matched in the perturbed path, or the perturbed path intersects an unmatched specular surface, the candidate is rejected. These restrictions ensure that specular bounces "cancel out" in computing the acceptance probability (see Veach [100, §10.3.5]).

4.4.2 User Path Transition Functions

The transition probability must consider all the possible ways a UP proposal may have generated the candidate:

$$T_{UP}(X'_t|X_{t-1}) = \frac{1}{N_{UP}} \sum_{i=1}^{N_{UP}} C(\overline{\mathbf{u}}_i) \prod_{j=0}^{m-1} p_j \frac{G(\mathbf{x}'_j \leftrightarrow \mathbf{x}_{j+1})}{\cos\theta'_j} \qquad [4.6]$$

$C(\overline{\mathbf{u}}_i)$ is 1 if the candidate could have been generated from path $\overline{\mathbf{u}}_i$, otherwise 0. The product of terms accounts for the probability of each perturbed bounce. If the bounce at $\mathbf{x}_j$ was non-specular, then $p_j = 1/2\pi\beta_j$. For a specular bounce, $p_j = 1$ because there is no random choice. The geometry terms are still required to convert from the solid angle measure to the surface area measure. The geometry and cosine term convert the direction sampled according to the solid angle measure into one sampled using the surface area measure. θ'_j is the angle between the normal at $\mathbf{x}'_j$ and the direction $\mathbf{x}'_j \rightarrow \mathbf{x}_{j+1}$.

To compute $C(\overline{\mathbf{u}}_i)$, we perform the procedure for building a candidate from $\overline{\mathbf{u}}_i$, but rather than creating the new candidate we check that the point $\mathbf{x}_0$ is common to X'_t and $\overline{\mathbf{u}}_i$ and that each ray direction in X'_t lies within the sample cone of $\overline{\mathbf{u}}_i$. Finally, the resulting number of path segments must correspond. The reverse transition probability, $T_{UP}(X_{t-1}|X'_t)$, is similarly computed.

The UP proposal generates a path, X'_t, close to a user given path regardless of the previous path, X_{t-1}. However, in most cases the path X_{t-1} could not have been generated from X'_t in the

same manner; most paths are not close to a user defined path. Hence, $T_{UP}(X_{t-1}|X'_t)$ will be zero in almost all cases. This leads to a zero acceptance probability, which is a problem because the proposed path will never be used. It is, however, possible to generate a UP proposal candidate using a BD proposal because the latter gives any path a non-zero transition probability. Hence, we combine the UP and BD proposal strategies when computing transition functions: if $chosen$ is either UP or BD, then

$$T(X'_t|X_{t-1}) = \frac{P_{UP}T_{UP}(X'_t|X_{t-1}) + P_{BD}T_{BD}(X'_t|X_{t-1})}{P_{UP} + P_{BD}} \quad [4.7]$$

Thus we have a two tiered proposal selection process. First, we decide if the proposal will be a UP-BD hybrid (with probability $P_{UP} + P_{BD}$) or one of the others. We apply Equation 4.5 for this selection. If the hybrid is chosen, we decide between UP and BD, and apply Equation 4.7.

The combination of UP and BD proposals in computing the transition functions is the key idea for enabling user input samples, and is possible because the acceptance probability mechanism of a Metropolis-Hastings sampler allows different sampling processes (proposal strategies) to be combined. Furthermore, the acceptance criteria ensures that the final distribution is unbiased provided the transition functions and target PDF values are correctly computed. Intuitively, the algorithm rejects just the right proportion of UP candidates to ensure that the final result is not biased toward them.

The values for P_{UP} and P_{BD} will influence performance of the algorithm. Assume that the reverse transition function, $T_{UP}(X'_{t-1}|X_t)$, is very small or zero and consider P_{UP}/P_{BD}, the ratio of UP to BD proposals. As P_{UP}/P_{BD} increases, the acceptance probability (Equation 4.3) will decrease, resulting in the chain repeating the same path more often. This results in fewer photons stored away from the user path (fewer candidates for these paths are proposed), but increases the power of those photons, resulting in a noisier image away from the user path. This effect is counterbalanced by the ratio of the $f_{eye,Y}$ terms, which favors transitions to important paths, including user paths, regardless of how they were proposed.

When using user paths to overcome hard-to-find paths, the ratio P_{UP}/P_{BD} should be higher to provide many user candidates which will be accepted due to their high $f_{eye,Y}$. In the context of

user-guided variance reduction, the ratio should be smaller to avoid frequent rejection of user path candidates and the higher variance that would result in regions away from the user paths. Varying the ratio gives the user control over how much influence their paths have on the distribution of variance over image.

Rather than users providing paths, the user-path proposal could be extended to include hints about important surface patches or reflectance directions. To use important surface patches, for instance, the candidate path can be constructed by randomly choosing points on the patches and joining them up. The terms inside the product in Equation 4.6 must be modified to account for the new probabilities of choosing the points. Otherwise the algorithm is unchanged.

4.4.3 Photon Map Proposal

The Photon Mapping proposal generates complete paths with eye sub-paths that are similar to those used in the final gather phase. Photons derived from complete paths will thus be at locations useful for gathering. Tracing back toward the light from the last diffuse surface point, $\mathbf{x}_d$, (that is closest to the eye) we find a sub-path $\langle \mathbf{x}_{d-k}, \ldots, \mathbf{x}_d \rangle$ of the form $(L|D)DS^*D$. That is, the sub-path back through any number of specular bounces (possibly 0) followed by a diffuse bounce and ending at the next diffuse surface, or the light. The candidate path keeps $\mathbf{x}_d$ and modifies the direction back to $\mathbf{x}_{d-1}$, similar to the way a final gather operation distributes rays to estimate indirect illumination.

Modify the central DS^* portion of the sequence by perturbing the direction of the ray $\mathbf{x}_d \rightarrow \mathbf{x}_{d-1}$ by an angle θ uniform in $[0, \gamma)$ and ϕ uniform in $[0, 2\pi)$ (as in the UP proposal). For all examples in this chapter we set $\gamma = 30°$, and the precise value seems not to impact the results. This ray is traced back through zero or more specular bounces until the next diffuse hit, forming a new DS^* sequence which is inserted in place of the original, resulting in $\left\langle \mathbf{x}_{d-k}, \mathbf{x}'_{d-k'-1}, \ldots, \mathbf{x}'_{d-1}, \mathbf{x}_d \right\rangle$. The diffuse (or light) points at the end of the modified segment allow for non-zero probability that the candidate path will carry some power.

Scene	Resolution	t_B (s)	t_{map} (s)		t_{FG} (s)		Total t (s)		# Photons		RMS Error	
			MPS	PM	MPS	PM	MPS	PM	MPS	PM	MPS	PM
Rooms	720×405	21	40	9	419	469	480	478	81004	300000	0.036	0.4239
Lantern	684×513	11	10	4	185	198	206	202	8675	37160	0.0728	1.165
Box	640×480	9	26	12	208	230	243	242	47798	250000	0.0214	0.0227

Table 4.1: Statistics for images of Rooms, Lantern and Cornell Box scenes. Timing is given for MPS and Photon Mapping: t_B is the time to estimate B_{eye}, t_{map} is the photon sampling time and t_{FG} is the final gather time. While MPS spends more time sampling, the fewer, well-distributed photons reduced the time required for the final gather. We also give the number of photons stored. Memory usage for the maps is linear in the number of photons, with 49 bytes per photon in the PBRT implementation that we use [73]. Finally, we give RMS errors for the images compared against path tracing solutions that ran for several days (Figure 4.7).

The transition probability is similar to that of the UP proposal, except that there is only one perturbed choice followed by a number of specular bounces:

$$T_{PM}(X'_t|X_{t-1}) = \frac{G(\mathbf{x}_d, \mathbf{x}_{d-1})}{2\pi\gamma\cos\theta_d} \prod_{j=d-1}^{d-k'-2} \frac{G(\mathbf{x}'_j, \mathbf{x}'_{j+1})}{\cos\theta'_j}$$

4.5 Results and Discussion

Our rendering system uses libraries and code from the PBRT toolkit [73] wherever possible, including for the final gather operation. There are a variety of parameters to the algorithm. Those for the MLT-style proposals were taken from Veach [100]. For the Photon Mapping final gather parameters, the formula for computing the maximum search distance for photons, d_{max}, was taken from Suykens [88, Pg. 159] ($\alpha = 0.1$) while the maximum number of photons in an estimate, n, was set at 60. We introduced new parameters for the probability of choosing a proposal strategy, P_{type}, which are given below on a per-image basis. We also introduced parameters for controlling the perturbation of a user path, β, which we varied per image, and the perturbation of a photon map sub-path, $\gamma = 30°$.

Timing results and other statistics for the images in this chapter are provided in Table 4.1. All images for comparison between methods were generated with nearly equal total computation

time. All were reconstructed with a Gaussian kernel of width 2 pixels and $\sigma = 1$. Irradiance caching [106] was used to speed up photon map estimation [45]. For tone reproduction we followed Reinhard et al. [77], with the parameter $y_{max} = 100$.

We implemented one further optimization borrowed from MLT. Rather than storing nothing for rejected paths, we store a photon with power reduced according to the acceptance probability, and reduced the power of the repeated path to compensate [100]. This increases the number of photons stored and extracts some benefit from rejected paths, but at the cost of increased variance in photon power. We found the benefits of increased usable photons outweighed the increase in variance.

We also computed error measurements with respect to a long running path tracing estimate of each image. For each pixel, we computed the relative error (before tone mapping):

$$E(x, y) = \frac{I(x, y) - I_{ref}(x, y)}{I_{ref}(x, y)}$$

where I_{ref} is the pixel luminance value from the path tracing reference image. In Table 4.1, we report the RMS value of these errors over the entire image, for MPS sampling and standard photon map sampling. MPS out-performs Photon Mapping in all cases (although by a negligible amount in the Box example). Note that we cannot expect zero error here – even the reference image contains noise.

The Room scene of Figure 4.1 contains about 42,000 primitives. Both the Photon Mapping and MPS images used 4 samples per pixel and 40 final gather rays per sample for estimating indirect illumination. The scene contained user paths specified by hand but no caustics, and we set $\beta = 5^\circ$ in the user path mutation. The proposal probabilities were: P_{UP}=0.1, P_{BD}=0.3, P_{PM}=0.2, P_{CP}=0, P_{LP}=0.2 and P_{LS}=0.2. These, like all our proposal probabilities, were chosen to give roughly equal proportion to each strategy that was useful for the scene. While MPS spent significantly more time than Photon Mapping in sampling photons, it was regained in the faster final gather phase; MPS's smaller number of well-distributed photons improved the performance of nearest neighbor searching in the photon map. We also rendered this scene with Photon Mapping using 6 million photons, which took almost an hour and reduced the noise in the result, but failed to remove the energy bleeding problems and used two orders of magnitude more memory than MPS.

Apart from managing difficult transport paths, a significant advantage of MPS is its ability to store photons only where relevant. Figure 4.5 demonstrates a scene in which Photon Mapping stores almost all photons inside the lantern, where they remain unused when gathering for the wall pixels. In contrast, MPS places almost all samples on the walls of the room. This results in reduced energy bleeding on the table around the box and far less noise in the image overall. These images used 30 samples for each indirect illumination estimate, and 4 samples per pixel. This scene contained no user paths (the important transport paths are not too hard to sample) nor caustics, hence the proposal probabilities were: P_{UP}=0, P_{BD}=0.4, P_{PM}=0.2, P_{CP}=0, P_{LP}=0.2 and P_{LS}=0.2.

Figure 4.6 shows a variant on the Cornell Box scene with complex caustic paths (the right wall and rear ball are mirrors, and the front ball is glass). We used ten user paths in this scene, five for each caustic under the ball. These were perturbed using $\beta = 1^\circ$ for segments between the light and mirror wall, and $\beta = 5^\circ$ for segments from the light direct to the glass ball. We set P_{UP}=0.1, P_{BD}=0.3, P_{PM}=0.2, P_{CP}=0.12, P_{LP}=0.08 and P_{LS}=0.2. Photon Mapping requires many photons to resolve the small caustic due to light bouncing off the mirror through the glass ball. Furthermore, the mirror wall subtends a large area at the light, so it is difficult to concentrate photon sampling toward the caustic producing region, and caustic photons sparsely stored on the rear wall cause excess noise due to their high power. Even with more photons, the caustic is not as good as that from MPS.

4.5.1 Limitations and Extensions

MPS is slower per photon than standard Photon Mapping, but a greater proportion of the stored photons are typically useful. The increase in per-photon cost is because more terms must be evaluated to determine the acceptance probability for each candidate. A path tracing phase is also required and its cost should be amortized over the stored photons. However, the significant improvement in photon distribution achieved with MPS allows for fewer photons overall and typically reduces the cost of the final gather, giving better images for a given computational effort. We have also lost the view invariance of standard photon map construction, as would any method using

visual importance. If the viewer's path were known, the eye location could be a variable included in the sampling process, just as locations on an area light source can vary.

Samples from a Metropolis-Hastings algorithm are correlated due to the Markov process, so the chain needs some time to explore the space adequately, whereas independent particles traced from the light will show no spatial correlation, and can be stratified across the light surface and outgoing direction. This may be important in scenes with very few photons. Parallel Markov chains could be used to generate samples, which would improve the distribution of samples over very short runs. We found this made no difference to the results for the photon counts required in our scenes.

Alternate methods could be used to sample paths, such as bidirectional path tracing or path tracing from the eye. These would be simpler to implement and less computationally expensive, but lack the ability of MPS to exploit correlation in power between neighboring paths. A production system should support multiple algorithms for populating photon maps and share the final gather code and many other modules, including those for ray-tracing and BRDF sampling. Our system is built this way.

We store photons only at a single point along a sampled path — the point most relevant to a final gather operation. However, other points along the path may also be useful, as is the case in the Box scene where any diffuse surface point may be called upon to compute a radiance estimate. We chose not to store additional points because of the memory overhead and the energy bleeding problem. An alternative is to use an importon map to measure the visual importance of surface points, and store photons at any sufficiently important point along the path [50]. This would probably reduce the number of iterations required for MPS on simple scenes, at the cost of an importon map construction phase.

The target PDF we use, f_{eye}, considers all paths that carry power from the lights to the image as important. We could support other forms of importance, such as perceptual metrics or shading discontinuities, simply by modifying the $W_{eye}(\overline{\mathbf{x}})$ component of f_{eye}. The only potential downside would be an increase in the variability of power stored at the photons, $\Phi^{(j)}$, which can increase noise in the final image.

The user path proposal can be used, unmodified, for Metropolis Light Transport (Figure 4.3). Its impact is even greater because the variance in MLT is not disguised by the final gather operation. Conversely, MLT offers a variance reduction technique that we did not implement: the brightness of image pixels is estimated in a first pass and used to modify the path probabilities to make all pixels equally probable. This could be implemented in MPS through importon maps that modify the probability of paths, but it may result in large variance in photon power. Finally, our work could be extended to atmospheric scattering by combining Photon Mapping for participating media [46] with Pauly et al.'s [71] MCMC sampler.

The photon mapping algorithm is one of the most important global illumination approaches and is widely used in industry. One disadvantage of the current photon mapping method is that there are many scenes for which an enormous number of photons must be traced in order to have enough of them in the right places to get good results during final rendering. MPS solves that problem by using Metropolis sampling to create photons that are guaranteed to contribute to the final gathering. Since MPS and traditional photon mapping share the same photon data structure and final gathering phase, it is relatively easy to incorporate the MPS method into an existing photon mapping system: we only need to replace the part for generating photons. Figure 4.8 shows how MPS can fit in the traditional rendering pipeline by modifying the photon generating phase in photon mapping.

4.6 Conclusion

Metropolis Photon Sampling succeeds in generating photon map samples that meet the needs of the final gather phase, without wasting storage or computation time on unnecessary photons. It achieves this by sampling only over light transport paths that reach the image, and storing photons only at appropriate points along the path. The photon distribution that results has more photons that contribute to visually important locations, and fewer in irrelevant places. This not only improves estimates from the map due to higher photon density, but also reduces the chance that inappropriate photons will be used and hence reduces energy bleeding artifacts. At the same time, MPS allows

users to supply information to the sampler in the form of important paths, something not achievable in most Monte Carlo algorithms.

The new sampler is best suited to scenes in which only a small portion of the lights' power arrives in visually important areas. Our method does not require any modification to the final gather phase of photon mapping, so it can be used in conjunction with a standard particle tracing sampler. Depending on the scene, one or other sampler could be used, but there is nothing preventing the use of both methods to fill the same map in scenes with multiple light sources that contribute differently to the image. Furthermore, any improvements to the final gather phase of Photon Mapping apply equally well to Metropolis Photon Sampling.

Figure 4.3: An example of variance control due to the user path proposal strategy. Top is the image rendered with no user paths, while center is the result when the user specified ten paths passing through the doorway. Bottom are zooms of the wall intersection and table regions, with no user paths on the left and user paths on the right. These are MLT images that directly visualize the sampled light paths. The improvements after a final gather, while present, are less apparent.

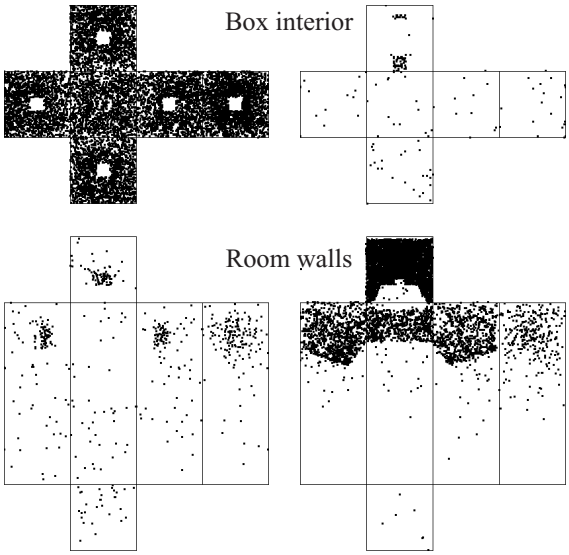

Figure 4.4: The photon distributions for the Jack-o-Lantern scene. Left column is Photon Mapping, while right column is MPS. The top row shows the interior of the box containing the light, while the lower row is the interior of the room. For effective gathering, most samples should be in the room, as is the case for MPS.

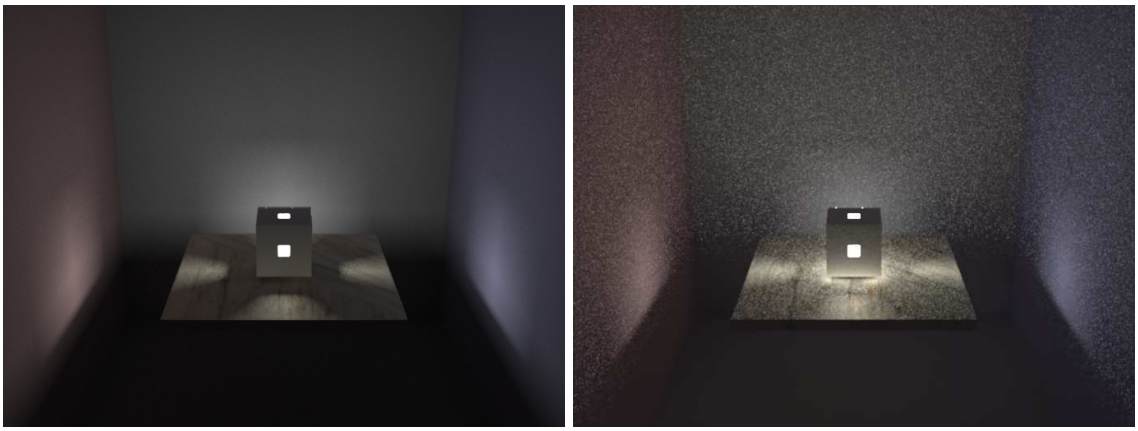

Figure 4.5: A Jack-o-Lantern scene demonstrating MPS's efficient placement of samples. The Photon Mapping scene (right) stores excess photons inside the box and an insufficient number on the walls of the room, resulting, respectively, in significant energy bleeding around the base of the box on the table and noise throughout the image.

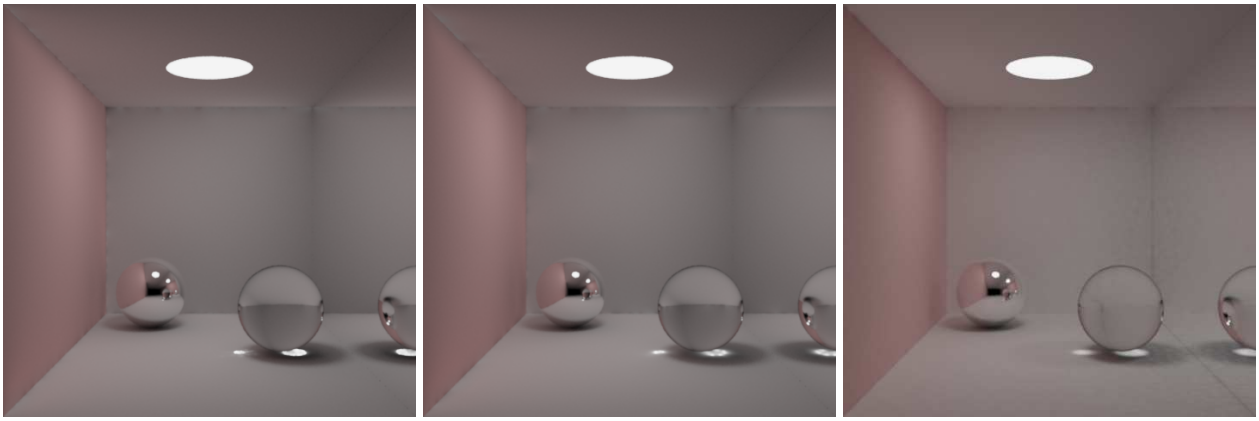

Figure 4.6: The Box scene has a mirror ball at the rear and a mirror right wall, while the front ball is glass. The left image included ten paths specified by the user: five contribute to the large caustic under the glass ball, while the others bounce off the mirror and through the ball to contribute to the smaller caustic. The center scene had no user paths, and consequently the caustics show high variance. Right is a Photon Mapping image of the Box scene computed in equivalent time. The large number of photons cast to resolve the small caustic result in slightly greater noise in the right-rear of the box.

Figure 4.7: Reference images for the scenes in the paper, generated using path tracing.

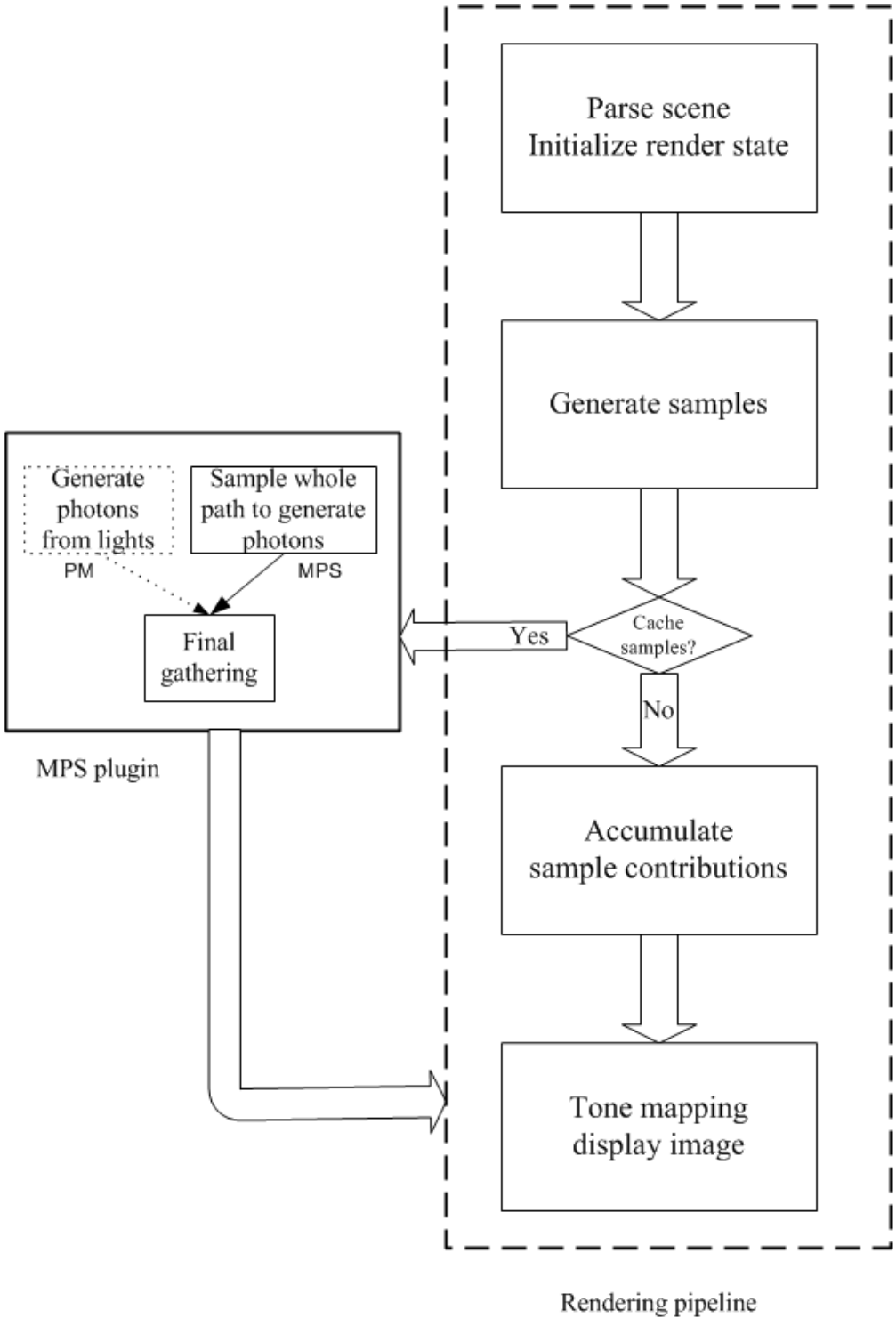

Figure 4.8: MPS's place in the physically based rendering pipeline. We only need to replace the sampling phase in traditional photon mapping with the sampling phase in MPS.

Chapter 5

Population Monte Carlo Rendering

Monte Carlo integration methods offer the most general solution to physically accurate lighting simulation. For production applications, algorithm efficiency is of primary concern: image noise (variance) must be low at practical computation times. We present sampling techniques that significantly improve rendering efficiency; for image-plane sampling, hemispheric integrals, and global illumination. Each is derived using the population Monte Carlo sampling framework, which is a technique that adapts sampling distributions over time and enables sample re-use, all with theoretical guarantees on error and little computational overhead.

PMC algorithms iterate on a population of samples. In our simplest sampler, for image-plane sampling (PMC-IP), the population is a set of image-plane locations (i.e., pixels). The population is initialized in some way, say using stratified sampling; and PMC-IP generates an image. Any information available at this stage can then be used to adapt a *kernel function* that produces a new population. In image-plane sampling, the perceptually-weighted variance in the intermediate images is used to construct the kernel function, resulting in more image plane samples in regions of high variance. The procedure is then iterated: sample, adapt, sample, ... The result is an unbiased algorithm.

In the case of direct lighting, or hemispheric integrals in general, importance sampling [73] is the primary variance reduction tool. However, a poor choice of importance function can *increase* variance, and, moreover, the best importance function can vary throughout a rendering depending on such things as surface properties, lighting configurations and the presence of shadows. For

example, the ideal importance function for a semi-gloss surface depends on whether the primary lobe points toward a light source, or the surface is in shadow, or neither. These configurations vary over a surface and they are difficult to discover before sampling begins, yet the choice of importance functions is typically made once and remains fixed. PMC for hemispheric integrals (PMC-HI) improves sampling efficiency by dynamically choosing importance functions based on information gathered during rendering.

Sample re-use is another way to reduce variance. Most rendering algorithms produce independent samples, so if a sample locates a small but important region of the domain, the information is lost to other samples. Markov chain Monte Carlo algorithms for global illumination, such as Metropolis Light Transport [103] and Energy Redistribution Path Tracing [13], enable sample re-use by mutating existing samples into new ones, but the choice of good mutation strategies is non-trivial and has a major impact on image quality. PMC path tracing (PMC-PL) exploits information from important samples through re-use, with a mutation process that is adapted on-the-fly. The resulting algorithm is self-tuning to a large extent.

Population Monte Carlo is a general purpose framework with many variants. The challenge in applying it to rendering lies in the small sample counts, hard-to-evaluate distributions, and visual sensitivity to noise. Our contribution is three specific tools for rendering that use the framework:

- An **Image-Plane Sampler**, PMC-IP, that adapts to guide samples to perceptually high variance image regions, is cheap to compute, maintains stratification, and is unbiased.

- An **Hemispheric Integral Sampler**, PMC-HI, that adjusts the sampling directions used to evaluate hemispheric integrals at a point and supports a variety of importance functions acting together. We can, for instance, avoid over-sampling a light source from a surface point within its shadow, or a BRDF specular lobe that makes no contribution. Furthermore, we can guide samples toward important illumination directions found by previous samples, without adding bias.

- **PMC Path Tracing**, PMC-PT, that adapts the amount of energy redistribution at different pixels and the area over which energy is redistributed. For example, pixels near a sharp

shadow boundary will not attempt to widely distribute energy, while those in a smooth diffuse image region will distribute over a wide area.

We include results comparing each algorithm to existing approaches, and discuss other rendering problems that are likely to benefit from the approach. We find that PMC-based algorithms improve efficiency by a factor of 2 to 5 over existing methods.

5.1 Related Work

Here we focus on three specific areas of related work: adaptive image-plane sampling, sampling for irradiance integrals, and methods that re-use samples. For an overview of Monte Carlo rendering in general, see Pharr and Humphreys [73].

Typically, adaptive image-plane algorithms perform a first pass with a small number of samples per pixel and use the resulting values to label pixels as adequately sampled or in need of further refinement [32]. The algorithm then iterates on the pixels requiring more samples. However, the labeling of pixels based on an initial sample introduces bias [51], a problem when physically accurate renderings are required. We develop an unbiased, easy to implement method.

Many metrics have been proposed for the test to trigger additional sampling. Lee et al. [57] used a sample variance based metric. Dippé and Wold [17] estimated the change in error as sample counts increase. Painter and Sloan [70] and Purgathofer [75] used a confidence interval test, which Tamstorf and Jensen [95] extended to account for the tone operator. Mitchell [65] proposed a contrast based criterion because humans are more sensitive to contrast than to absolute brightness, and Schlick [83] included stratification into an algorithm that used contrast as its metric. Bolin and Meyer [6], Ramasubramanian et al. [76], and Farrugia and Péroche [29] used models of human visual perception, of which we use a variant. Most recently, Rigau et al. [78, 79] introduced entropy-based metrics.

Our algorithm views the image plane as a single sample space for the purposes of sampling. Dayal et al. [16] took a similar view in the context of frameless rendering. They used a variance-based metric to control a kD-tree subdivision where samples are drawn uniformly within each

adaptively sized cell of the subdivision. Stokes et al. [87] also used a global approach with their perceptual metric.

There is a large body of work on computing irradiance integrals (direct lighting), mostly concerned with importance sampling functions. Veach's thesis [100] provided a good description of the basic methods and analysis of variance. Importance functions have most commonly been based on surface BRDFs (see Pharr and Humphreys [73] for an overview of these), or light sources [85, 1]. Recent advances include wavelet-based importance functions for environmental lighting [12], and resampling algorithms [7, 94] that avoid visibility queries for samples that are likely to be unimportant. However, the former is applicable only to environment maps, while the latter throws away samples and still requires a-priori choice of importance functions. No existing importance sampling approach for irradiance integrals offers adaptable importance functions.

Work on adaptive PDFs for importance sampling has focused on path tracing or irradiance caching applications. Dutré and Willems [24] used piecewise linear functions to determine shooting directions out of light sources in a particle tracing application. Dutré and Willems [25] used piecewise constant functions, and Pietrek and Peter [74] used wavelets to build adaptive PDFs for sampling gather directions in path tracing. A diffuse surface and piecewise constant PDF assumption is required to reduce the number of coefficients to a manageable level, and even then very high sample counts are required. It is important to note that a bad approximation can *increase* variance. Lafortune and Willems [56] used a 5D tree to build an approximation to radiance in the scene, and then used it for importance sampling in a path tracing framework. The same problems with sample counts and approximation errors arise in their work. Our algorithm works with arbitrary BRDFs and uses a low-parameter adaptive model to minimize the sample count required to control adaptation.

Adaptive algorithms have also been suggested for shadow computations. Ward [104] proposed an algorithm for scenes with many lights, where shadow tests for insignificant lights are replaced by probabilistic estimates. Ward's approach works best with many light sources (tens or hundreds) while our technique works best with few sources. Ohbuchi and Aono [68] adaptively sample an

area light source (which introduces bias). They achieve good stratification by employing quasi-Monte Carlo (QMC) techniques to place the samples, a technique we also use.

Sample re-use via Markov chain Monte Carlo (MCMC) algorithms is a powerful means of exploiting hard-to-find light transport paths in global illumination. Metropolis Light Transport [103] was the first algorithm to use this approach, but very large numbers of samples are required, and stratification is difficult. Energy redistribution path tracing (ERPT) attempts to address this problem by starting with a well-stratified set of initial samples and locally redistributing energy using MCMC. The noise-reduction techniques they propose introduce bias. Our PMC path tracing algorithm automatically adapts parameters in an ERPT-like algorithm and is unbiased.

5.2 Population Monte Carlo (PMC)

The population Monte Carlo algorithm [8] is an iterated importance sampling method with dynamically adaptive importance functions which approach the target distribution with the iterations. We outlined a general PMC algorithm in Section 2.6.2.

Several steps are required to apply PMC to rendering problems:

- Decide on the sampling domain and population size. Computational concerns and stratification typically drive the choice of domain. In the image-plane case, working on a discrete pixel domain rather than a continuous one makes stratification simpler to implement and sampling more efficient. We discuss the choice of population size in the context of each algorithm, and later in the discussion.

- Define kernel functions and their adaptation criteria. This is the most important task, and we give examples for our applications and suggest some general principles in the discussion. For rendering applications, two key concerns are the degree to which the kernel supports stratification and whether it works with a small population size (as low as 4 in our hemispheric integrals sampler).

- Choose the techniques for sampling from the kernel functions and the resampling step. The deterministic sampling we use significantly reduces variance, much like stratification.

The following sections describe each of our samplers in detail, before we conclude with results and a general discussion on PMC for rendering problems.

5.3 PMC-IP: Image-Plane Sampling

Physically based rendering algorithms compute the intensity, $I(i,j)$, of each pixel (i,j), by estimating the integrals:

$$I_{i,j} = \int_{\mathcal{I}} W_{i,j}(\mathbf{u}) L(\mathbf{x}, \omega) d\mathbf{u} \qquad [5.1]$$

where $\mathcal{I}$ is the image plane, $W_{i,j}(\mathbf{u})$ is the measurement function for pixel (i,j) – non-zero if $\mathbf{u}$ is within the support of the reconstruction filter at (i,j) – and $L(\mathbf{x},\omega)$ is the radiance leaving the point, $\mathbf{x}$, seen through $\mathbf{u}$ in the direction $-\omega$, determined by the projection function of the camera. We are ignoring depth of field effects, which would necessitate integration over directions out of the pixel, and motion blur, which would require integration over time.

An image-plane sampler selects the image-plane locations, $\mathbf{x}$, in Equation 5.1. For simplicity, assume we are working with a ray-tracing style algorithm that shoots from the eye out into the scene. Adaptive sampling aims to send more rays through image locations that have high noise, while avoiding bias in the final result.

Taking an importance sampling view, given a set of samples, $\{\mathbf{X}_1, \ldots, \mathbf{X}_n\}$ from an importance function $p(\mathbf{x})$, each pixel is estimated using

$$\hat{I}_{i,j} = \frac{1}{n} \sum_{k=1}^{n} \frac{W_{i,j}(\mathbf{X}_k) L(\mathbf{X}_k, \omega)}{p(\mathbf{X}_k)} \qquad [5.2]$$

The source of bias in most existing adaptive image-plane samplers is revealed here. To be unbiased, an importance sampling function must always be non-zero when the target function is non-zero, which is not the case if a pixel is explicitly cut off from sampling ($p(\mathbf{x}) = 0$ within the pixel). Adaptive sampling without bias must avoid decisions to terminate sampling at an individual pixel, and instead look at the entire image plane to decide where a certain number of new samples

will be cast. Every pixel with non-zero brightness must have non-zero probability of being chosen for a sample, regardless of its estimated error.

We also note that Equation 5.2 can be broken into many integrals, one for the support of each pixel. Provided $p(\mathbf{x})$ is known in each sub-domain, the global nature of $p(\mathbf{x})$ is not important.

5.3.1 The PMC-IP Kernel Function

The kernel function is the starting point in creating a PMC algorithm for adaptive image-plane sampling. We need a function that has adaptable parameters, is cheap to sample from, and supports stratification. This can be achieved with a *mixture model* of component distributions, $h_{IP,k}(\mathbf{x})$, one for each pixel:

$$K_{IP}^{(t)}(\mathbf{x}) = \sum_{k=1}^{m} \alpha_k^{(t)} h_{IP,k}(\mathbf{x}), \qquad \sum_{k=1}^{m} \alpha_k^{(t)} = 1.$$

where m is the number of components in the mixture model. Each component is uniform over the domain of a single pixel integral. The parameters to the distribution are all the $\alpha_k^{(t)}$ values, and these change at each iteration, t. We achieve an unbiased result if every $\alpha_k^{(t)} \geq \epsilon$, where ϵ is a small positive constant (we use 0.01). We enforce this through the adaptive process, and the use of ϵ, rather than 0, provides some assurance that we will not overlook important contributions (referred to as *defensive sampling* [42]).

The use of a mixture as the kernel results in a D-kernel PMC [19] algorithm. Sampling from such a distribution is achieved by choosing a component, k, according to the $\alpha_k^{(t)}$, and then sampling from $h_{IP,k}(\mathbf{x})$. The latter can be done with a low-discrepancy sampler within each pixel, giving sub-pixel stratification. Stratification across the entire image plane can be achieved through deterministic mixture sampling, which we describe shortly.

It is important to correctly determine the importance function $p(\mathbf{x})$ in Equation 5.2 for a given pixel. All the samples attributed to a particular pixel come from a single component; all other components have zero probability of producing that pixel, and the $\alpha_k^{(t)}$ sum to one. Hence, $p(\mathbf{x}) = h_{IP,k}(\mathbf{x})$.

Notice that this kernel function is not conditional: $K_{IP}(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)}) = K_{IP}(\mathbf{x}^{(t)})$. Hence, for image-plane sampling we do not include a resampling step in the PMC algorithm because no

samples are re-used. The knowledge gained from prior samples is used instead to adapt the kernel function.

5.3.2 Adapting the PMC-IP Kernel

The adaptation method is responsible for determining the value of each $\alpha_k^{(t)}$ given the populations from previous iterations and any information available from them, such as the image computed so far. Pixels that require more samples should have high $\alpha_k^{(t)}$ for the component that covers the pixel, and we need to define some $\alpha_k^{(t)}$ for every pixel.

An appropriate criterion assigns $\alpha_k^{(t)}$ proportional to an estimate of the perceptually-weighted variance at each pixel. The algorithm tracks the sample variance in power seen among samples that contribute to each pixel. To account for perception, the result is divided by the threshold-versus-intensity function $tvi(L)$ introduced by Ferweda et al. [30]. Normalization also accounts for ϵ.

$$\begin{aligned} \alpha'_k &= \frac{\overline{\sigma}_k^2}{tvi(L_k)} \\ \alpha_k^{(t)} &= \frac{\epsilon}{m} + \frac{(1-\epsilon)\alpha'_k}{\sum_{i=1}^{m} \alpha'_i} \end{aligned}$$

The first iteration of the algorithm samples uniformly over the image plane, so this criteria can always be computed. The left images in Figure 5.2 show an example of an $\alpha_k^{(0)}$ map for a given initial image. The perceptual term in the error image prevents very high errors in both bright regions (a problem with unweighted variance) and dark areas (a problem with luminance-weighted variance).

5.3.3 Deterministic Mixture Sampling

Randomly sampling from the discrete distribution defined by the $\alpha_k^{(t)}$ produces excess noise — some pixels get far more or fewer samples than they should. This problem can be avoided through the use of *deterministic mixture sampling*, DMS, which is designed to give each component (pixel) a number of samples roughly proportional to its $\alpha_k^{(t)}$. Deterministic mixture sampling always gives lower variance when compared to random mixture sampling, as proved by Hesterberg [42].

The number of samples per iteration, N, (the population size) is fixed at a small multiple of the number of pixels. We typically use 4, which balances between spending too much effort on any one iteration and the overhead of computing a new set of kernel parameters. For each pixel, the deterministic sampler computes $n'_k = N\alpha_k$, the target number of samples for that pixel. It takes $\lfloor n'_k \rfloor$ samples from each pixel k's component. The remaining un-allocated samples are sampled from the *residual distribution* with probability $n'_k - \lfloor n'_k \rfloor$ at each pixel (suitably normalized).

Figure 5.1 summarizes the PMC-IP algorithm:

---

1 Generate the initial image

2 **for** $t = 1, \cdots, T$

3 Compute the perceptually-weighted variance image

4 Compute $\alpha_k^{(t)}$ for each pixel k

5 Use DMS to allocate samples according to $\alpha_k^{(t)}$

6 Generate samples from $K_{IP}^{(t)}(\mathbf{x})$ and accumulate in the image

---

Figure 5.1: The PMC-IP algorithm.

5.3.4 PMC-IP Results

Adaptive image-plane sampling can be used in many situations where pixel samples are required and an iterative algorithm can be employed. We have implemented it in the context of direct lighting using a Multiple Importance Sampler (MIS) and for global illumination with path tracing. Other potential applications include bidirectional path tracing and photon-mapping. Algorithms that are not concerned with physical correctness would be better served by a simpler, biased criterion.

Figure 5.2 shows the Buddha direct lighting example. The surface is diffuse with an area light source. Each pixel sample used 8 illumination samples, and the images were rendered at 256×512, with statistics presented in Table 5.1. We introduce the perceptually-based mean squared efficiency

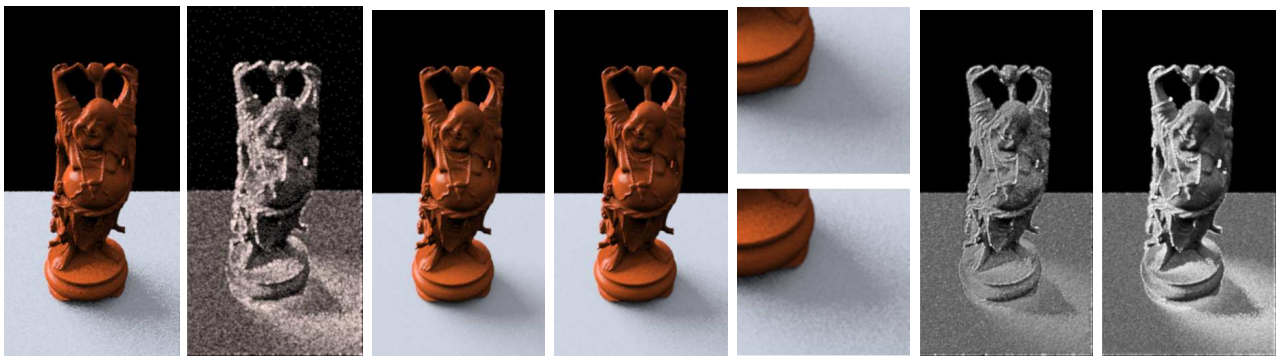

Figure 5.2: A comparison between adaptive and uniform image-plane sampling on a direct lighting example. Leftmost is the initial image for PMC-IP sampling, and the $\alpha_k^{(0)}$ image. The initial image used 2 samples per pixel. The next image is the result of PMC-IP sampling with two iterations at 4spp on average. Center is a 10spp image uniformly distributed. The zooms show the shadow near the Buddha's base (PMC-IP top, uniform bottom). To the right are the corresponding variance images. Note that the variance image for the PMC-IP sampler has few high variance regions, and is lower contrast in general, representing a more even distribution of error.

(P-Eff) metric for comparing algorithms, computed as:

$$Err = \frac{\sum_{pixels} e^2}{tvi(L)}, \quad \textit{P-Eff} = 1/(T \times Err)$$

where e is the difference in intensity between a pixel and the ground truth value, and T is the running time of the algorithm on that image. P-Eff is a measure of how much longer (or shorter) you would need to run one algorithm to reach the perceptual quality of another [73].

The final adaptive image shown is the unweighted average of three sub-images (initial and two iterations). While weighting each sub-image may be helpful, in this context it is not clear that the samples from one iteration are any better than those from another because they all used the same per-sample parameters. We obtained more samples in places that needed it, but not better samples.

The path tracing algorithm differs from a standard version only in how pixel locations are chosen. The improvement due to PMC-IP sampling is more pronounced in this situation because some areas of the image (the caustic, for instance) have much higher error than others due to the difficulty of sampling such paths. In this example (Figure 5.3), we see that PMC-IP sampling with a total of 16spp produces lower error than uniform sampling at 24spp, in 25% less time.

We ran our examples for fixed number of iterations (bounded computation time). If working toward a error bound, then we would continuing iterating the PMC-IP sampler until total error

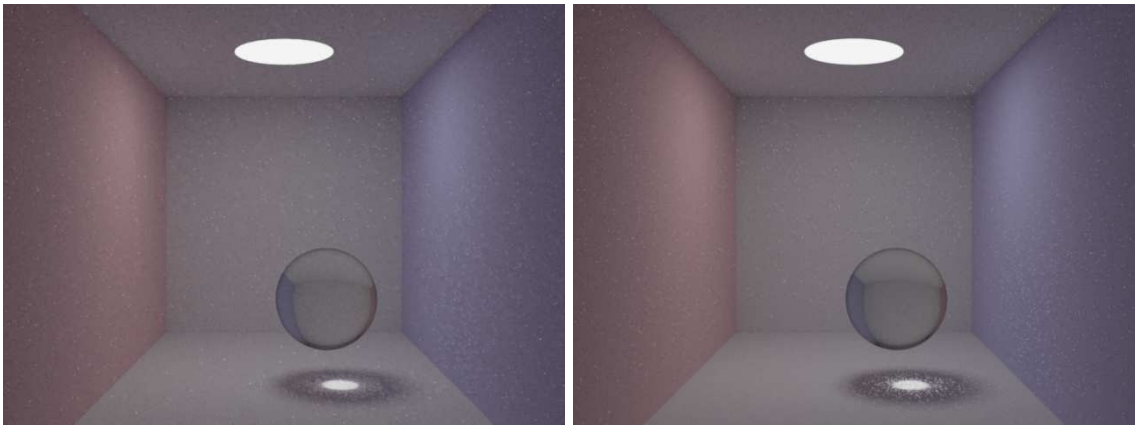

Figure 5.3: A Cornell Box image computed using path tracing with 16spp adaptively sampled on the left and 32spp uniformly distributed on the right. Even with about half less computation time than the uniform image with 32spp, the adaptive image has superior quality around the caustic which is the hardest region to sample.

dropped below a bound. Note that because the PMC-IP sampler evenly spreads variance over the image, an overall image error bound is very unlikely to leave any high-error pixels.

5.4 PMC-HI: Adaptive Hemispheric Integrals Sampling

Hemispheric samplers generate incoming directions, ω', at a surface point, $\mathbf{x}$. One application is in direct lighting, which assumes that the light leaving a surface point, $L(\mathbf{x}, \omega)$, can be evaluated by the following integral, composed of terms for light emitted from and reflected at $\mathbf{x}$:

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_\Omega f(\mathbf{x}, \omega, \omega') d\omega' \qquad [5.3]$$

where $L_e(\mathbf{x}, \omega)$ is light emitted at $\mathbf{x}$, Ω is the hemisphere of directions *out* of $\mathbf{x}$ and $f(\mathbf{x}, \omega, \omega')$ is the light reflected at $\mathbf{x}$ from direction $-\omega'$ into direction ω:

$$f(\mathbf{x}, \omega, \omega') = L_{in}(\mathbf{x}, -\omega') f_r(\mathbf{x}, \omega, \omega') |\cos(\theta')| \qquad [5.4]$$

where $L(\mathbf{x}, -\omega')$ is the light arriving at $\mathbf{x}$ from direction ω', $f_r(\mathbf{x}, \omega, \omega')$ is the BRDF, and θ' is the angle between ω' and the normal at $\mathbf{x}$.

Image	Method	# SPP	T(s)	Err	P-Eff
Buddha	Uniform	10	58.1	0.625	0.027
	PMC-IP	2+4+4	62.4	0.116	0.138
Box	Uniform	16	163	0.545	0.011
	Uniform	32	328	0.255	0.012
	PMC-IP	4+6+6	184	0.182	0.030

Table 5.1: Measurements comparing PMC-IP and uniform image-plane sampling, for equal total sample counts. The Buddha image computed direct lighting with the MIS method, with a total of 8 lighting samples for each pixel sample. PMC-IP sampling improves the perceptual-based RMS error by a factor 5.4 over uniform sampling with only 7.5% more computation time. It corresponds to an improvement in efficiency of 5.01. The Cornell Box images use path tracing to compute global illumination including caustics. Comparing with images of 16ssp, PMC-IP improves the efficiency by a factor of 2.65.

A standard importance sampling algorithm for $L(\mathbf{x}, \omega)$ samples directions, $\{\omega'_1, \ldots, \omega'_n\}$, out of $\mathbf{x}$ according to an importance function, p, and computes the estimate:

$$\hat{L}(\mathbf{x}, \omega) = \frac{1}{n} \sum_{i=1}^{n} \frac{f(\mathbf{x}, \omega, \omega'_i)}{p(\omega'_i)} \qquad [5.5]$$

The variance of this estimator improves as p more closely approximates f, and is zero when p is proportional to f.

In the local direct lighting situation, one common choice for p is proportional to $L_{in}(\mathbf{x}, -\omega') f_r(\mathbf{x}, \omega, \omega') |\cos(\theta')|$ or a normalized approximation to it. An alternative is to break the integral into a sum over individual light sources and sample points on the lights to generate directions [73, §16.1]. In an environment map lighting situation, the wavelet product approach of Clarberg et al. [12] currently provides the best way to choose p. However, none of these individual importance functions behaves well in all cases.

Figure 5.4 demonstrates the various difficult cases for importance sampling. The floor consists of a checker based pattern with diffuse and glossy squares (with two types of gloss settings). There are two lights, one large and one small. In pixels that image diffuse squares, an importance function based on the lights is best. In highly glossy pixels that reflect the large light, BRDF sampling is

best. For glossy pixels that do not reflect a light, sampling from the light is best, and rough glossy pixels benefit from both BRDF and light sampling. But we have no way of knowing this a-priori, and most practitioners would use BRDF sampling. In rough glossy regions that reflect only one light, sampling from the other light is wasteful, but again most algorithms would sample equally or according to total emitted power.

Multiple Importance Sampling (MIS) addresses many of these problems by trying several importance functions and combining their results. While this does very well at reducing variance, it is wasteful in cases where one of the importance functions is much better than the others and could be used alone. Other techniques assume knowledge of which strategy will dominate where.

PMC-HI is a sampler that generates directions out of a point by adapting a kernel function to match the integrand of interest — $L_{in}(\mathbf{x}, -\omega')f_r(\mathbf{x}, \omega, \omega')|\cos(\theta')|$ in the direct lighting case. For example, the leftmost images in Figure 5.6 indicate the relative usefulness of different importance functions at each pixel. Furthermore, the PMC framework enables important samples from one iteration to guide sampling in subsequent iterations.

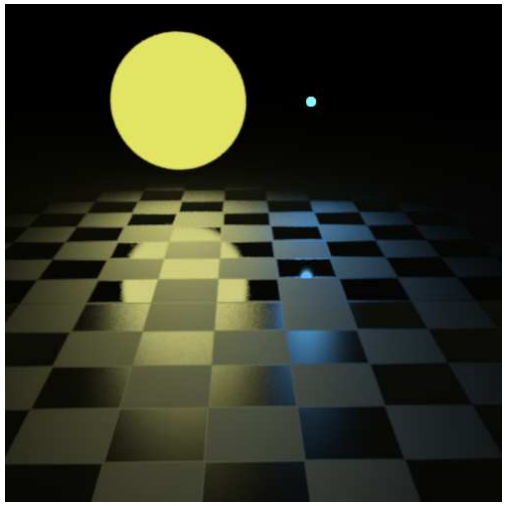

Figure 5.4: A scene constructed to demonstrate how the optimal sampling strategy varies over an image. The checkerboard contains diffuse and glossy squares, with near-pure specular reflection toward the back and rougher toward the front. There are two light sources.

5.4.1 The PMC-HI Kernel Function

Each direct lighting estimate takes place at a single surface point and is only one small step in a larger computation. The same surface point, and hence the same target function, f_r, essentially never re-appears. We choose to adapt on a per-estimate basis, which avoids the need to store information about the adaptation state at surface points and interpolate to find information at new points. Hence, the number of samples on which to base adaptation is low, certainly less than 100 and less than 10 in some of our examples.

A mixture distribution of a few candidate importance functions is a good starting point. At least one such component is likely to be a good approximation to f_r, and we expect to adapt to use that function most often. To catch cases where good sampling directions are hard to find, we include a component, h_{cone}, that samples based on important sample directions from the previous iteration. For one light, the mixture is

$$\begin{aligned} K_{IR}^{(t)}(\omega^{(t)}|\mathbf{d}^{(t)},\beta^{(t)}) &= \alpha_{BRDF}^{(t)} h_{BRDF}(\omega^{(t)}) \\ &\quad + \alpha_{light}^{(t)} h_{light}(\omega^{(t)}) \\ &\quad + \alpha_{cone}^{(t)} h_{cone}(\omega^{(t)}|\mathbf{d}^{(t)},\beta^{(t)}) \end{aligned} \qquad [5.6]$$

There is one term for the BRDF-based importance function, one for a light (or one per light for multiple lights) and the cone perturbation function. The cone function samples a direction uniformly within a cone of directions with axis $\mathbf{d}^{(t)}$ and half-angle $\beta^{(t)}$, which is set based on the population in the previous iteration. It is particularly useful for situations like partial shadowing where previous samples that found visible portions of the light generate more samples that also reach the light. Figure 5.5 shows the mixture PDF and their component PDFs.

The population in PMC-HI is a set of sample directions out of the surface point we are estimating. The population size must be large enough to obtain reasonable estimates for the $\alpha_k^{(t)}$ values at each iteration, but not so large as to increase computation times significantly. We typically use $n = 2m$, where n is the population size and m is the number of mixture components. This is a sufficient size to see the benefits of adaptation, as the result in Figure 5.6 demonstrates.

5.4.2 Adapting for PMC-HI

An initial population of n_0 samples, $\left\{\Omega_1^{(0)}, \ldots, \Omega_{n_0}^{(0)}\right\}$, is generated using $\alpha_{cone}^{(0)} = 0$ and the other $\alpha_k^{(0)}$ equal and summing to one. Deterministic mixture sampling is used to select the number of samples from each component. Each sample is tagged with the mixture component that was used to generate it, and their importance weights are computed:

$$w_i^{(0)} = \frac{f(\mathbf{x}, \omega, \omega')}{K_{IR}^{(0)}(\omega^{(0)})} \qquad [5.7]$$

There is no resampling step for direct lighting. The sample size is so small that resampling tends to unduly favor high-weight directions at the expense of others, thus reducing the degree to which sampling explores the domain. Instead, the cone mixture component is used to incorporate the information from previous samples.

The new component weights, $\alpha_k^{(1)}$, can now be determined, along with the $\mathbf{d}^{(1)}$ and $\beta^{(1)}$ parameters for $h_{cone}(\omega^{(1)}|\mathbf{d}^{(1)}, \beta^{(1)})$. The cone direction $\mathbf{d}^{(1)}$ is found by taking a weighted average of the $t = 0$ population samples, with weights $w_i^{(0)}$. The cone size is set to the standard deviation of those samples. The component weights are set based on the sample importance weights:

$$\alpha_k^{(t)} = \frac{\sum_{i \in \mathcal{S}_k} w_i^{(t-1)}}{\sum_{j=1}^{n} w_j^{(t-1)}} \qquad [5.8]$$

where $\mathcal{S}_k$ is the set of samples that were generated using component k. In the first iteration there is no sample from the cone perturbation, so we set $\alpha_{cone}^{(1)} = 0.2$ and adjust the other α's by a factor of 0.8 to make them all sum to one.

We now begin the next iteration. A new set of samples is generated using deterministic mixture sampling from the kernel $K_{IR}^{(t)}(\omega^{(t)}|\mathbf{d}^{(t)}, \beta^{(t)})$, weights are computed, and the kernel function is updated based on the weights. To form the estimate, use Equation 2.32 with each sample, $\Omega_i^{(t)}$, weighted by $w_i^{(t)}$ from Equation 5.7.

5.4.3 Adaptive Direct Lighting Results

We present results on two examples of PMC-HI for direct lighting: the Checker scene (Figure 5.7) and a plant rendering with complex shadows and glossy BRDFs (Figure 5.8). Timing and

Image	Method	# SPP	T(s)	Err	P-Eff
Checks	MIS	12	46	0.379	0.057
	MIS	48	183	0.153	0.035
	PMC-HI	12	54	0.146	0.127
Plant	MIS	27	53	0.403	0.047
	PMC-HI	27	64	0.128	0.122

Table 5.2: Measurements comparing PMC-HI sampling with MIS, for equal total sample counts. In all cases we used a single direct lighting estimate for each pixel. For Checks scene, PMC-HI improves the efficiency by a factor 2.21, which takes four times more samples for uniform MIS to reach the approximately same perceptual based variance (Err). The efficiency gain for the Plant scene is 2.60.

error comparisons with MIS (the best of several existing algorithms we tried on these scenes) appear in Table 5.2. The checkerboard image resolution is 500×500 and the plant image is 720×405.

The Checker scene clearly demonstrates that adaptation is a stable process that finds a good kernel function, or evenly weights the components if none dominates (Figure 5.6). The cone component is not particularly helpful in this case because visibility is simple. Timing results show that PMC-HI halves the variance for a given sample count compared to MIS, with only 20% more computation time. The Plant scene demonstrates the usefulness of the cone function in partially shadowed regions. It shows major improvement in the soft shadow boundaries on the table.

5.5 PMC Path Tracing

PMC Path Tracing (PMC-PT) is an algorithm motivated by energy redistribution path tracing (ERPT) [13] that adaptively selects pixels for redistribution, and can also adapt algorithm parameters. ERPT as originally proposed traces a path into the scene from each pixel, using path tracing to form complete light transport paths from the eye to a light. For each pixel, the path is used as the initial state for a Markov chain Monte Carlo (MCMC) sample chain that redistributes the path's energy to nearby pixels and finds additional light paths. The intuition is that different pixels will

find different initial paths, and the information can then be conveyed to neighboring pixels through the Markov chain. Due to space limitations, we cannot discuss ERPT in detail; readers are referred to the original paper.

ERPT uses a constant length chain for every pixel, regardless of how much energy the initial path carries or how much it differs from its neighbors. This is sub-optimal — some pixels have high energy initial paths that take longer to redistribute, while others are in a neighborhood where most light transport paths are similar and redistribution achieves nothing. To address the former problem, Cline et al. [13] designed filters that introduce bias into the calculation, making the image darker than it should be.

Our PMC-PT algorithm uses the same basic premise as ERPT: high-energy paths should be mutated to distribute the information they carry to neighboring pixels. The sample population is a set of light transport paths through the scene. The kernel function mutates these paths to create new paths. The resampling step removes low energy paths in regions of low image variance and duplicates high-energy paths in regions of high variance. As a result, work is focused on the important transport paths.

5.5.1 PMC-PT Kernel Function

The kernel function for PMC-PT is a conditional kernel, $K^{(t)}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{X}}_i^{(t-1)})$, that generates sample i in iteration t, $\tilde{\mathbf{X}}_i^{(t)}$, given sample i in iteration $t-1$, $\tilde{\mathbf{X}}_i^{(t-1)}$ (see Figure 2.4). Again we use a mixture distribution:

$$\begin{aligned} K^{(t)}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{x}}^{(t-1)}) &= \alpha_5^{(t)} h_{lens}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{x}}^{(t-1)} : 5) \\ &\quad + \alpha_{10}^{(t)} h_{lens}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{x}}^{(t-1)} : 10) \\ &\quad + \alpha_{50}^{(t)} h_{lens}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{x}}^{(t-1)} : 50) \end{aligned} \quad [5.9]$$

Each component $h_{lens}(\tilde{\mathbf{x}}^{(t)}|\tilde{\mathbf{x}}^{(t-1)} : s)$ performs a *lens perturbation* from ERPT, described in detail by Cline et al. [13]. The perturbation takes the existing path and moves the image point through which it passes. In our case, the new lens location is uniformly sampled within a square of half-side-length s, a parameter to the kernel. The remainder of the path is reconstructed to pass through

the new image point while retaining as much as possible of the existing path. In the original ERPT work, the size of the perturbation was a parameter to be fixed at startup. We use three different sized perturbations in the mixture. The large perturbation is effective for redistributing information over a wide area, while the smallest is best for image regions where illumination is changing quickly.

We could also include a component for the *caustic perturbation* from ERPT and Metropolis Light Transport, which would improve performance in scenes with significant caustic transport. In practice, we found this to be unnecessary because the smallest lens perturbation achieved a similar effect.

5.5.2 Resampling and Adapting

A startup phase of PMC-PT creates an initial image using path tracing at about 3spp. Not all the generated paths are valid – they are terminated by Russian Roulette before a point ever sees the light. Of those that are valid, we take every kth one for the initial population, where k is chosen to give us the desired population size.

In PMC-PT we resample and adapt the mixture component weights at a lower frequency than we iterate the kernel:

T_R is the number of kernel iterations per resample step. We resample less often because it is moderately expensive and there is no advantage to adapting at every iteration. After exploring several values for T_R, we found a wide range of values to be effective. The optimal value depends on the population size and the relative cost of kernel perturbations compared to resampling.

The resampling step achieves three purposes: samples that need more energy redistribution are carried forward to the next round, the information about which samples are chosen during resampling guides the adaptation, and it provides an opportunity to add some completely new paths into the population. The proportion of samples that survive the resampling from any given component, the *survival rate*, indicates the usefulness of that component for sampling, and hence are used to set the $\alpha_k^{(s)}$.

Resampling is a standard technique that forms a discrete distribution over the existing sample set and then samples with replacement from the old population to generate the new one. We use

deterministic sampling (the residual method described above) to sample the new population. The resampling probabilities are the importance weights, $w_i^{(T_R)}$.

Each sample was tagged with the kernel mixture component that generated it. After resampling, we set the $\alpha_k^{(s)}$ mixture weights to the proportion of surviving samples that were generated with component k.

To add completely new paths, we resample fewer paths from the population and create the remaining paths using the original path tracing algorithm, as we did to create paths for the initial sample. The aim of adding new paths is to limit the potential for the resampling to produce many repeats of a few very high weight paths. We could include a component in the mixture to create new paths from scratch, but that limits the extent to which any given path's energy is redistributed because the probability that it would survive more than a few inner loop iterations is low. Adding new paths in this way does not add bias because neither the resampled population nor the new samples are biased, so their union is not biased. In practice, we resample for 70% of the population and generate the remaining 30% from scratch.

After every step in the inner loop of Figure 5.9, we accumulate the weights, $w_i^{(t)}$, to the appropriate pixels to form the image. Computing these weights requires the kernel function probabilities, also called transition probabilities. Cline et al. [13] provide an excellent detailed discussion of the computation of these functions, and PMC-PT uses exactly the same techniques ($K^{(s)}(x|y)$ in our terminology is $T(x \to y)$ in theirs).

5.5.3 PMC-PT Results

We compared PMC-PT with the energy redistribution path tracing (ERPT) algorithm on the Cornell Box scene and a basic Room scene. In both cases we used a population size of 10,000. The Box scene began with 9spp path tracing. ERPT performed 400 iterations on each initial path, while PMC-PT did 10 resampling iterations each with 40 mutation iterations. The scene was rendered at 640×480 resolution. PMC-PT achieves a 45% reduction in RMS error over ERPT, with only 8% more computation time (see Table 5.3). The images (Figure 5.10) demonstrate that PMC-PT expends more effort on the difficult regions – the ceiling, glass ball and caustic – and

Image	Method	Total time (s)	Err	P-Eff
Box	ERPT	203.6	2.013	2.44e-3
	PMC-PT	212.8	1.554	3.02e-3
Room	ERPT	1021	1.434	6.83e-4
	PMC-PT	1132	0.326	27.1e-4

Table 5.3: Measurements comparing energy redistribution path tracing (ERPT) with PMC-PT, for a roughly equal number of mutations. The efficiency gains of PMC-PT are 1.24 and 3.97 for the Box scene and Room scene, respectively.

hence has lower variance in those regions, at the expense of slightly higher variance in other parts of the image. This is a recurring property of both the PMC image plane sampler and PMC-PT: PMC produces a more even distribution of noise, with lower noise levels overall but higher in some parts of the image that are over-sampled with non-adaptive techniques.

The Room scene (Figure 5.11) was rendered at 720×405 and used 16spp to obtain the initial paths. ERPT performed 600 iterations on each initial path, while PMC-PT did 5 resampling iterations each with 120 mutation iterations. Note that for both PMC-PT and ERPT implementations, we did not use the filter in the original ERPT paper to smooth the final image.

5.6 Discussion

The most important parameter in a PMC algorithm is the population size. A small population reduces the number of samples per iteration, which gives more flexibility in the total sample count in an algorithm, but relatively more time is then spent adapting mixture parameters. Furthermore, the quality of the adapted functions is lower because they are derived from less information. Hence, we use small populations only for the hemispheric integrals case, where we aim to keep the total number of samples per estimate low and the kernel function has a very small number of parameters. Larger populations result in more robust adaptation and less overhead, and in general are to be favored. However, if the population is too large the benefits of adaptation are lost as relatively more samples are drawn using a mal-adapted importance function.

In Equation 5.7 we use the full mixture distribution as the importance function, $K(\omega_i')$. This is a form of Rao-Blackwellization, which reduces variance but at the expense of additional computation. The algorithm remains correct if we use only the mixture component from which the sample came, $h_k(\omega'^{(i)})$, and we need not compute the other mixture functions. In some cases the resulting reduction in computation may exceed the increase in noise, but in rendering the greatest cost is usually in obtaining a sample, rather than evaluating its probabilities.

5.6.1 Relationships with Existing Algorithms

The PMC algorithms we have presented can be viewed as generalizations of some existing algorithms:

- MIS is a special case of deterministic mixture sampling. It corresponds to fixing the α_k weights ahead of time, which fixes the number of samples from each function. The MIS balance heuristic results in the same estimator that we use. We improve upon MIS by adapting the weights over time, which avoids wasting samples on unimportant component functions.

- PMC-PT may be viewed as a form of Metropolis light transport with multiple parallel chains (the population), that are allowed to die and split (resampling). The PMC framework places this in a sound statistical setting.

5.6.2 Designing Adaptable Kernel Functions

Many PMC kernels in the literature are mixture models. Mixtures are typically formed by combining several components that are each expected to be useful in some cases but not others. The adaptation step then determines which are useful for a given input. Mixtures allow otherwise unrelated functions to be combined, such as the light area importance function and the BRDF importance function in Equation 5.7. If an environment map was present we could even include the wavelet importance functions of Clarberg et al. [12] in the mixture. Typically, the common rule of choosing importance functions applies here also: when f is a product of several unrelated functions, then a good choice of mixture components is something proportional to each factor.

Other adaptable functions can be used as kernel functions, such as Gaussian distributions parameterized by their standard deviation. Such a choice would be appropriate if a single Gaussian of unknown size was thought to be sufficient even when acting alone, but the ability to simultaneously sample from several functions is lost. The most common reason to use non-mixture kernels is when the aim of PMC is finding the adapted parameters themselves, not the samples, and hence the kernel function is chosen to represent the expected form of the underlying statistical model.

5.6.3 PMC in the rendering pipeline and its limitations

PMC can be easily incorporated into the physically based rendering system pipeline, as seen in the Figure 5.12. The image-plane sampler and direct lighting integrator are common components in many rendering algorithms. PMC-IP sampling can be used as a plugin component for essentially any algorithm that forms light paths through the eye, including the gather phase of photon-mapping, bidirectional path tracing, and irradiance caching. The PMC-HI sampler can be used in any situation where estimates of an integral over the hemisphere are required. Irradiance caching would benefit greatly from a PMC sampler in the computation of each cached value. Photon-mapping could also use a PMC sampler in the final gather, but we expect the improvement to be less apparent.

The most notable limitation of PMC is the high sample counts required when the kernel has many adaptable parameters. This precludes, for instance, using one component per light when there are many lights. Such a strategy would be appealing for efficient sampling in complex shadow situations (some components would see the lights, others wouldn't), but the sample count required to adequately determine the mixture component weights would be too large. Instead we use a single mixture component for all the lights and rely on the cone perturbation component to favor visible lights, but this does not work well if illumination sources are widely spaced.

An alternate approach for integrating functions defined on surfaces is to store the mixture component weights in a surface map and interpolate. This amortizes the cost of adapting over many surface points. We did not explore this possibility, but it offers potential for the multi-light problem

or cases where many light transport paths must be constructed through a scene, such as bidirectional path tracing or photon mapping.

We rely on deterministic mixture sampling to achieve stratification in the image-plane sampler and to a lesser extent in the other samplers. This is not entirely satisfactory. For example, in PMC-PT the mutation kernels are responsible for moving samples around the image plane, and these are not stratified. This could be addressed using techniques similar to those in Metropolis Light Transport, but at some greater complexity.

5.7 Conclusion

The have shown how algorithms for adaptive image-plane sampling, hemispheric integral computations, and energy redistribution path tracing can be derived within a PMC framework. In each case the algorithm learns an effective sampler based on the results from earlier iterations. This alleviates one of the greatest problems in Monte Carlo rendering: the choice of importance functions and other parameters.

PMC is just one approach from the family of iterated importance sampling algorithms [80]. The Kalman filter is another well-known example. Common to these techniques is the idea of sample re-use through resampling and the adaptation of sampling parameters over iterations. Computer graphics certainly offers further opportunities to exploit these properties.

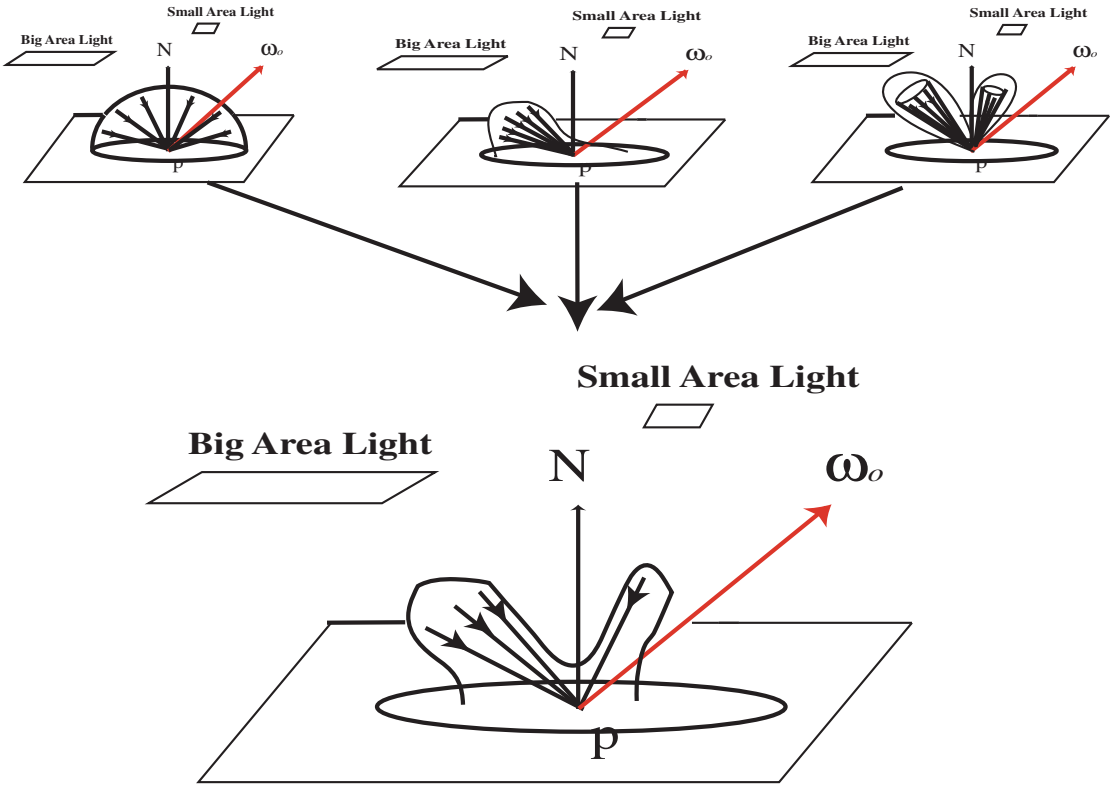

Figure 5.5: Mixture PDF. The top is the BRDF sampling, light sampling, and cone sampling respectively. The bottom is a linear combination of the top three sampling strategies.

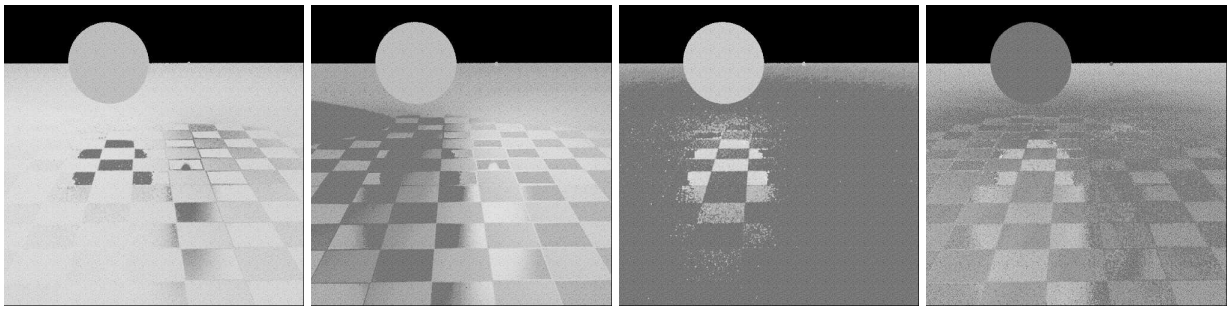

Figure 5.6: These maps show how the mixture component weights for PMC-HI vary over the image, after two iterations. Bright means high weight. From left to right: $\alpha_{L1}^{(2)}$, the left light's weight; $\alpha_{L2}^{(2)}$, the right light's weight; $\alpha_{BRDF}^{(2)}$; and $\alpha_{cone}^{(2)}$, which in this image is of limited use. The large light dominates in regions where no light is seen in a glossy reflection, while the right light is favored in nearby diffuse squares. The BRDF component is favored only when the large light is specularly reflected at a pixel. The images are quite noise-free for such small sample counts (16 total samples per estimate), indicating that the adaptation mechanism converges to a consistent result.

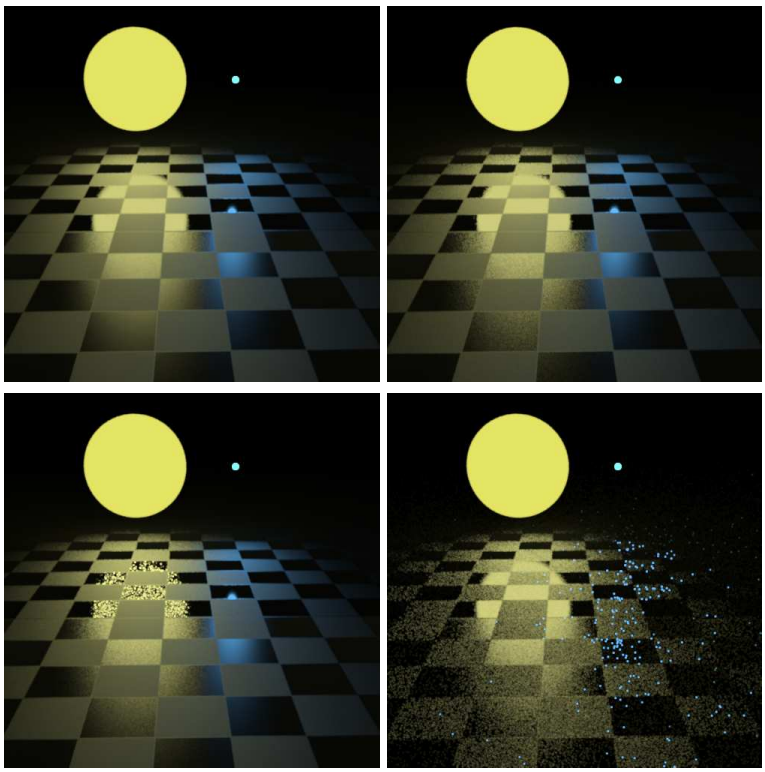

Figure 5.7: Checker images generated from different algorithms with the same number of samples. The PMC-HI image (top-left) is better overall than the MIS image(top-right), especially in the glossy region in front of the big area light where neither sampling from light nor sampling from BRDF works well. Light sampling (bottom-left) does poorly in the specular region in front of the big area light, while the BRDF image (bottom-right) appears very noisy at the diffuse surface.

Figure 5.8: An image involving complex soft shadows and glossy surfaces. Top-left is PMC-HI sampling, and top-right is MIS with equal total sample count. Note the significant improvement in the soft shadows achieved with PMC-HI, shown in the zoomed images at the bottom (PMC-HI left, MIS right).

---

1 **for** $s = 1, \cdots, T$

2 determine $K^{(s+1)}(x^{(t+1)}|x^{(t)})$

3 **for** $t = 1, \cdots, T_R$

4 **for** $i = 1, \cdots, n$

5 generate $\hat{X}_i^{(t)} \sim K^{(s)}(x|X_i^{(t-1)})$

6 $w_i^{(t)} = f(\hat{X}_i^{(t)})/K^{(s)}(\hat{X}_i^{(t)}|X_i^{(t-1)})$

7 resample for the new population

Figure 5.9: The PMC-PT iteration loop.

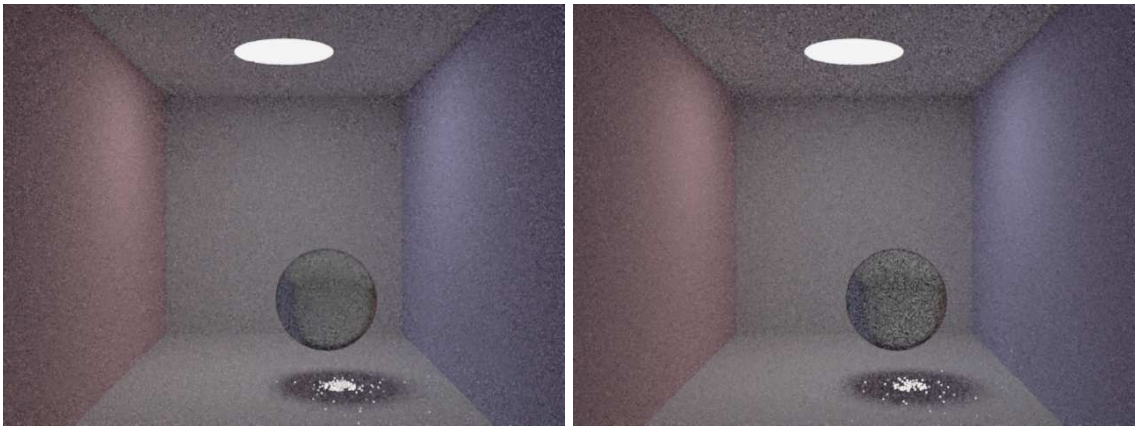

Figure 5.10: A Cornell Box image computed using PMC-PT on the left and ERPT on the right. The improvement with PMC-PT is most evident in the caustic, the glass ball, and the ceiling. We did not use the biased filters of Cline et al. [13], hence the remaining very bright spots.

Figure 5.11: A Room scene computed using PMT-PT at top and ERPT below. PMC-PT has fewer artifacts overall. PMC-PT improves over ERPT by sharing more information among paths and better re-using the high contribution paths.

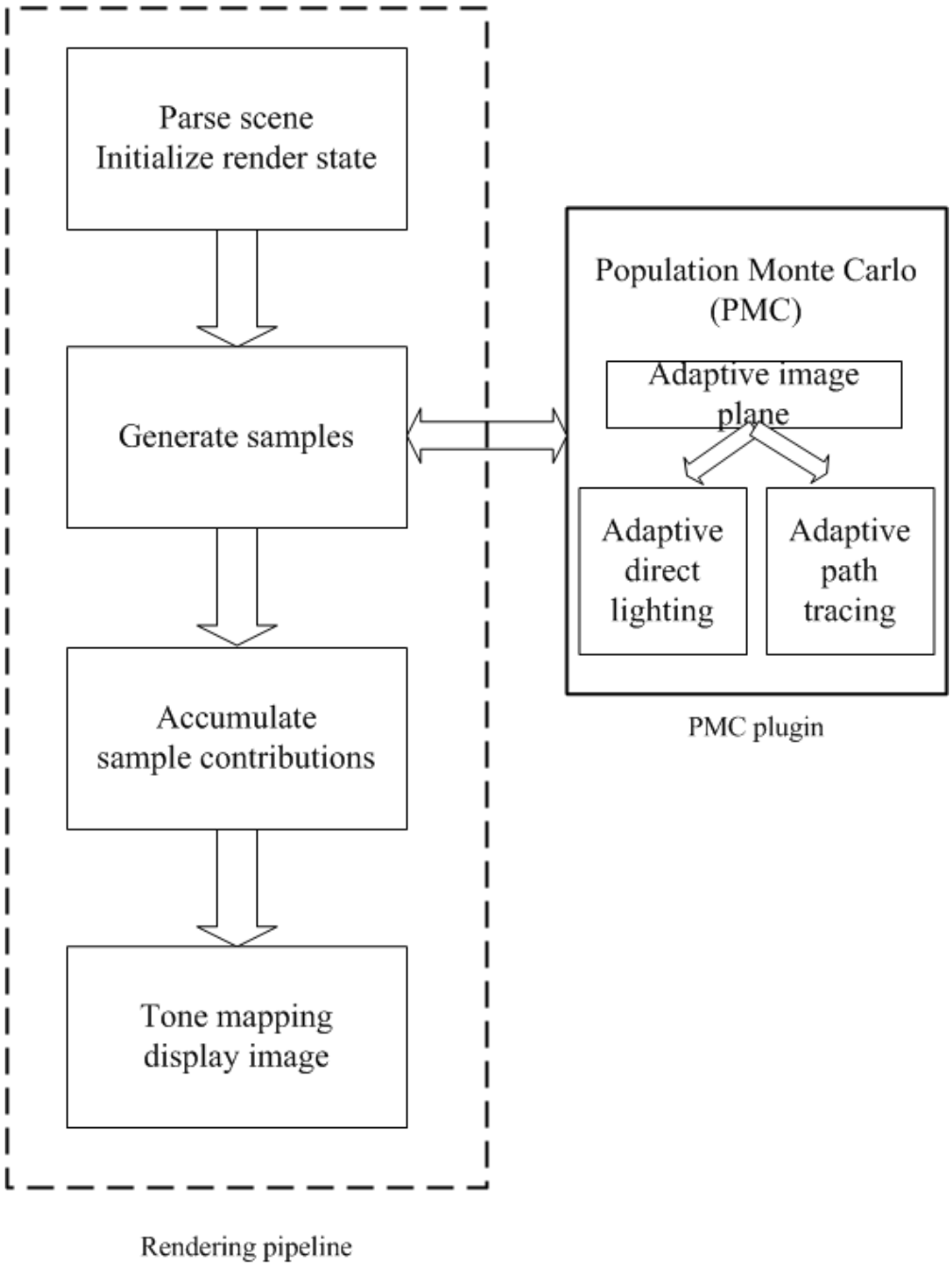

Figure 5.12: Population Monte Carlo rendering in physically based rendering pipeline.

Chapter 6

Optimizing Control Variate Estimators for Rendering

Monte Carlo integration methods offer the most general solution to physically accurate lighting simulation: they handle near-arbitrary geometry, material properties, participatory media, etc. All Monte Carlo methods require an *estimator* that takes the information found in the samples and determines a single final value. A good estimator is unbiased and has low variance. In rendering, the unbiased property guarantees the image has, on average, the correct pixel values, while variance determines the noise levels in the image, or how much neighboring pixels tend to differ in value.

There are many possible estimators, each of which combines the samples in a different way to get the final answer. If we focus on unbiased estimators, then a good strategy is to choose one that minimizes variance while remaining relatively fast to compute. The most common estimator in rendering is the sample mean or an importance weighted mean. Alternatives exist, however, such as the Multiple Importance Sampling (MIS) estimator [102] or control variate estimators [90] (also referred to as correlated sampling).

In this chapter we apply an Optimizing Control Variate (OCV) estimator to the problem of estimating irradiance integrals for direct lighting. The same basic problem is also a sub-component of many rendering algorithms, such as irradiance caching and photon-map gathering, for which we also demonstrate some results. The OCV estimator solves a small optimization problem to find a good control variate distribution given a set of samples. Unlike existing control variate methods which require a single control variate distribution for all estimates, OCV allows the distribution to

vary over the scene depending on surface properties and lighting conditions. Furthermore, users are not burdened with finding an optimal correlated function; they can provide a generic parameterized function that the estimator optimizes.

OCV works with the *deterministic mixture sampling* (DMS) framework for constructing importance functions, sampling from them, and computing estimates from the samples [69]. In addition to providing better estimators, DMS allows for multiple importance sampling functions to be combined in a general way. The optimizing nature of the estimator ensures that the combination of samplers performs at least as well as the best among them. In this way, OCV can be viewed as a generalization of multiple importance sampling.

6.1 Estimating Irradiance Integrals

This chapter concentrates on the problem of computing integrals over hemispheric domains. The most common such integral in rendering computes the radiance, $L(\mathbf{x}, \omega)$, leaving a point $\mathbf{x}$ in the direction ω:

$$L(\mathbf{x}, \omega) = L_e(\mathbf{x}, \omega) + \int_\Omega f(\mathbf{x}, \omega, \omega') d\omega' \qquad [6.1]$$

where $L_e(\mathbf{x}, \omega)$ is light emitted at $\mathbf{x}$, Ω is the hemisphere of directions *out* of $\mathbf{x}$ and $f(\mathbf{x}, \omega, \omega')$ is the light reflected at $\mathbf{x}$ from direction $-\omega'$ into direction ω:

$$f(\mathbf{x}, \omega, \omega') = L_{in}(\mathbf{x}, -\omega') f_r(\mathbf{x}, \omega, \omega') |\cos(\theta')|$$

$L(\mathbf{x}, -\omega')$ is light arriving at $\mathbf{x}$ from direction ω', $f_r(\mathbf{x}, \omega, \omega')$ is the BRDF, and θ' is the angle between ω' and the normal at $\mathbf{x}$. Monte Carlo renderers use statistical sampling to estimate the integral for the reflected component of $L(\mathbf{x}, \omega)$.

A standard importance sampling algorithm for $L(\mathbf{x}, \omega)$ samples directions, $\omega'_1, \ldots, \omega'_N$, out of $\mathbf{x}$ according to an importance distribution, p, and computes the estimate:

$$\hat{L}(\mathbf{x}, \omega) = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}, \omega, \omega'_i)}{p(\omega'_i)} \qquad [6.2]$$

The variance of this estimator improves as p more closely approximates f, and is zero when p differs from f by a constant scale.

In local direct lighting situations, a common choice for p is a normalized version of $f_r(\mathbf{x}', \omega, \omega')|\cos(\theta')|$ or an approximation to it. We refer to this as BRDF-based importance sampling. An alternative is light-based sampling where the integral is broken into a sum over individual light sources and points are sampled on the lights to generate directions [73, §16.1]. In environment map lighting situations, the wavelet product approach of Clarberg et al. [12] currently provides the best way to choose p.

Control variate approaches (see Section 2.4.2) introduce a correlated function, g, which should have the property that $f - g$ is close to a constant, and then use the estimator:

$$\hat{L}(\mathbf{x}, \omega) = \int_\Omega g(\omega')d\omega' + \frac{1}{N}\sum_{i=1}^{N}\frac{(f(\mathbf{x}, \omega, \omega'_i) - g(\omega'_i))}{p(\omega'_i)} \quad [6.3]$$

The difficulty of applying this approach in rendering problems is in finding a function g that is sufficiently close to f in all places. We solve this problem by defining a parameterized function, $g(\omega' : \beta_1, \ldots, \beta_m)$, and optimizing the vector of parameters, $\langle\beta_1, \ldots, \beta_m\rangle$, in order to best approximate f.

The MIS estimator [102] uses multiple importance functions, $p_1, \ldots, p_m$, and draws a fixed number of samples from each, $n_1, \ldots, n_m$. It then computes one of several possible estimators, of which the simplest is the *balance heuristic*:

$$\hat{L}(\mathbf{x}, \omega) = \frac{1}{N}\sum_{j=1}^{m}\sum_{i=1}^{n_j}\frac{f(\mathbf{x}, \omega, \omega'_{i,j})}{\sum_{k=1}^{m} c_k p_k(\omega'_{i,j})} \quad [6.4]$$

where $c_k = n_j/N$, the proportion of samples drawn from p_j. The major advantage of MIS is that it enables importance functions to be combined in an unbiased manner. Using a slightly different estimator, the *power heuristic*, the weight of samples coming from poor importance functions can be implicitly reduced in the final estimate.

6.2 Related Work

The simplest effective use of control variates is in cases where the incoming illumination can be approximated by a constant ambient term – Lafortune and Willems [55] described this technique – but it offers less improvement with more complex illumination. Szirmay-Kalos et al. [91] improve

upon this using radiosity to obtain an estimate of the diffuse illumination which serves as the correlated function in a Monte Carlo step that accounts for other illumination. It works well for diffuse environments but not for specular surfaces.

Szécsi et al. [90] combined control variate and importance sampling estimators (Equations 6.2 and 6.3) in a linear combination with weights optimized to reduce variance, but the approach is very limited in the BRDFs that can be handled. Note that this approach combines estimates, not sampling strategies, so a single importance sampling function must still be chosen. An alternate estimator, weighted importance sampling, has been used for particle tracing algorithms by Balázs et al. [5], but a scene discretization is required and improvement is only seen under specific BRDF and lighting configurations.

The work of Lafortune and Willems [56] on adaptive BRDF sampling includes a control variate component. They built a 5D-tree approximation to radiance in the scene, and used it for both importance sampling and control variate estimation. In some sense this is optimizing the control variate estimator. However, large sample counts are required to adequately adapt the necessary functions, and failure to adapt correctly actually increases variance. Our algorithm uses a low-parameter function for the control variate distribution, so few samples are required to optimize.

OCV with deterministic mixture sampling offers a way to combine samples from multiple importance functions. As discussed above, Veach's [102] MIS is an existing approach to this problem. DMS includes the balance heuristic (Equation 6.4) as a special case. We improve upon MIS with a simple optimization process for selecting a better estimator at each pixel.

6.3 Deterministic Mixture Sampling

The optimizing control variate estimator begins with a deterministic mixture sampling process to generate the samples. This is practically equivalent to MIS's step of generating a fixed number of samples from each of multiple importance functions, but motivated differently.

A mixture PDF is one composed of a weighted sum of component PDFs:

$$p(x:\alpha) = \sum_{j=1}^{m} \alpha_j p_j(x) \qquad [6.5]$$

where m is the number of components and α is a vector of *mixture weights*, $\langle \alpha_1, \ldots, \alpha_m \rangle$, with $\alpha_j > 0$ and $\sum_{j=1}^{m} \alpha_j = 1$. The simplest way to draw a sample from a mixture density is to first select a component, j, with probability $p(j) \propto \alpha_j$, and then sample from $p_j(x)$.

For rendering, the mixture can include any importance function that is typically used alone. Hence, we include a component for sampling according to the BRDF and one for each light source. In environment lighting conditions, a component for sampling the environment map should be included. We could break the BRDF into sub-components (diffuse, glossy, etc.) but we did not experiment with this. Also note that the environment map sampling of Clarberg et al. [12] can be viewed as a mixture where each wavelet basis function is a component.

Deterministic mixture sampling chooses a fixed number of samples from each component: $n_j = N\alpha_j$ samples are drawn from component $p_j(x)$ where N is the total sample size. We can view this as a form of stratification over the mixture components, and Hesterberg [42] showed that this reduces variance. Note that this is exactly what MIS does, and Equation 6.4 can be re-written in terms of $p(\omega' : \alpha)$:

$$\hat{L}(\mathbf{x}, \omega) = \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}, \omega, \omega_i')}{p(\omega_i' : \alpha)} \qquad [6.6]$$

We can also construct a control variate estimate using a mixture of functions as the correlated distribution in addition to the importance distribution [69]:

$$\hat{L}(\mathbf{x}, \omega) = \sum_{j=1}^{m} \beta_j + \frac{1}{N} \sum_{i=1}^{N} \frac{f(\mathbf{x}, \omega, \omega_i') - p(\omega_i' : \beta)}{p(\omega_i' : \alpha)} \qquad [6.7]$$

where the β_j are a vector of real valued variables. This estimator is unbiased, as can be seen by writing

$$\begin{aligned} E[\hat{L}_{\alpha,\beta}] &= \int \frac{f(x) - \sum_{j=1}^{m} \beta_j p_j(x)}{p(x : \alpha)} p(x : \alpha) dx + \sum_{j=1}^{m} \beta_j \\ &= \int f(x) dx - \sum_{j=1}^{m} \beta_j \int p_j(x) dx + \sum_{j=1}^{m} \beta_j \\ &= \int f(x) dx \end{aligned}$$

Note that $p_j(x)$ is a PDF so integrates to 1. The variance of the estimator in Equation 6.7 is

$$\sigma^2_{\alpha,\beta} = \int \left(\frac{f(x) - \sum_{j=1}^{m} \beta_j p_j(x)}{p(x : \alpha)} - I + \sum_{j=1}^{m} \beta_j \right)^2 p(x : \alpha) dx \qquad [6.8]$$

where I is the true value of the integral being estimated.

There is no improvement over importance sampling if we set $\beta_j = \alpha_j$ for all j; it is the same estimator as Equation 6.6. However, we are free to choose the β_j in a variety of ways – they need not even sum to 1. In particular, we can solve an optimization problem, which results in an OCV estimator.

6.4 Optimizing Control Variates

A natural strategy for choosing the β_j is to minimize the variance in Equation 6.8. We can't do this, however, because we don't know I, the value we are trying to estimate. Instead, we form a linear problem that minimizes the following objective function with respect to the β_j:

$$\sum_{i=1}^{N} \left(\frac{f(X_i) - \sum_{j=1}^{m} \beta_j p_j(X_i)}{p(X_i : \alpha)} \right)^2 \qquad [6.9]$$

This is a standard linear least squares problem, but we modify it in three ways. First, we include an intercept term, β_0 [69], which after optimization evaluates to

$$\frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i) - \sum_{j=1}^{m} \beta_j p_j(X_i)}{p(X_i : \alpha)}$$

Putting β_0 into Equation 6.7 and simplifying, we get a simpler form of the OCV estimator:

$$\hat{L}(\mathbf{x}, \omega) = \beta_0 + \sum_{j=1}^{m} \beta_j \qquad [6.10]$$

The second problem is that the condition $\sum_{j=1}^{m} \alpha_j = 1$ required to make $p(x : \alpha)$ a distribution function means that the $p_j(x)/p(x : \alpha)$ terms are linearly dependent. This can be solved by dropping p_m from the optimization and setting $\beta_m = 0$. This leaves us minimizing $\|\mathbf{y} - \mathrm{A}\beta\|^2$ with

$$\mathbf{y} = \begin{bmatrix} \frac{f(X_1)}{p(X_1:\alpha)} \\ \vdots \\ \frac{f(X_N)}{p(X_N:\alpha)} \end{bmatrix}$$

$$\mathrm{A}\beta = \begin{bmatrix} 1 & \frac{p_1(X_1)}{p(X_1:\alpha)} & \cdots & \frac{p_{m-1}(X_1)}{p(X_1:\alpha)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \frac{p_1(X_N)}{p(X_N:\alpha)} & \cdots & \frac{p_{m-1}(X_N)}{p(X_N:\alpha)} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{m-1} \end{bmatrix}$$

A further problem occurs when all of the samples from some component are zero. In rendering, this is quite likely due to occlusion or some other factor that gives zero radiance from some directions. To deal with this we use *penalized least squares* with a penalty term pushing the β_i toward zero. The resulting objective function is $\|\mathbf{y} - \mathrm{A}\beta\|^2 + \lambda\|\beta\|^2$. The solution to this problem is

$$\hat{\beta} = (\mathrm{A}'\mathrm{A} + \lambda\mathrm{I})^{-1}\,\mathrm{A}'\mathbf{y} \qquad [6.11]$$

where A' is the transpose of A and I is the identity matrix. We found $\lambda = 1$ to be good in practice.

6.4.1 OCV for Rendering

Optimizing control variate estimation is useful in rendering when evaluating integrals over a single domain, with the same PDF used for each sample, and a choice of importance functions. While Veach [102] showed a bidirectional path tracing application, in practice the conditions are met in *gather* integrals where we integrate incoming irradiance at a point by sampling over the hemisphere. Such integrals arise in direct lighting, irradiance caching, photon-mapping, and radiosity. We show examples from the first two applications.

Apart from choosing components for the mixture, we must also set their weights, α_i. In all our experiments we used a single BRDF-based component and one component for each light (we did not use environmental lighting). We made a conservative choice: half of the samples came from the BRDF, $\alpha_{BRDF} = 0.5$, while the remainder were divided equally among the lights. If for some reason a user thought some sampling function was more likely to succeed, then the weight for that component could be increased. It is possible to set the weights adaptively [80], and we are aware of a paper that experiments with this [28], but does not use an OCV estimator.

To summarize, each time we require an estimate of the integral in Equation 6.1, we draw a fixed number of direction samples, n_j, from each importance function in the mixture, p_j. We

trace rays for each sample to determine the incoming radiance, $L_{in}(\mathbf{x}, -\omega_i')$. With each sample direction evaluated, we form the matrices and vectors and solve Equation 6.11 for the β_j. Finally, Equation 6.10 is evaluated to compute the estimate of outgoing radiance.

In direct lighting, an irradiance integral estimate is obtained for every surface point hit with a pixel sample. For irradiance caching, another application we have implemented, the incoming irradiance must be estimated at diffuse surface points when a nearby cached estimate is not available. The irradiance integral is broken into two terms:

$$Ir(\mathbf{x}) = \int_\Omega L_{sources}(\mathbf{x}, -\omega')d\omega' + \int_\Omega L_{ind}(\mathbf{x}, -\omega')d\omega'$$

where $Ir(\mathbf{x})$ is the irradiance at point $\mathbf{x}$, $L_{sources}$ is incoming radiance due to light or environmental sources, and L_{ind} is radiance due to indirect lighting. In our implementation [73], $L_{ind}(\mathbf{x}, -\omega')$ is computed using path tracing, but each point along the path also evaluates the direct lighting integral.

We only use OCV for the irradiance due to sources. All indirect lighting estimates happen at diffuse surfaces, and samples that directly hit a light contribute nothing because they are already accounted for. Hence, BRDF-based importance sampling is the only importance function suitable, and therefore OCV is not useful – there is no way to form a mixture. Note, however, that the irradiance integral is evaluated as part of the path tracing procedure, so OCV does still contribute to indirect lighting.

6.5 Results

We first experimented with a scene (Figure 6.2) that demonstrates the importance of including multiple sampling functions for direct lighting (following [73]). This example contains two lights, so half of all the samples come from sampling a BRDF-based component, while one quarter come from sampling the area of the yellow light and a quarter from the blue light. Table 6.1 presents timing and error results, where error is a perceptually weighted error metric:

$$E = \left[\frac{1}{n}\sum_{pixels}\left(\frac{L - L_{true}}{tvi(L_{true})}\right)^2\right]^{\frac{1}{2}} \qquad [6.12]$$

Image	Method	SPE	SPP	Time (s)	Err
Checks	MIS	64	4	172.8	0.60
	OCV	64	4	180.8	0.48
Buddha	MIS	64	4	98.3	0.72
	OCV	64	4	105.6	0.46
Room	MIS	18	2	37.4	0.75
	OCV	18	2	43.2	0.68
Box	MIS	18	9	196.5	4.9
	OCV	18	9	207.2	4.0

Table 6.1: Measurements comparing MIS to OCV for direct lighting computations. SPE is the sample count per estimate, with SPP estimates per pixel. Err is the error computed using Equation 6.12.

where n is number of pixels, L is the luminance of the result, L_{true} is the true luminance, and $tvi(x)$ is the perceptual threshold-vs-intensity function introduced by Ferwerda et al. [30]. We use perceptual weighting to avoid giving too much weight to very bright or very dark areas of the image. The ground truth image is computed using MIS running for several hours.

Figure 6.2 shows comparison between MIS, OCV and the correlated sampling approach of Szécsi et al. [90]. These images were rendered at 500×500 resolution. They highlight primarily the value in using multiple importance functions, which correlated sampling cannot do. OCV performs better than MIS on this scene with little additional computation time. Improvement in the form of lower variance is most apparent in the glossy region reflected in the yellow light. In this scene the OCV estimator results in a 18% improvement in image quality with about 5% more computation time.

The Buddha images (Figure 6.1) show a more marked improvement with OCV over MIS. These images were rendered at 256×512 resolution, and the OCV estimator results in a 37% improvement for 7% more time. This scene has a greater variety of lighting conditions, ranging from tight specularities to occluded regions. Our final direct lighting test used a Room scene (Figure 6.3),

for which the OCV estimator produced lower error compared to MIS, but the additional computation cost resulted in comparable rendering efficiency. The scene requires relatively few samples to obtain a good estimate because the light sources are small and there is limited occlusion. Our method performs best when occlusion is complex and with larger light sources. Still, due to the optimization in OCV, the results are unlikely to be worse than alternative methods.

The Cornell Box scene (Figure 6.4) demonstrates OCV estimates in irradiance caching. The perceptual RMS error (Equation 6.12) for the standard implementation is 4.9, which OCV reduces to 4.0 with about 5% more computational time.

We compare the algorithms based on the same number of samples instead of the same computational time, because presetting number of samples makes the implementation more efficient by taking advantage of the stratified sampling. Given the efficiency is measured by the perceptual RMS error and its computational time [73], their product provides a fair comparison of the algorithms running with the same number of samples.

6.5.1 OCV in the rendering pipeline and its limitations

We do not use OCV for the indirect lighting component of the irradiance caching integral because our techniques for forming a mixture result in a single component. We could form a mixture by sub-dividing the hemisphere and using one component for each sub-region. This would allow things such as occluded paths to be accounted for in the estimator.

As stated above, an OCV estimator is only useful in situations when all the samples come from the same mixture distribution. In bidirectional path tracing, this means we can only use it on a per-path basis with a mixture component for each method of forming the path. Path tracing is ruled out because each path has a different length and hits a different set of material properties, and hence has a different PDF. Integrals of the form in Equation 6.1 are very common, however, so OCV does cover a large set of practical cases. Figure 6.5 shows how OCV can be used to construct an estimator for accumulating sample contributions in the traditional rendering pipeline.

The primary limitation with the OCV estimator comes from the relationship between the number of components in the mixture and the number of samples required. A larger mixture requires

more samples to obtain reliable values for optimized β – at least as many samples as components. Furthermore, more mixture components and samples increases the cost of the optimization, to the extent that MIS would perform better for the same computation time. Hence, very small sample counts (less than about 10) cannot be used and situations with many light sources cause problems, at least as we have constructed the mixture. In a many-light situation, nearby lights could be grouped into one component or an environmental lighting approach could be used.

6.6 Conclusion

We have presented a new estimator for use in computing irradiance gather integrals. The OCV estimator maximizes the benefits of control variate sampling by optimizing the correlated function at each estimate. This also reduces the user's burden of finding correlated functions. In addition, OCV allows multiple importance functions to be combined, which is particularly useful when no one function works well across an entire image.

In importance sampling applications, one use of mixtures is in *defensive* sampling [42], where one component of the mixture is certain to have "heavier tails" than the integrand to ensure finite variance of the estimate. In rendering, situations where a defensive component is useful are rare: one example is a glossy surface under environmental lighting where the dominant reflectance lobe is blocked by an occluder, and wavelet product sampling is in use. A cosine-weighted mixture component could be used as a defensive choice in such situations.

There are several alternative importance functions that could be used as components. One particularly interesting possibility is using the low-frequency wavelets from Clarberg et al. [12]. The potential advantage is that wavelets representing occluded directions could have their weight in the estimate reduced. Even more advantage could come from an approach that adapts the mixture weights, and hence avoids any sampling in occluded directions.

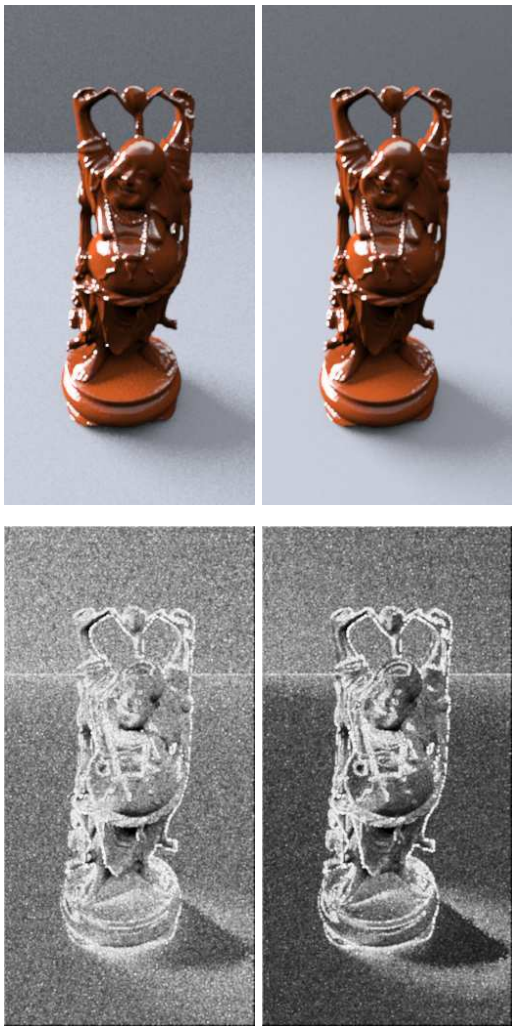

Figure 6.1: Results for MIS and OCV for the Buddha model. MIS, left, has noticeably higher variance in the soft shadow boundary and the base of the Buddha. The variance images, below, reveal significant reduction in variance with OCV over the entire image.

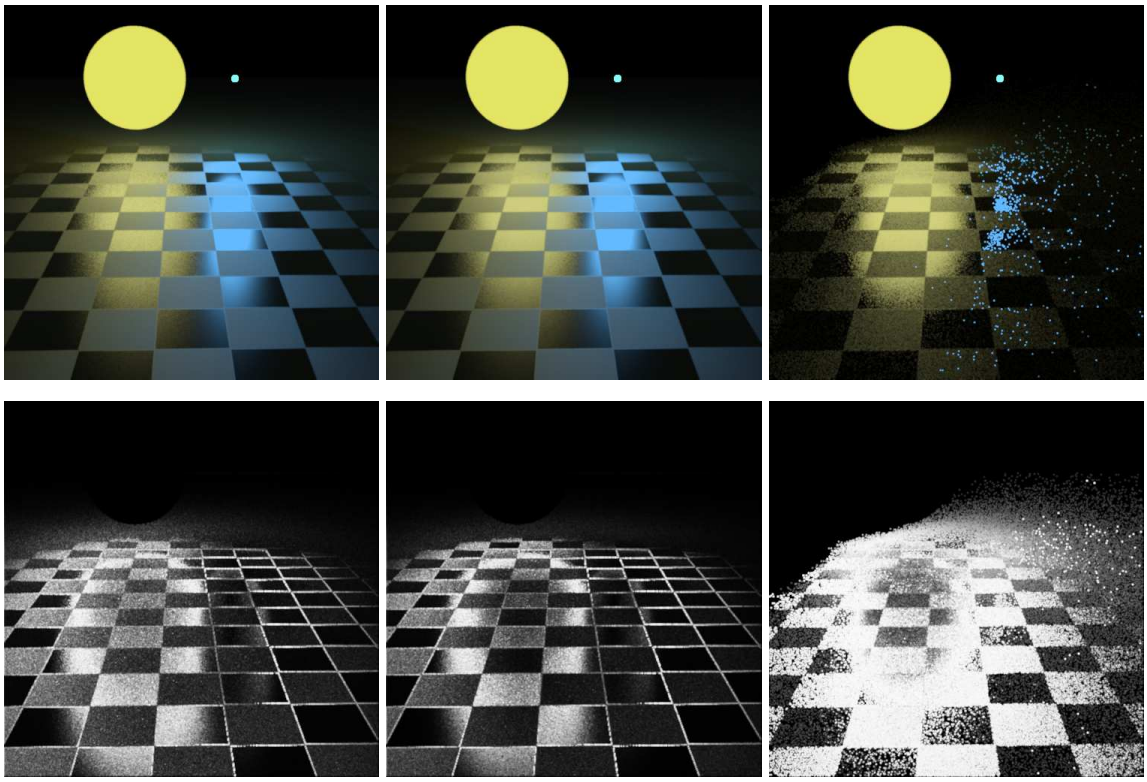

Figure 6.2: Images for the checkers scene. Left is MIS, center is OCV and right is correlated sampling. Correlated sampling performs poorly because it must choose only one importance function before rendering begins (typically BRDF-based, as we have here) and the best choice is not always obvious. Bottom are perceptually-based variance images, which show the variance of the direct illumination estimates obtained at each pixel. The most significant improvement of OCV over MIS is apparent within the left glossy reflection of the large light source. Note that variance is expected to be large at material property boundaries because different pixel samples are hitting different materials.

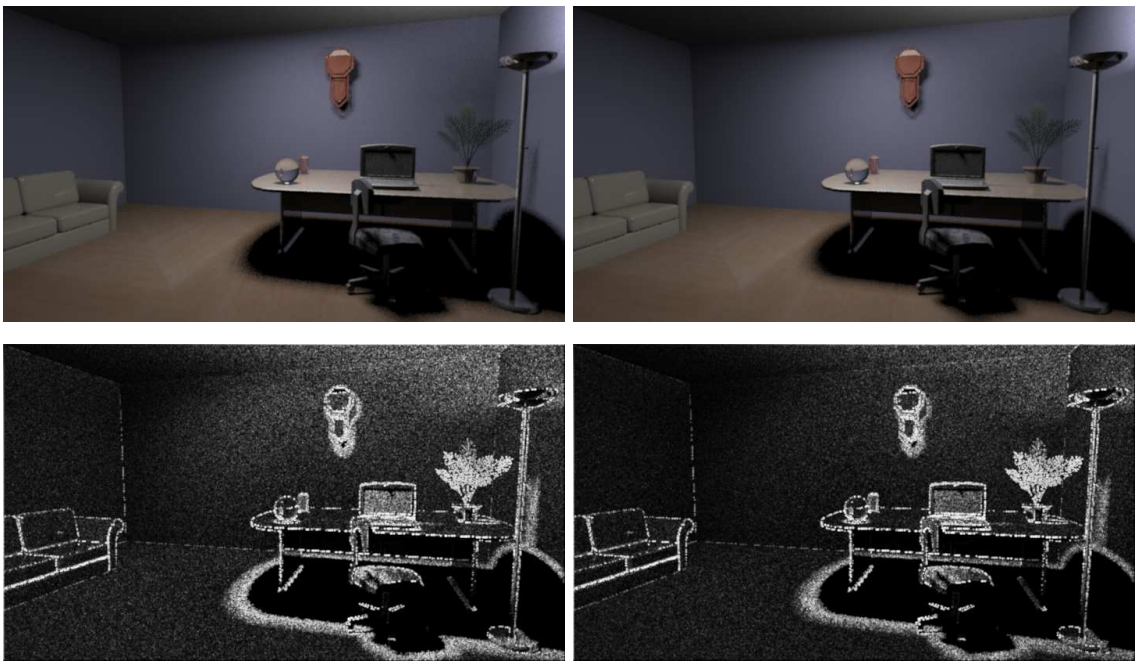

Figure 6.3: Results for MIS (left) and OCV (right) for the Room scene. The images are very similar, but the variance images below reveal an overall improvement with OCV over MIS.

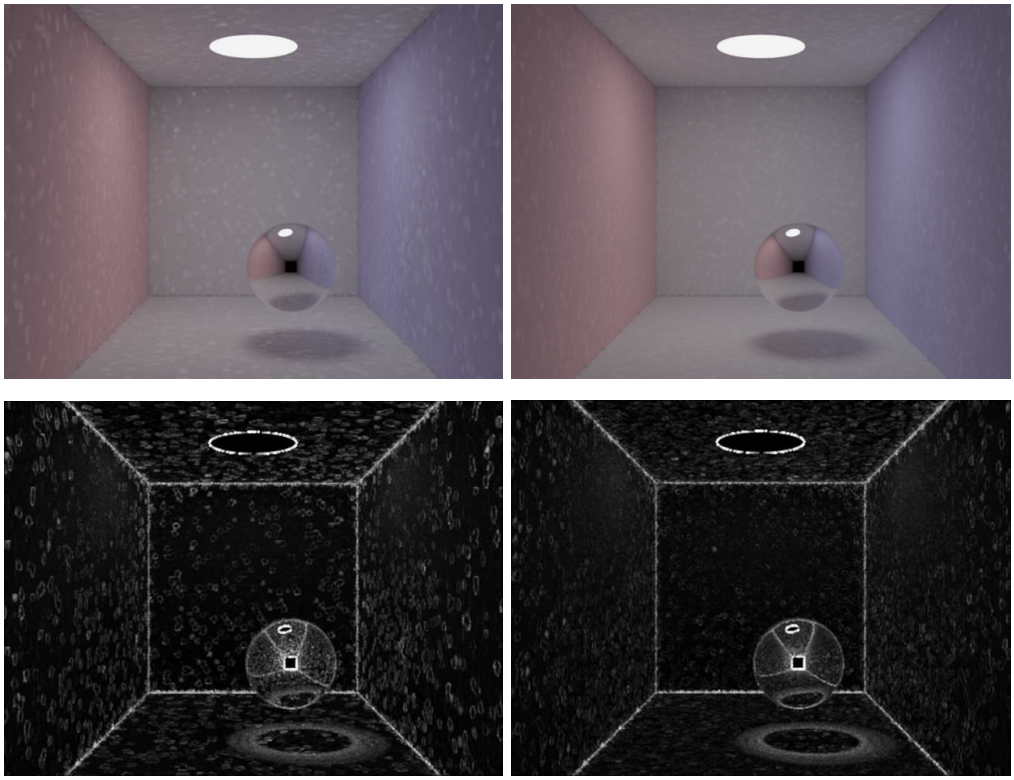

Figure 6.4: Results for MIS and OCV for irradiance caching computations on a Box scene. Standard irradiance caching, which uses MIS for its estimates, is on the left, while a version using OCV estimators is on the right.

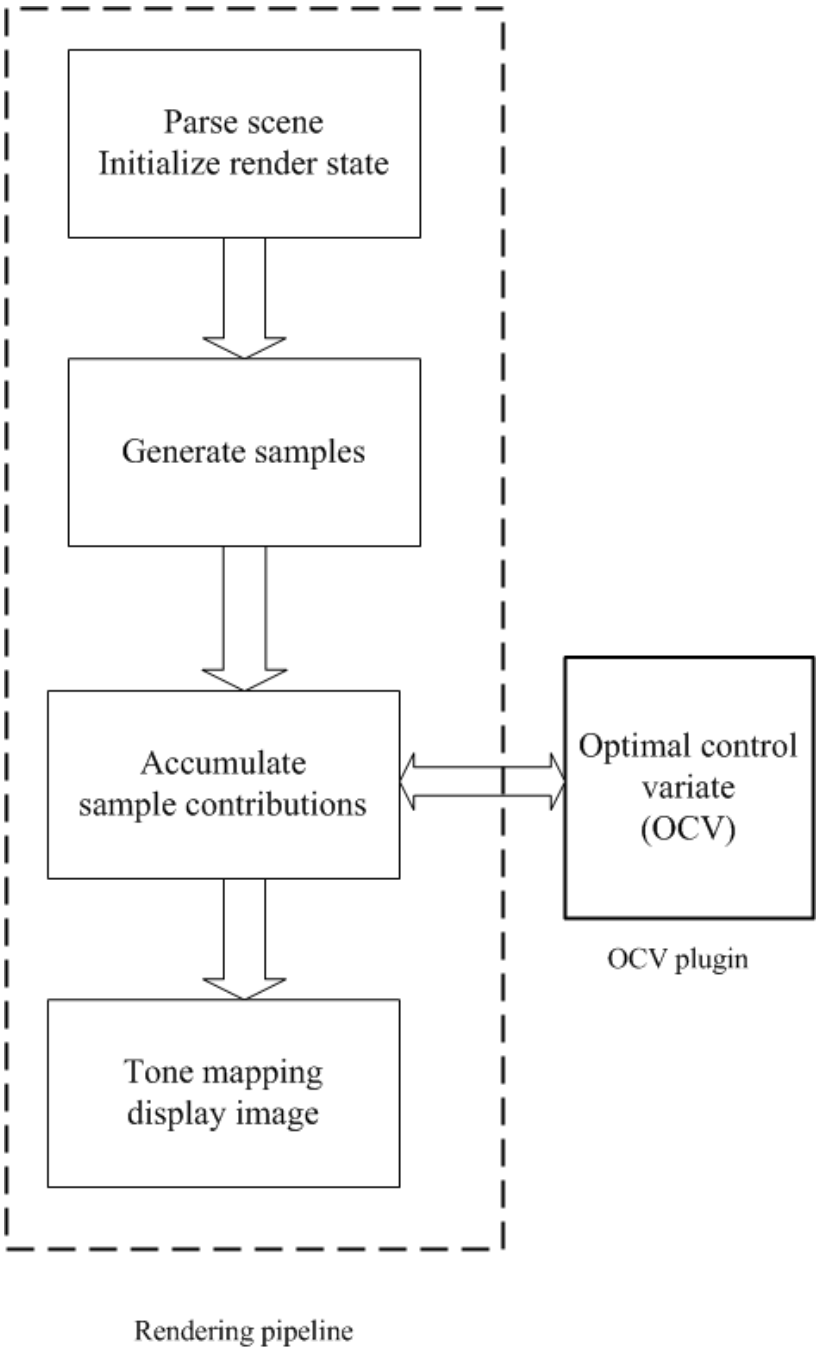

Figure 6.5: OCV in the physically based rendering pipeline. It can be used to construct a better estimator based on the samples from a mixture of multiple distributions.

Chapter 7

Discussion and Conclusion

The Monte Carlo method is the most general and robust method for solving the global illumination problem. The major challenges in Monte Carlo rendering are to sample the path space efficiently and to construct good estimators to reduce the variance in the rendered images. One promising avenue to face those challenges is to adaptively sample the important regions of the integrand and reuse the high-contribution path samples as much as possible. This thesis has introduced a novel statistical framework for adapting and reusing samples and demonstrated its applications to the global illumination problem. This chapter presents a summary of the main contributions of this work and a discussion of future work.

7.1 Contributions

The idea of adaptive sampling has been explored for image rendering by many researchers (e.g., [70, 6]). The major disadvantage of adaptive sampling is that it can introduce bias if not used with special care, as pointed out by Kirk and Arvo [51]. Most of the adaptive algorithms in the literature are biased and there is no analysis of how the bias affects the final rendering results. Two-stage sampling proposed by Kirk and Arvo [51] eliminates the bias, however, it also wastes samples from the first stage and cannot adjust sampling during the second stage.

Sequential Monte Carlo puts sample adaptation and sample reuse into a new statistical framework that enables repeated updates to importance distributions based on the performance of the sampling process. As a specific type of SMC method, the population Monte Carlo algorithm

makes it much easier to construct adaptive sampling schemes without introducing bias. Allowing samples to be dependent on previous samples provides a straightforward way for sample reuse. Resampling according to the importance weights of the samples not only keeps high-contribution samples for reuse, but also prunes low-contribution samples from the population. Working in an importance sampling framework, PMC removes the ergodicity issue of the MCMC framework.

Applications of this framework are demonstrated with a variety of problems in physically based rendering. For the task of photo-realistic rendering, only light paths that reach the image plane are important because only those paths contribute to the final image. As a way of generating and reusing important path samples, we proposed a visual importance-driven algorithm, *Metropolis Photon Sampling* (MPS), for populating photon maps in the photon mapping context [27]. Our sampling strategy is independent of photon mapping and can be used either alone to generate visually important paths, or with photon maps that are used to further exploit sample reuse. Metropolis Photon Sampling succeeds in generating photon map samples that meet the needs of the final gather phase without wasting storage or computation time on unnecessary photons. It achieves this by sampling only over light transport paths that reach the image, and storing photons only at appropriate points along the path. The photon distribution that results has more photons that contribute to visually important locations, and fewer in irrelevant places. This not only improves estimates from the map due to higher photon density, but also reduces the chance that inappropriate photons will be used and hence reduces energy bleeding artifacts. At the same time, MPS allows users to supply information to the sampler in the form of important paths or difficult paths, something not achievable in most Monte Carlo algorithms.

To further demonstrate the utility of the sequential Monte Carlo framework for physically based rendering problems, the population Monte Carlo rendering algorithm was proposed and applied to a number of problems in realistic rendering [28]. Those applications are adaptive image-plane sampling (PMC-IP), hemispheric integral sampling (PMC-HI), and energy-redistribution-based path tracing (PMC-PT). The adaptive image-plane sampler selects pixels for refinement according to a perceptually-weighted variance criterion and has no statistical bias. The adaptive hemispheric

integrals sampler learns an importance sampling function for computing common rendering integrals. Adaptive energy redistribution path tracing concentrates computation in regions of high variance or important light transport paths. Each algorithm is derived in the generic population Monte Carlo statistical framework.

The image-plane sampler and direct lighting integrator are common components in many rendering algorithms. PMC-IP sampling could be used as a plugin component for essentially any algorithm that forms light paths to the eye, including the gather phase of photon mapping, bidirectional path tracing, irradiance caching, and so on. The PMC-HI sampler could be used in any situation where estimates of an integral over the hemisphere are required. Irradiance caching can benefit greatly from a PMC sampler in the computation of each cached value. Photon mapping can also use a PMC sampler in the final gather, but we expect the improvement to be less apparent because the final gathering tends to smooth the result.

To address the problem of optimally constructing estimators that combine samples from several different PDFs, we presented the Optimizing Control Variate estimator, a new estimator for rendering that uses both importance sampling and the control variate method [26]. This is an important issue because in the population Monte Carlo rendering framework, samples are generated from a sequence of distributions and how the estimator combines them has a big impact on the image variance. Based upon a deterministic sampling framework, OCV allows multiple importance sampling functions to be combined in a general way, which can be viewed as a generalization of the multiple importance sampling method. The optimizing nature of OCV addresses a major problem with control variate estimators for rendering: users supply a generic, correlated function that is optimized for each estimate, rather than a single highly-tuned one that must work well everywhere. The same basic problem is also a sub-component of many rendering algorithms, such as irradiance caching and photon-map gathering.

7.2 System and Limitations

The algorithms presented in this dissertation can be easily combined as a whole or used as separate components to speed up the rendering process in a commercial rendering system. Figure 7.1 shows how each algorithm can be used in the global illumination rendering pipeline.

The algorithms proposed in this thesis all deal with the problems of how to generate and reuse samples, and how to build efficient estimators. Each algorithm can be selected independently and plugged into the pipeline. The adaptive image plane method can be used for almost any Monte Carlo ray tracing algorithm since all the contributing samples eventually penetrate through the image plane. It is practical and simple to implement. With a little overhead for generating pixel samples based on a mixture distribution, the adaptive image plane strategy promises great efficiency gain for images that have regions with highly varied rendering complexities; for example, a scene mixed with both high noise regions such as soft shadows and caustics that requires many samples, and low noise regions such as plain direct lighting on a non-textured plane. Similarly, the optimal control variate algorithm can be used in the rendering pipeline: an algorithm with samples generated from multiple sampling distributions can be efficiently combined with the OCV algorithm. The adaptive hemispherical integral algorithm can be used in situations when multiple samples are needed for estimating the irradiance in a hemisphere. Those cases include direct lighting and the final gathering phase in photon mapping. Currently all the sampling distributions for adaptive sampling and optimal control variate are deterministic mixtures. Usually using a relatively small number of sampling distributions is preferable because otherwise the samples required to adequately determine the mixture component weights will be too large.

7.3 Future Research

PMC is just one approach from the family of sequential Monte Carlo methods [80]. Common to these techniques is the idea of sample re-use through resampling and the adaptation of sampling parameters over iterations. There are many problems in computer graphics where integrals of a specific form must be evaluated or a sampling process has to be employed, so it certainly offers

many further opportunities to exploit the properties of sequential Monte Carlo methods. Since solving the rendering equation is essentially approximating an integral, all photo-realistic rendering problems may potentially benefit from the sequential Monte Carlo framework. For example, algorithms for interactive rendering and animation rendering can be improved in efficiency if samples can be reused and adapted. In the context of rendering, the following questions are still open:

- **User guidance for interactive global illumination rendering**

  Interactive global illumination rendering is very important for applications such as lighting design and architectural modelling. Speed is critical for interactive rendering, yet computing a global illumination solution is very expensive. These two conflicting requirements suggest algorithms to trade off rendering accuracy with speed (see [98, 35] for reviews). However, in interactive global illumination rendering, some specific regions or objects are more important for the viewer than others in the scene. For example, in product design, the photo-realistic appearance of the objects with modified materials is likely what the user really cares about. It will speed up the rendering process if user guidance can be incorporated into the sampling strategy so that more important regions are rendered with higher accuracy. This could also provide a nice tool for users doing interactive material design, for example.

- **Adaptive transitional kernels in path space for PMCR**

  In current applications in population Monte Carlo rendering, the domain for adaptation is either the image plane or a hemisphere for one local direction bounce. It would be useful to adapt the sampling process along the whole path space, i.e., the transitional kernel can change the path in a less restricted way. One way of doing that is to construct transitional kernels such as bidirectional path mutations and lens mutations in MLT, and then adaptively select mutation strategies based on the path performance. This might also provide a way to automatically tune MLT or energy redistribution path tracing algorithms.

- **Optimal estimator for the samples from a sequence of correlated distributions**

OCV provides a provably good way to combine samples from different distributions. Currently it is assumed that those samples are generated independently from un-correlated distributions. In the population Monte Carlo rendering work, the distributions in the sequence are correlated. While weighting those samples based on the sample variances at each iteration provides a valid way of combining samples, more efficient ways may exist and are worth further study.

- **Sequential Monte Carlo for animation rendering**

  In animation rendering, the samples from neighboring frames can provide useful hints for generating good samples for the current frame, i.e., when a sample with high importance is found, it should be exploited temporally. Reusing the path samples to create temporal coherence in the lighting distribution between subsequent animation frames will dramatically reduce temporal flickering effects. Applying SMC methods in the global illumination animation rendering context will offer three major advantages over existing methods: (1) reduce temporal noise by choosing samples correlated across frames without introducing bias; (2) make it easier to find important, rare light paths in one frame by sharing information obtained in neighboring frames; and (3) provide a natural way to discard low contribution samples and retain high contribution samples based on the sample weight.

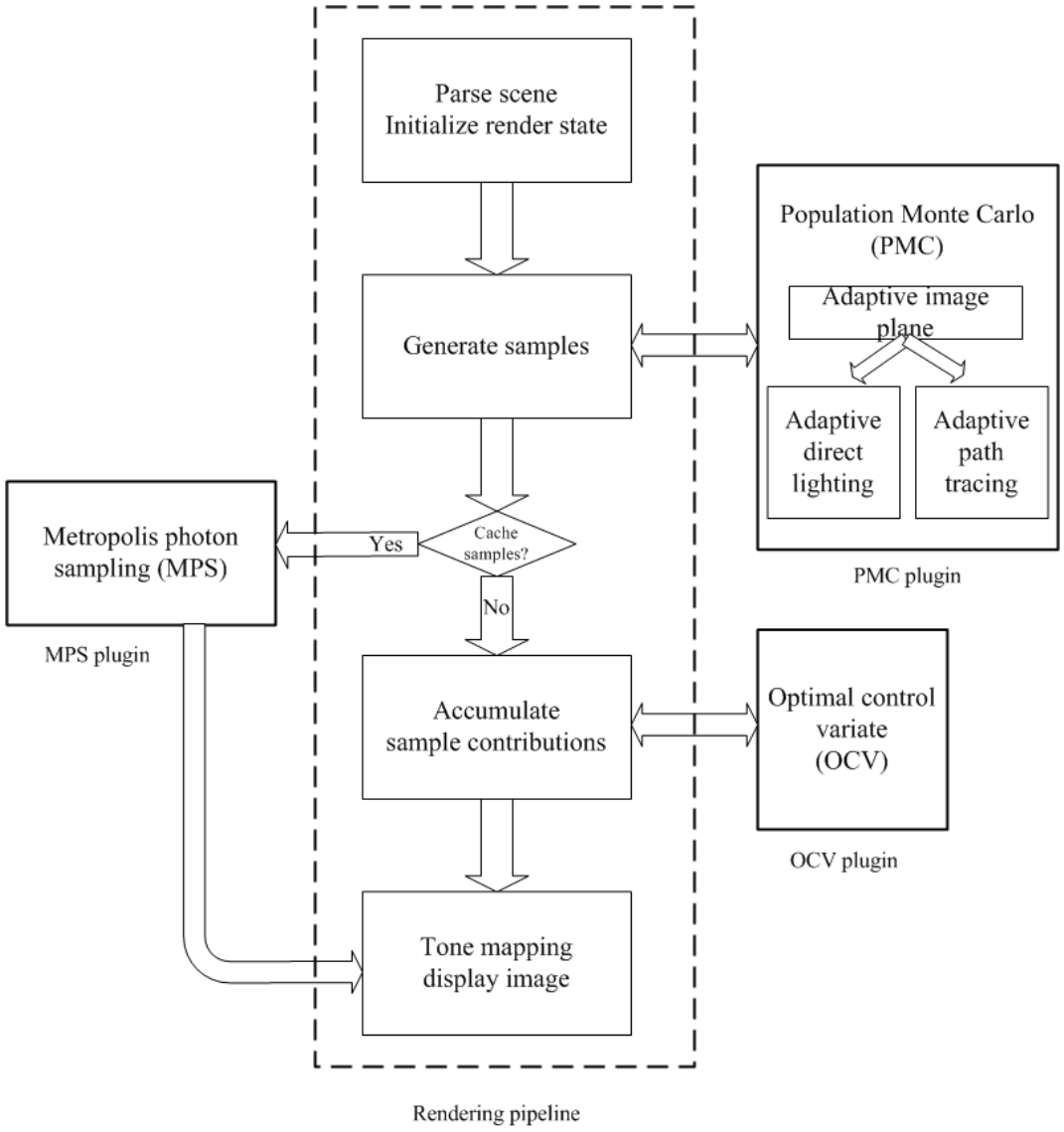

Figure 7.1: Physically based rendering system diagram. Inside the dotted box is the traditional rendering system flowchart. The three bold boxes show the potential plugins presented in this dissertation.

LIST OF REFERENCES

[1] Sameer Agarwal, Ravi Ramamoorthi, Serge Belongie, and Henrik Wann Jensen. Structured importance sampling of environment maps. In *SIGGRAPH '03: Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques*, pages 605–612, 2003.

[2] Christophe Andrieu and Arnaud Doucet. Joint Bayesian model selection and estimation of noisy sinusoids via reversible jump MCMC. *IEEE Transactions on Signal Processing*, 47(10):2667–2676, 1999.

[3] James Arvo. Backward ray tracing. *Developments in Ray Tracing. ACM SIGGRAPH course notes*, 12:259–263, 1986.

[4] James Arvo. Transfer Functions in Global Illumination. In *ACM SIGGRAPH '93 Course Notes - Global Illumination*, pages 1–28. 1993.

[5] Benedek Balázs, László Szirmay-Kalos, and Antal György. Weighted importance sampling in shooting algorithms. In *Proceedings of the Spring Conference on Computer Graphics*, pages 177–184, 2003.

[6] Mark R. Bolin and Gary W. Meyer. A perceptually based adaptive sampling algorithm. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 299–309, 1998.

[7] David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. Bidirectional importance sampling for direct illumination. In *Rendering Techniques'05: Proceedings of the Eurographics Symposium on Rendering*, pages 147–156, 2005.

[8] O. Cappé, A. Guillin, Jean-Michel Marin, and Christian Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13(4):907–929, 2004.

[9] Shenchang Eric Chen, Holly E. Rushmeier, Gavin Miller, and Douglass Turner. A progressive multi-pass method for global illumination. In *SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 165–174, 1991.

[10] Per H. Christensen. Adjoints and importance in rendering: An overview. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):1–12, 2003.

[11] P.H. Christensen. Industrial-strength global illumination. In *SIGGRAPH '03, Course Notes No. 27*, pages 139–149, 2003.

[12] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet importance sampling: efficiently evaluating products of complex functions. In *SIGGRAPH '05: Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 1166–1175, 2005.

[13] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. In *SIGGRAPH '05: Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 1186–1195, 2005.

[14] Steven Collins. Adaptive splatting for specular to diffuse light transport. In *Rendering Techniques '94 (Proceedings of the 5th Eurographics Workshop on Rendering)*, pages 119–135, 1994.

[15] D. Crisan. Particle filters – a theoretical perspective. *Sequential Monte Carlo in Practice*, pages 17–38, 2001.

[16] Abhinav Dayal, Cliff Woolley, Benjamin Watson, and David Luebke. Adaptive frameless rendering. In *Rendering Techniques'05: Proceedings of the Eurographics Symposium on Rendering*, pages 265–275, 2005.

[17] Mark A. Z. Dippé and Erling Henry Wold. Antialiasing through stochastic sampling. In *SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 69–78, 1985.

[18] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive global illumination using selective photon tracing. In *Rendering Techniques '02 (Proceedings of the 13th Eurographics Workshop on Rendering)*, pages 25–36, 2002.

[19] R. Douc, A. Guillin, J. M. Marin, and C. P. Robert. Convergence of adaptive sampling schemes. Technical Report 2005-6, University Paris Dauphine, 2005.

[20] R. Douc, A. Guillin, J. M. Marin, and C. P. Robert. Minimum variance importance sampling via population Monte Carlo. Technical report, University Paris Dauphine, 2005.

[21] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[22] A. Doucet, De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo Methods in Practice*, pages 3–14, 2001.

[23] Thomas Driemeyer. *Rendering with Mental Ray*. Springer, 2nd edition, 2001.

[24] Philip Dutré and Yves D. Willems. Importance-driven Monte Carlo light tracing. In *Proceedings of the 5th Eurographics Workshop on Rendering*, pages 185–194, 1994.

[25] Philip Dutré and Yves D. Willems. Potential-driven Monte Carlo particle tracing for diffuse environments with adaptive probability functions. In *Rendering Techniques '95: Proceedings of the Sixth Eurographics Workshop on Rendering*, pages 306–315, 1995.

[26] Shaohua Fan, Stephen Chenney, Bo Hu, Kam-Wah Tsui, and Yu-chi Lai. Optimizing control variate estimators for rendering. *Computer Graphics Forum (Proceedings of Eurographics 2006)*, 25(3), 2006. To appear.

[27] Shaohua Fan, Stephen Chenney, and Yu-chi Lai. Metropolis photon sampling with optional user guidance. In *Rendering Techniques'05: Proceedings of the Eurographics Symposium on Rendering*, pages 127–138. Eurographics Association, 2005.

[28] Shaohua Fan, Stephen Chenney, and Yu-chi Lai. Population Monte Carlo rendering, 2006. Under review.

[29] Jean-Philippe Farrugia and Bernard Péroche. A progressive rendering algorithm using an adaptive perceptually based image metric. *Computer Graphics Forum (Proceedings of Eurographics 2004)*, 23(3):605–614, 2004.

[30] James A. Ferwerda, Sumanta N. Pattanaik, Peter Shirley, and Donald P. Greenberg. A model of visual masking for computer graphics. In *SIGGRAPH '96:Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 249–258, 1996.

[31] Walter R Gilks, Sylvia Richardson, and David J Spiegelhalter. *Markov chain Monte Carlo in Practice*. Chapman & Hall, 1996.

[32] A. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, 1995.

[33] Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg, and Bennett Battaile. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84: Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, pages 213–222, 1984.

[34] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

[35] Eric Haines. An introductory tour of interactive rendering. *IEEE Computer Graphics and Applications*, 26(1):76–87, 2006.

[36] A. Hall. On an experimental determination of π. *Messeng. Math.*, 2:113–114, 1873.

[37] J. H. Halton. On the relative merits of correlated and importance sampling for Monte Carlo integration. *Proceedings of the Cambridge Philosophical Society*, 61:497–498, 1965.

[38] Hammersley and Handscomb. *Monte Carlo Methods*. John Wiley & Sons, 1965.

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

[40] Paul S. Heckbert. Adaptive radiosity textures for bidirectional ray tracing. In *SIGGRAPH '90: Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, pages 145–154, 1990.

[41] T. Hesterberg. *Advanced in Importance Sampling*. PhD thesis, Stanford University, 1988.

[42] Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37:185–194, 1995.

[43] David S. Immel, Michael F. Cohen, and Donald P. Greenberg. A radiosity method for non-diffuse environments. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 133–142, 1986.

[44] Henrik Jensen. Global illumination using photon maps. In *Rendering Techniques'96: Proceedings of the Eurographics Workshop on Rendering*, pages 21–30, 1996.

[45] Henrik Wann Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001.

[46] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. In *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, pages 311–320, 1998.

[47] James T. Kajiya. The rendering equation. In *SIGGRAPH '86: Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 143–150, 1986.

[48] Kalos and Whitlock. *Monte Carlo Methods, Volume I: Basics*. John Wiley & Sons, 1986.

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

[50] Alexander Keller and Ingo Wald. Efficient importance sampling techniques for the photon map. In *Proc. Vision, Modelling and Visualization 2000*, pages 271–279, 2000.

[51] David Kirk and James Arvo. Unbiased sampling techniques for image synthesis. In *SIGGRAPH '91: Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques*, pages 153–156, 1991.

[52] Genshiro Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, March 1996.

[53] Thomas Kollig and Alexander Keller. Efficient bidirectional path tracing by randomized Quasi-Monte Carlo integration. In K.-T. Fang, F.J. Hickernell, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods*, pages 290–305. Springer-Verlag, 2000.

[54] Eric P. Lafortune and Yves D. Willems. Bidirectional path tracing. In *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, 1993.

[55] Eric P. Lafortune and Yves D. Willems. The ambient term as a variance reducing technique for Monte Carlo ray tracing. In *Photorealistic Rendering Techniques (Proceedings of the Fifth Eurographics Workshop on Rendering)*, pages 168–176, 1994.

[56] Eric P. Lafortune and Yves D. Willems. A 5D tree to reduce the variance of Monte Carlo ray tracing. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)*, pages 11–20, 1995.

[57] Mark E. Lee, Richard A. Redner, and Samuel P. Uselton. Statistically optimized sampling for distributed ray tracing. In *SIGGRAPH '85: Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, pages 61–68, 1985.

[58] Jun Liu, Rong Chen, and Tanya Logvinenko. A theoretical framework for sequential importance sampling and resampling. *Sequential Monte Carlo in Practice*, pages 225–246, 2001.

[59] Jun S. Liu. *Monte Carlo Strategies in Scientific Computing*. Spinger-Verlag, 2001.

[60] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[61] Barbara J. Meier. Painter rendering for animation. In *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 477–484, 1996.

[62] N. Metropolis. The beginning of the Monte Carlo method. In Necia Grant Cooper, editor, *From Cardinals to Chaos: reflections on the life and legacy of Stanislaw Ulam*. Cambridge University Press, 1989. Los Alamos Science Special Issue (15): Stan Ulam, 1909-1984. Book was published in 1989.

[63] N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44:335–341, 1949.

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

[65] Don P. Mitchell. Generating antialiased images at low sampling densities. In *SIGGRAPH '87: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, pages 65–72, 1987.

[66] Pierre Moral, Arnaud Doucet, and Ajay Jasra. Sequential Monte Carlo samplers. *J. Royal Statist. Soc. B*, 68:1–26, 2006.

[67] F. E. Nicodemus, J. RICHMOND, J. HSIA, I. GINS-BERG, and T. LIMPERIS. Geometric considerations and nomenclature for reflectance. *Monograph*, page 71, 1977.

[68] Ryutarou Ohbuchi and Masaki Aono. Quasi-Monte Carlo rendering with adaptive sampling. Technical Report RT0167, IBM Tokyo Research Laboratory, 1996.

[69] Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95:135–143, 2000.

[70] James Painter and Kenneth Sloan. Antialiased ray tracing by adaptive progressive refinement. In *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, pages 281–288, 1989.

[71] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis light transport for participating media. In *Rendering Techniques '00 (Proceedings of the 11th Eurographics Workshop on Rendering)*, pages 11–22, 2000.

[72] Ingmar Peter and Georg Pietrek. Importance driven construction of photon maps. In *Rendering Techniques '98 (Proceedings of the 9th Eurographics Workshop on Rendering)*, pages 269–280, 1998.

[73] Matt Pharr and Greg Humphreys. *Physically Based Rendering from Theory to Implementation*. Morgan Kaufmann, 2004.

[74] Georg Pietrek and Ingmar Peter. Adaptive wavelet densities for Monte Carlo ray tracing. In V. Skala, editor, *WSCG'99 Conference Proceedings*, pages 217–224, 1999.

[75] Werner Purgathofer. A statistical method for adaptive stochastic sampling. In *Proceedings EUROGRAPHICS 86*, pages 145–152, 1986.

[76] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. In *SIGGRAPH '99: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, pages 73–82, 1999.

[77] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 267–276, 2002.

[78] Jaume Rigau, Miquel Feixas, and Mateu Sbert. New contrast measures for pixel supersampling. In *Proceedings of CGI'02*, pages 439–451. Springer-Verlag, 2002.

[79] Jaume Rigau, Miquel Feixas, and Mateu Sbert. Entropy-based adaptive sampling. In *Proceedings of Graphics Interface 2003*, pages 149–157, 2003.

[80] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer-Verlag, 2nd edition, 2004.

[81] Donald Rubin. A noniterative sampling/importance resampling alternative to the data augmentation algorithm for creating a few imputations when fractions of missing information are modest: the SIR algorithm. *Journal of the American Statistical Association*, 82:543–546, 1987.

[82] R. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley & Sons, 1981.

[83] Christophe Schlick. An adaptive sampling technique for multidimensional integration by ray-tracing. In *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, pages 21–29, 1991.

[84] Peter Shirley, Bretton Wade, Phillip Hubbard, David Zareski, Bruce Walter, and Donald Greenberg. Global illumination via density-estimation. In *Rendering Techniques '95 (Proceedings of the 6th Eurographics Workshop on Rendering)*, pages 219–230, 1995.

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

[86] Jerome Spanier and Ely M. Gelbard. *Monte Carlo principles and neutron transport problems*. Reading, Mass., Addison-Wesley Pub. Co, 1969.

[87] William A. Stokes, James A. Ferwerda, Bruce Walter, and Donald P. Greenberg. Perceptual illumination components: a new approach to efficient, high quality global illumination rendering. In *SIGGRAPH '04: Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques*, pages 742–749, 2004.

[88] Frank Suykens. *On Robust Monte Carlo Algorithms for Multi-pass Global Illumination*. PhD thesis, Computer Science, K.U. Leuven, Belgium, 2002.

[89] Frank Suykens and Yves D. Willems. Density control for photon maps. In *Rendering Techniques '00 (Proceedings of the 11th Eurographics Workshop on Rendering)*, pages 23–34, 2000.

[90] László Szécsi, Mateu Sbert, and László Szirmay-Kalos. Combined correlated and importance sampling in direct light source computation and environment mapping. *Computer Graphics Forum (Proceedings of the Eurographics 2004)*, 23(3):585–593, 2004.

[91] L. Szirmay-Kalos, F. Csonka, and Gy. Antal. Global illumination as a combination of continuous random walk and finite-element based iteration. *Computer Graphics Forum (Proceedings of the Eurographics 2001)*, 20(3):288–298, 2001.

[92] László Szirmay-Kalos. Monte Carlo methods for global illumination. In *Spring Conference of Computer Graphics99*, pages 1–28, 1999. Invited talk.

[93] Eric Tabellion and Arnauld Lamorlette. An approximate global illumination system for computer generated films. In *SIGGRAPH '04: Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques*, pages 469–476, 2004.

[94] Justin Talbot, David Cline, and Parris Egbert. Importance resampling for global illumination. In *Rendering Techniques '05: Proceedings of the Eurographics Symposium on Rendering*, pages 139–146, 2005.

[95] Rasmus Tamstorf and Henrik Wann Jensen. Adaptive sampling and bias estimation in path tracing. In *Rendering Techniques '97: Proceedings of the Eighth Eurographics Workshop on Rendering*, pages 285–296, 1997.

[96] Steven Thompson and George Seber. *Adaptive Sampling*. New York, Wiley, 1996.

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

[98] Parag Tole. *Two Algorithms for Progressive Computation of Accurate Global Illumination*. PhD thesis, Cornell University, 2003.

[99] Parag Tole, Fabio Pellacini, Bruce Walter, and Donald P. Greenberg. Interactive global illumination in dynamic scenes. In *SIGGRAPH '02: Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, pages 537–546. ACM Press, 2002.

[100] Eric Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

[101] Eric Veach and Leonidas J. Guibas. Bidirectional estimators for light transport. In *Rendering Techniques '94 (Proceedings of the 5th Eurographics Workshop on Rendering)*, pages 147–162, 1994.

[102] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques*, pages 419–428, 1995.

[103] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *SIGGRAPH '97: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 65–76, 1997.

[104] Gregory Ward. Adaptive shadow testing for ray tracing. In *Photorealistic Rendering in Computer Graphics (Proceedings of the Second Eurographics Workshop on Rendering)*, pages 11–20, 1991.

[105] Gregory J. Ward. The RADIANCE lighting simulation and rendering system. In *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 459–472, 1994.

[106] Gregory J. Ward and Paul Heckbert. Irradiance gradients. In *Proceedings of the 3rd Eurographics Workshop on Rendering*, pages 85–98, 1992.

[107] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. In *SIGGRAPH '88: Proceedings of the 15th Annual Conference on Computer Graphics and Interactive Techniques*, pages 85–92, 1988.

Appendix A: Statistical Proofs

A.1 Relationships among DDIS, MIS and DMS

Given a set of sampling techniques, $p_1, \cdots, p_m$, Deterministic Defensive Importance Sampling (DDIS), Multiple Importance Sampling (MIS) and and Deterministic Mixture Sampling (DMS) all provide a way for constructing estimators that combine the samples. DDIS uses a linear combination of samples, MIS uses different weighting methods to combine samples, while DMS incorporates both correlated sampling and importance sampling. With some derivation, it can be shown that DDIS is a special case of MIS and MIS with balance heuristic weights is a special case of DMS.

- DDIS

  Let $p_\alpha(X) = \sum_{j=1}^{m} \alpha_j p_j(X), \sum_{j=1}^{m} \alpha_j = 1$. One takes $n_j = n\alpha_j$ samples from the density p_j. Let $X_{ji} \sim p_j$ be independent, for $j = 1, \cdots, m$ and $i = 1, \cdots, n_j$. The estimator for DDIS is

  $$\hat{I}_{DDIS} = \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji})}{p_\alpha(X_{ji})} \qquad \text{[A.1]}$$

  Plugging $p_\alpha(X) = \sum_{k=1}^{m} \alpha_k p_k(X)$ and $\alpha_k = \frac{n_k}{n}$ into Equation A.1, we have

  $$\begin{aligned} \hat{I}_{DDIS} &= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji})}{\sum_{k=1}^{m} \alpha_k p_k(X_{ji})} \\ &= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji})}{\sum_{k=1}^{m} \frac{n_k}{n} p_k(X_{ji})} \\ &= \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji})}{\sum_{k=1}^{m} n_k p_k(X_{ji})} \\ &= \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{n_j f(X_{ji})}{\sum_{k=1}^{m} n_k p_k(X_{ji})} \end{aligned} \qquad \text{[A.2]}$$

- MIS using balance heuristic weights

Let n_j to be the number of samples from p_j, and $\sum_{j=1}^{m} = n$. Let $X_{ji} \sim p_j$ be independent, for $j = 1, \cdots, m$ and $i = 1, \cdots, n_j$. The estimator for MIS is

$$\hat{I}_{MIS} = \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} w_j(X_{ji}) \frac{f(X_{ji})}{p(X_{ji})} \quad \text{[A.3]}$$

where $0 \leq w_j(x) \leq \sum_{j=1}^{m} w_j(x) = 1$.

The balance heuristic weights for MIS are:

$$w_j(x) = \frac{n_j p_j(x)}{\sum_{k=1}^{m} n_k p_k(x)} \quad \text{[A.4]}$$

So the MIS estimator with balance heuristic is

$$\begin{aligned} \hat{I}_{MISBal} &= \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{n_j p_j(X_{ji})}{\sum_{k=1}^{m} n_k p_k(X_{ji})} \frac{f(X_{ji})}{p_j(X_{ji})} \\ &= \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{n_j f(X_{ji})}{\sum_{k=1}^{m} n_k p_k(X_{ji})} \end{aligned} \quad \text{[A.5]}$$

where $0 \leq w_j(x) \leq \sum_{j=1}^{m} w_j(x) = 1$.

- DMS

Let $p_\alpha(X) = \sum_{j=1}^{m} \alpha_j p_j(X), \sum_{j=1}^{m} \alpha_j = 1$. One takes $n_j = n\alpha_j$ samples from the density p_j. Let $X_{ji} \sim p_j$ be independent, for $j = 1, \cdots, m$ and $i = 1, \cdots, n_j$. The estimator for DMS is

$$\begin{aligned} \hat{I}_{DMS} &= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji}) - \sum_{k=1}^{m} \beta_k p_k(X_{ji})}{p_\alpha(X_{ji})} + \sum_{j=1}^{m} \beta_j \\ &= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji}) - \sum_{k=1}^{m} \beta_k p_k(X_{ji})}{\sum_{i=1}^{m} \alpha_i p_i(X_{ji})} + \sum_{i=1}^{m} \beta_i \end{aligned} \quad \text{[A.6]}$$

If we set $\beta_j = 0$ and plug in $\alpha_j = \frac{n_j}{n}$, we have

$$\begin{aligned} \hat{I}_{DMS} &= \frac{1}{n} \sum_{j=1}^{m} \sum_{i=1}^{n_j} \frac{f(X_{ji})}{\sum_{k=1}^{m} \frac{n_j}{n} p_k(X_{ji})} \\ &= \sum_{j=1}^{m} \frac{1}{n_j} \sum_{i=1}^{n_j} \frac{n_j f(X_{ji})}{\sum_{k=1}^{m} n_k p_k(X_{ji})} \end{aligned} \quad \text{[A.7]}$$

$\hat{I}_{DDIS}$ in Equation A.2 is the same as $\hat{I}_{MISBal}$ in Equation A.5, so DDIS is same as MIS with balance heuristic weights. One difference between DDIS with MIS is that the coefficients in DDIS are used to determine how the samples are generated and combine the samples afterwards, while the weights in MIS are only used to combine the samples.

Comparing $\hat{I}_{MISBal}$ in Equation A.5 and $\hat{I}_{DMS}$ in Equation A.7, it is obvious to see the MIS using balance heuristic weights is a special case of DMS with $n_j = n\alpha_j$ and $\beta_j = 0$. It means DMS with optimal $\beta' s$ is always better than MIS using balance heuristic.