Numéro d'ordre: 197 Année 2007

THESE

présentée devant

L'Université Claude Bernard - Lyon I

pour l'obtention du

Diplôme de Doctorat
(arrêté du 7 août 2006)

présentée et soutenue publiquement le 26 octobre 2007 par

Benjamin SEGOVIA

Interactive Light Transport with Virtual Point Lights

Directeur de Thèse: Bernard PÉROCHE

Jury:

Président:	M. Jean-Michel JOLION
Rapporteurs:	M. George DRETTAKIS
	M. Nicolas HOLZSCHUCH
Examinateurs:	M. Jean-Claude IEHL
	M. Mathias PAULIN
Invité:	M. Raphaël LABAYRADE

To Anne . . .
who relentlessly helped me

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to thank all the people who have helped me in one way or another during the three years of my Ph.D. thesis. First, I would like to express my gratitude to my advisor, Bernard Péroche, for his support and encouragements over the last three years: he always pushed me towards new theoretical challenges and finally created the necessary conditions to achieve a valuable academic work.

I would also like to thank my co-advisor, Jean-Claude Iehl: he was always available for help and suggestions and participated in several projects during these three years. Without his help, the projects I developed, including yaCORT, would not have been possible. He was also a great teacher and I learnt many things about computer graphics and computer science with him.

Similarly, I have to thank (in random order), Brice Michoud, Jean-Philippe Farrugia and Erwan Guillou for the great discussions about computer graphics we had together; David Sibai and Jean-Baptiste Bianquis for their helps as proof readers for all the articles I submitted during three years; Vincent Forest and Tamy Boubekeur, my conference fellows, who made every Eurographics, a pleasant moment.

Finally, I would like to thank my parents and sisters for their love, guidance and support over the years. Most importantly, my deepest gratitude goes to Anne who had to bear the stress of having me write this thesis and without whose great patience, this Ph.D. would not have been possible. I dedicate this thesis to her.

Abstract of the Dissertation

Interactive Light Transport with Virtual Point Lights

by

Benjamin Segovia
Doctor of Philosophy in Computer Science
Université Lyon I, 2007

Since about twenty years, the field of computer graphics has been striving towards increased realism. The goal is to achieve photorealistic outputs, which actually involves to precisely describe lighting phenomena, their intrinsic models and the light transport equations and to finally propose numerical schemes to solve them. In the field of *off-line* rendering, efficient and aggressive numerical methods have already been explored and set up. Therefore, even if there is still room for improvement, the most challenging and appealing problems are today related to *real-time rendering* which consists in ensuring to display at least 20 frames per second with some interactivity offered to the user.

In this Ph.D. thesis, I therefore focus on the real-time or, if not possible, interactive simulation of light transport phenomena. During the three years while I tried to propose new ideas, I thus mainly aimed at efficiently using current existing hardware systems, GPUs or CPUs and designing new numerical schemes which may be easily implemented in such systems. It is finally the most common role of numericians: making a subtle mixture of mathematical and algorithmic methods and efficiently combining them.

CHAPTER 1

Introduction

The goal of this Ph.D. thesis is to propose fast numerical algorithms to solve the light transport problem with a set of virtual point lights. Our motivation is to replace the whole incoming radiance field which is generally a quite complex function by a simpler set of virtual point lights which illuminates the parts of the scene seen by the camera. Once the set has been sampled, computing the final picture becomes much easier since all lighting contributions, those due to physical light sources or those due to light ray bounces on the surfaces of the scene, can be obtained by checking if one point seen by the camera can see the virtual point light.

The majority of this thesis actually focuses on two points. The first point studies how the final image can be computed once the set of virtual point lights is given. Two implementations are presented: the first one uses different rasterization techniques and GPUs while the second one proposes to use ray tracing and CPUs. The second point proposes different manners to compute an effective set of virtual point lights. Indeed, the properties of each point light (their positions, their "virtual" radiance outgoing functions and so on ...) are major issues. For example, imagine that only a small part of a building is seen by the camera; if virtual point lights are stored on surfaces of the scene which do *not* illuminate the parts of the scene seen by the camera, all the computations which will be done with those point lights will be useless. To achieve this goal, we will focus on Monte-Carlo techniques i.e. on probabilistic numerical schemes which randomly deposit virtual point lights on the surfaces of the scene with interesting densities.

The remainder of the chapter presents the light transport problem in an informal manner and explains why light transport is an interesting challenge with useful applications for movie productions, video games or lighting design. We will finally sum up our contributions (already published or not).

1.1 The Light Transport Problem

Computing synthesized pictures in a fast manner is today an important challenge. Even if major progresses have been done during the last two decades, ensuring high-quality and fast rendering of virtual scenes still remains an open problem.

Movie Rendering. Rendering computer-synthesized special effects or complete scenes during movie production is strongly limited by the computation time available for each frame. This makes the developers and the artists implement more restrictive algorithms (which, for example, do not handle correctly all lighting phenomena like non-diffuse reflections or indirect lighting configurations) or to correct the weakness of the numerical schemes by adding artificial light sources. Providing new rendering techniques to improve movie rendering quality, decrease production time or more surely, decrease production costs is therefore a major challenge.

Lighting Design. Lighting design which concerns either the illumination of architecture of spaces such as gardens or public squares (*Architectural Lighting Design*), the overall aspect within a theater, or more generally the conception of lights (*Industrial Design*) requires both exact solutions and totally predictive behaviors of the numerical algorithms. Indeed, the lighting configurations are often particularly awkward. We can give two classical examples. The first one is SERRAGLAZE, a complex light redirection system which consists in two thin sheets of acrylic incorporating microreplicated prisms bonded together to create microscopic air pockets. This particular configuration tends to increase daylight at the rear of the room and to provide more uniform lighting. The second example is the classical headlight used in all cars. The light ray must often bounce more than ten times before going out of the headlight thereby creating complex caustics difficult to simulate. These two examples illustrate the necessity to make robust numerical algorithms. Speed seems less important but once again, this can lead to smaller rendering times and therefore large cost reductions.

Video Games. Contrary to lighting design, video games require a high frame rate and the rendering quality is therefore completely bounded by the hardware performance. Making more realistic games by improving light transport numerical schemes and adapting them to the current existing hardware is therefore a major challenge.

These problems therefore strongly motivate our work and the Ph.D. thesis presented here.

1.2 Summary of Contributions

Our contributions fall into three areas: a revisited formalism for the light transport problem to classify most of Monte-Carlo rendering techniques, new implementations of rendering algorithms using existing hardware and new statistical numerical schemes.

1.2.1 A Formalism for the Light Transport Problem

We will present in this section of the thesis how the light transport problem can be formalized using the path integral formulation. Even if it is not really new, we will expose how most Monte-Carlo algorithms can be sorted into two major classes. We will show that this can lead to a simple understanding of all rendering techniques which use Monte-Carlo numerical schemes.

1.2.2 New Rendering Techniques using GPUs or CPUs

In this section, we expose new rendering techniques which efficiently use either GPUs or CPUs to perform the illumination of the scene due to a set of virtual point lights.

Deferred Shading Techniques using Interleaved Sample Patterns. We first show that uncorrelated light contributions or more generally uncorrelated computations can be achieved with a GPU for nearby pixels. Two implementations are presented: The first one extends the deferred shading rendering technique in a general way such that every existing algorithm already using deferred shading can be accelerated. The second one is more restrictive but faster and can easily speed up common rendering techniques such as the generation of soft shadows or the approximate rendering of indirect illumination effects.

Coherent Ray Tracing Techniques. We will also expose alternative rendering techniques using ray tracing and CPUs. Even if most of the presented approaches are not new, we show that performing illumination with virtual point lights and ray tracing is more efficient that performing it with rasterization techniques on GPUs.

1.2.3 New Numerical Schemes to Compute Virtual Point Light Sets

We present two new numerical schemes to compute an efficient set of virtual point lights.

Bidirectional Instant Radiosity. We first present a bidirectional way to generate virtual point lights. A first part of the set is generated from the camera while the second one is generated from the light sources. We show that this method provides efficient sample sets which can handle either direct or indirect lighting configurations.

Metropolis Instant Radiosity. We then expose a new way to generate virtual point lights by considering the virtual point light sampler as a Markovian process with a useful invariant distribution: each virtual point light brings the same amount of energy to the camera. This technique provides satisfactory results with many different input layouts.

Coherent Metropolis Light Transport with Multiple-Try Mutations. We finally present an alternative technique to solve the light transport problem in a fast way. Indeed, even if virtualizing the radiance field with a point light set remains very efficient, our experiments make us think that it often brings only a partial solution. This motivates a completely different approach which does not use virtual point lights but a set of complete light paths which can be perturbed in a coherent and fast manner.

1.3 Thesis Organization

The first chapter presented here, motivates the other parts of this thesis, explains why the light transport problem remains a challenging problem and summarizes our contributions. Chapter 2 presents the mathematical roots of all numerical methods presented in the remainder of the thesis: probability theory and Monte-Carlo methods. Chapter 3 exposes the physics of light transport. In Chapter 4, we show how Monte-Carlo techniques can be applied to solve the light transport problem. Next chapters focus on our contributions which are mostly already published. Chapters 5 and 6 present two GPU techniques to perform uncorrelated computations on nearby pixels. Chapters 7 and 8 explore new sampling strategies for the generation of virtual point light sources. In Chapter 8, we present an alternative technique: instead of tackling variance problems with virtual point lights, we conversely try to make the successful algorithm, Metropolis Light Transport, more coherent and faster. Chapter 10 concludes and exposes some possible future work.

CHAPTER 2

Monte-Carlo Integration

Since all the numerical schemes presented in this thesis rely on Monte-Carlo methods, we present in this chapter some basic notions of probability theory and a short overview of Monte-Carlo integration methods that are commonly used in computational statistics and in Monte-Carlo rendering.

2.1 Introduction

Monte Carlo methods are a large class of computational algorithms: they are actually statistical simulation methods which use sequences of random numbers that, conversely to many other simulation approaches, make the simulation non-deterministic. Today, Monte-Carlo methods are certainly ones of the most popular computational techniques. Indeed:

- Because of the repetition of algorithms and the large number of calculations involved, Monte Carlo is suited to computer calculation;
- Conversely to deterministic numerical methods such as quadrature rules, Monte-Carlo methods remain robust as the dimension of the problem increases;
- Monte-Carlo methods do not require that the estimated quantities are smooth. Only a measurability criterion is most of the time necessary.

For these reasons, Monte Carlo methods are today successfully used in many different fields such as radiation transport, the simulation of the sub-nuclear processes, financial mathematics or light transport simulation.

The name ”Monte Carlo” was actually given by Nick Metropolis (inspired by Ulam’s interest in poker) during the Manhattan Project of World War II, because of the similarity of statistical simulation to games of chance, and since the capital of Monaco was the world center for gambling and casino games.

Even if most of the modern ideas related to Monte-Carlo methods was thus invented after World War II, there are several isolated instances on earlier occasions. For instance, in 1777, the Comte de Buffon performed experiments in which he threw a needle onto a board ruled with parallel straight lines to estimate the value of π from observations of the number of intersections between the needle and lines. In 1899, Lord Rayleigh showed that a one-dimensional random walk without absorbing barriers provides a solution to a parabolic differential equation. In 1931, Kolmogorov proved the relationship between Markov stochastic processes and integro-differential equations and therefore invented stochastic calculus. In early part of the twentieth century, British statistical schools indulged in a fair amount of unsophisticated Monte Carlo work. Even if most of this work seems to have been first didactic and actually not used for research, original discoveries were done. In 1908, Student (W.S. Gosset) used experimental sampling to help him towards the discovery of the distribution of the correlation coefficient. In the same year, he also used sampling to guess his so-called t-distribution. It is however just after the creation of the first electronic computer in 1946, that Monte-Carlo methods have been becoming popular: to design thermo-nuclear weapons, Los Alamos scientists and more particularly, Stanislaw Ulam, John Von Neumann and Nick Metropolis, set up the roots of all Monte-Carlo techniques which are used today.

Historically, Monte-Carlo methods have been first used to solve problems of a non-deterministic nature which involve random processes. For the neutron transport problem where the neutron behavior is random, the use of Monte-Carlo simulations can provide the value of physical quantities such as average neutron fluxes. It mainly consists of simulating the microscopic physical phenomena by generating microscopic particles and simulating their random behavior: the simulation therefore replicates the non-deterministic particle trajectory. By averaging all simulated results, one can retrieve meaningful physical quantities. Similar approaches have been used in many other scientific areas such as financial mathematics where stochastic calculus is actually used to obtain the fair values of derivatives of the stock.

Conversely, Monte-Carlo methods can also be applied to problems of a deterministic nature as it is done with Monte-Carlo rendering, i.e. the application of Monte-Carlo methods to the generation of virtual pictures. The approach often called, sophisticated Monte-Carlo, simply consists in using the Monte-Carlo background to solve difficult problems that classical deterministic approaches cannot handle. It is for instance particularly true for high-dimensional problems or integration problems where the integrand is not sufficiently smooth.

One can finally find more insightful discussions in the books by Hammersley and Handscomb [33], by Kalos and Whitlock [43], by Shreider [73] and by Rubinstein [69]. The book by Spanier and Gelbard [76] is a classical reference for Monte-Carlo methods and their applications to neutron transport problems and it is also a great source of inspiration for the application of Monte-Carlo techniques to the light transport problem.

2.2 Some Probability Theory

In this section, we present the basic roots of probability theory, some notions of discrete and continuous probabilities.

2.2.1 Discrete Probability Theory

Historically, probability theory was first studied within its simplest form, i.e. the form only involving a finite set of possible states. First, let $\Omega = \{x_1, x_2, \dots\}$ be a sample space. Then, for $x \in \Omega$, we define a probability distribution $f(x)$ such that:

1. $f(x) \in [0, 1]$ for all $x \in \Omega$
2. $\sum_{x \in \Omega} f(x) = 1$

An event is then defined as any subset $\Lambda \in \mathrm{P}(\Omega)$ of Ω and the probability $P(\Lambda)$ of event Λ is given by:

$$\mathrm{P}(\Lambda) = \sum_{x \in \Lambda} f(x)$$

The probability of the entire sample space is therefore equal to 1 and the probability of the null event is equal to 0. More generally, discrete probabilities easily handle problems like finding the number of occurrences of an even number when a die is rolled, the configuration of a discrete random walk on $\mathbb{Z}^2$, specific experiments with chance games like Poker or the very popular Texas Hold'em. For our particular application, the simulation of light transport phenomena, the discrete probability theory is insufficient since we have to deal with continuous energy state spaces. This leads us to introduce the general probability theory.

2.2.2 General Probability Theory

General probability theory (which includes discrete and continuous probability theories) deals with events that occur in a general sample space. It was mostly introduced and formalized by Kolgomorov at the beginning of the twentieth century. Even if it may seem useless to expose general probabilities, we think that it is important to remind the underlying definitions and theorems required to formalize a given problem as a "Monte-Carlo problem" which can directly be solved by the techniques presented further in this thesis. As we show it, it is fundamental to define the considered sample space (which is not necessarily trivial in the particular case of global illumination problem), its associated measure and finally, the functions we specifically want to integrate.

2.2.2.1 Probability Space $(\Omega,\ \mathcal{F},\ \mathrm{P})$

We first define a sample set Ω to which is associated a σ-algebra $\mathcal{F} \in \mathcal{P}(\Omega)$ such that:

1. $\Omega \in \mathcal{F}$
2. if $\Lambda \in \mathcal{F}$ then $\Lambda^c \in \mathcal{F}$
3. if $(\Lambda_n)_{n\in\mathbb{N}} \in \mathcal{F}^{\mathbb{N}}$, then $\bigcup_{n\in\mathbb{N}} \Lambda_n \in \mathcal{F}$

We then define a probability measure P on $(\Omega, \mathcal{F})$ such that:

1. $\mathrm{P}(\Omega) = 1$
2. if $(\Lambda_0, \Lambda_1) \in \mathcal{F}^2$ and $\Lambda_0 \cap \Lambda_1 = \emptyset$ then $\mathrm{P}(\Lambda_0 \cup \Lambda_1) = \mathrm{P}(\Lambda_0) \cup \mathrm{P}(\Lambda_1)$
3. if $(\Lambda_n)_{n\in\mathbb{N}} \in \mathcal{F}^{\mathbb{N}}$ and $\Lambda_n \subset \Lambda_{n+1}$, then $\mathrm{P}(\bigcup \Lambda_n) = \lim_{n\to\infty} \mathrm{P}(\Lambda_n)$

If a σ-algebra is associated to a given space E (for example the Borel σ-algebra, for the real numbers, $\mathbb{R}^n$ or an euclidean space), we define a *random variable* as a measurable function $X : \Omega \to E$.

If X is a *real* random variable (i.e. $E = \mathbb{R}$), we say that X has a first-order moment if and only if $\int_\Omega |X(\omega)| d\mathrm{P}(\omega) < \infty$. In this case, we define the *expectancy* (or the *mean*) of X by:

$$\mathrm{E}[X] = \int_\Omega X(\omega) d\mathrm{P}(\omega)$$

Ω may be finally considered as the "source" of the randomness of X. As this form does not lead to a computational expression of a random variable, it is practical to define the distribution, the probability density function and the cumulative distribution function notions. Before presenting them in the next subsections, we shortly remind some important theorems from the measure theory.

Monotonous Convergence Theorem. If $(X_n)_{n\in\mathbb{N}}$ is a positive increasing random variable sequence (i.e. $\forall n \in \mathbb{N},\ 0 \leq X_n \leq X_{n+1}$) and if $X_\infty = \lim_{n\to\infty} X_n$, then:

$$\mathrm{E}[X_\infty] = \lim_{n\to\infty} \mathrm{E}[X_n]$$

Fatou Lemma. If $(X_n)_{n\in\mathbb{N}}$ is a positive random variable sequence, then:

$$\mathrm{E}[\liminf_{n\to\infty} X_n] \leq \liminf_{n\to\infty} \mathrm{E}[X_n]$$

Dominated Convergence Theorem. If $(X_n)_{n\in\mathbb{N}}$ is a positive random variable sequence which converges towards X_∞ such that $\mathrm{P}\left(\lim_{n\to\infty} X_n = X_\infty\right) = 1$, and if it exists a random variable Y such that $\forall n \in \mathbb{N},\ \mathrm{P}\left(X_n \leq Y\right) = 1$, then:

$$\mathrm{E}[X_\infty] \;=\; \lim_{n\to\infty} \mathrm{E}[X_n]$$

2.2.2.2 Probability Measure and Probability Density Function

Let $X : (\Omega, \mathcal{F}) \to (E, \mathcal{E})$ be a random variable and P_X denote the probability measure on Ω such that:

$$\forall A \in \mathcal{E},\ P_X(A) = \mathrm{P}_X(X \in A) = \mathrm{P}(\{\omega \in A : X(\omega) \in A\})$$

P_X is the only measure on $(E, \mathcal{E})$ such that for all bounded and measurable functions $f : E \to \mathbb{R}$:

$$\mathrm{E}[f(X)] = \int_E f(x) d\mathrm{P}(x)$$

$d\mathrm{P} : E \to \mathbb{R}$ is called the probability density function (pdf) of X.

2.2.2.3 Some Examples of Distributions

We give here some basic examples of discrete distributions on $\mathbb{R}$:

- the Dirac mass δ_x in $x \in \mathbb{R}$ is the distribution of the random variable which is equal to x almost everywhere;
- the Bernoulli distribution with parameter $p \in [0,1]$ defined by $p\ \delta_1 + (1 - p)\ \delta_0$ with δ_x is the Dirac mass in x.

Some classical examples of distributions on $\mathbb{R}$:

- the Lebesgues measure on $[0,1]$ is a probability measure generally called uniform distribution on $[0,1]$. More generally, if $a < b$, then the uniform law on $[a,b]$ is defined by:

$$(b-a)^{-1}\mathbb{I}_{[a,b]}(x)dx$$

where $\mathbb{I}_{[a,b]}(x) = \begin{cases} 1 \text{ if } x \in [a,b] \\ 0 \text{ otherwise;} \end{cases}$

- the density defined on $\mathbb{R}$ by $\frac{1}{\pi}(1+x^2)^{-1}dx$ is a probability measure on $\mathbb{R}$ called Cauchy distribution;
- the density defined on $\mathbb{R}$ by: $\frac{1}{\sqrt{2\pi}}e^{-x^2/2}dx$ is a probability measure on $\mathbb{R}$ (its mass is equal to 1): it is the "Normal distribution" on $\mathbb{R}$. More generally, for $\sigma > 0$ and $m \in \mathbb{R}$, the density defined by $\frac{1}{\sqrt{2\pi\sigma^2}}e^{\{\frac{-(x-m)^2}{2\sigma^2}\}}$ is a probability measure (its mass is also equal to 1) denoted by $\mathcal{N}(m, \sigma^2)$ and called "Gaussian distribution".

2.2.2.4 Cumulative Distribution Function (CDF) of a Random Variable

If P_X is the law of a real random variable X, we define its cumulative distribution function F_X by:

$$F_X(x) = P_X(]-\infty, x]) = \mathrm{P}(X \le x)$$

For example, the cumulative distribution function of the uniform distribution U on $[0,1]$ is $F_U(x) = x$ if $x \in [0,1]$ and $F_U(x) = 0$ otherwise.

2.2.2.5 Conditional and Marginal Densities

We first define two random variables $X_0 \in (\Omega_0, \mathcal{F}_0, \mathrm{P}_0, \mu_0)$ and $X_1 \in (\Omega_1, \mathcal{F}_1, \mathrm{P}_1, \mu_1)$ and the pair $(X_0, X_1) \in \Omega = (\Omega_0, \Omega_1)$. The probability measure P of (X_0, X_1) is called joint probability measure and the corresponding joint probability density $p(x,y)$ verifies:

$$\forall \Lambda \in (\mathcal{F}_0, \mathcal{F}_1), \mathrm{P}(\Lambda) = \int_\Lambda p(x_0, x_1) d\mu(x_0) d\mu(x_1)$$

The probability density functions p_0 and p_1 of X_0 and X_1 are called marginal density functions of X and satisfy:

$$p_0(x_0) = \int_{\Omega_1} p(x_0, x_1) d\mu(x_1) \text{ and}$$

$$p_1(x_1) = \int_{\Omega_0} p(x_0, x_1) d\mu(x_0)$$

The conditional density $p(x_0|x_1)$ is defined by:

$$p(x_0|x_1) \;=\; \frac{p(x_0, x_1)}{p(x_1)} \text{ and satisfies:}$$

$$p(x_0, x_1) \;=\; p(x_0|x_1)\; p(x_0) \;=\; p(x_1|x_0)\; p(x_1)$$

2.2.2.6 Moments, Expectancy and Variance

If F_X is a cumulative distribution function of a given random variable X, then the nth moment of the probability distribution is given by:

$$\mathrm{E}[X^n] \;=\; \int_{-\infty}^{\infty} x^n \, dF_X(x)$$

If $n \;=\; 1$, $\mathrm{E}[X] \;=\; \mu$ is the expectancy of random variable X. For higher orders, the moments are generally centered on zero and they are defined by:

$$\mu_n \;=\; \mathrm{E}[(X-\mu)^n]$$

The second central moment $\mu_2 \;=\; \sigma^2 \;=\; \mathrm{V}[X]$ is generally called variance of random variable X. From these definitions, we can easily show that:

$$\forall a \in \mathbb{R},\; \mathrm{E}[aX] \;=\; a \cdot \mathrm{E}[X] \text{ and } \mathrm{V}[aX] \;=\; a^2 \cdot \mathrm{V}[X]$$

As the expectancy function E is linear, we also have:

$$\mathrm{E}\left[\sum_{i=0}^{n} X_i\right] = \sum_{i=0}^{n} \mathrm{E}[X_i]$$

Another practical formula to compute the variance is also given by:

$$\mathrm{V}[X] \;=\; \mathrm{E}[(X-\mathrm{E}[X])^2] \;=\; \mathrm{E}[X^2] \;-\; \mathrm{E}[X]^2$$

Some interesting and common remarks can be finally made about moments:

- All moments do not necessarily exist. However, if the nth moment exists, then $\forall k \leq n$, the kth moment exists;
- The moments do not characterize the distribution. Indeed, two different distributions can have equal moments;
- Variance σ^2 is often replaced by the standard deviation $\sigma = \sqrt{\sigma^2}$. It is commonly known as the standard deviation or the Root Mean Square (RMS)

error.

2.2.2.7 Independent Random Variables

Two random variables $X_0 \in (\Omega_0, \mathcal{F}_0, \mathrm{P}_0) \to (E_0, \mathcal{E}_0)$ and $X_1 \in (\Omega_1, \mathcal{F}_1, \mathrm{P}_1) \to (E_1, \mathcal{E}_1)$ are independent if and only if:

$$\forall(\Lambda_0, \Lambda_1) \in (\mathcal{E}_0, \mathcal{E}_1),\ \mathrm{P}(X_0 \in \Lambda_0,\ X_1 \in \Lambda_1) = \mathrm{P}(X_0 \in \Lambda_0) \cdot \mathrm{P}(X_1 \in \Lambda_1)$$

We can also express the independence directly with distributions without involving Probability Space $(\Omega, \mathcal{F}, \mathrm{P})$: two random variables X_0 and X_1 with distributions P_{X_0} and P_{X_1} are said to be independent if and only if the distribution $P_{X_0 \times X_1}$ of (X_0, X_1) satisfies:

$$P_{(X_0 \times X_1)}(\Lambda_0, \Lambda_1) = P_{X_0}(\Lambda_0)\ P_{X_1}(\Lambda_1)$$

This leads to the following necessary and sufficient condition: two random variables are independent if and only if for all bounded and measurable functions $f:\ E_0 \to \mathbb{R}$ and for all bounded and measurable functions $g:\ E_1 \to \mathbb{R}$,

$$\mathrm{E}\left[f(X_0) \cdot g(X_1)\right] \ = \ \mathrm{E}[f(X_0)] \cdot \mathrm{E}[g(X_1)]$$

More particularly, if f and g are both equal to the identity function:

$$\text{if } X_0 \text{ and } X_1 \text{ are independent then } \ \mathrm{E}[X_0 \cdot X_1] = \mathrm{E}[X_0] \cdot \mathrm{E}[X_1]$$

2.2.3 Convergence of Random Variable Families

We remind here the different convergence modes of random variable sequences and most of the theorems which ensure these convergences. As we will show it, the strong law of large numbers and the central limit theorem are the theoretical roots of Monte-Carlo integration and therefore, the roots of most of the algorithms presented in this thesis. In probability theory, the convergence of sequences of random variables to some specific limiting random variable is a major concept with large applications to computational statistics. For example, the average of n uncorrelated and identically distributed random variables $(Y_i)_{i \in \{1,n\}}$ given by $Y_i = \frac{1}{n} \sum_{k=1}^{n} X_k$ almost surely converges to the mean μ of X_i if $E[|X_i|] < \infty$. In the next sections, we assume that $(X_n)_{n \in \mathbb{N}}$ is a sequence of random variables defined on the probability space $(\Omega, \mathcal{F}, \mathrm{P})$ and that X is a given random variable defined on the same probability space.

2.2.3.1 Convergence in Distribution

Suppose that $(X_n)_{n\in\mathbb{N}}$ is a sequence of random variables with cumulative distribution functions F_n and X a random variable with cumulative distribution function F. Then, X_n converges towards X in distribution, if and only if:

$$\forall a \in \mathbb{R}_c,\ \lim_{n\to\infty} F_n(a) = F(a)$$

where $\mathbb{R}_c$ is the set of real numbers where F is continuous. Convergence in distribution is often denoted by adding the letter D over an arrow indicating convergence:

$$X_n \xrightarrow{D} X$$

It is actually the weakest form of convergence: it is therefore implied by all other modes of convergence but rarely implies any other one.

2.2.3.2 Convergence in Probability

$(X_n)_{n\in\mathbb{N}}$ converges towards X in probability if and only if:

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \lim_{n\to\infty} \mathrm{P}\left(|X_n - X| \geq \varepsilon\right) = 0$$

Convergence in probability is commonly denoted by adding the letter P over the convergence arrow:

$$X_n \xrightarrow{P} X$$

Convergence in probability is the notion of convergence used in the *weak* law of large numbers presented further in this chapter. Convergence in probability finally implies convergence in distribution (we do not give any formal proof since it is a bit out of the scope of this thesis).

2.2.3.3 Almost Sure Convergence

$(X_n)_{n\in\mathbb{N}}$ converges almost surely (or almost everywhere as commonly said in measure theory or with probability 1) towards X if and only if:

$$\mathrm{P}\left(\lim_{n\to\infty} X_n = X\right) = \mathrm{P}\left(\left\{\omega \in \Omega : \lim_{n\to\infty} X_n(\omega) = X(\omega)\right\}\right) = 1$$

Almost sure convergence actually implies convergence in probability and it is finally the notion of convergence used in the *strong* law of large numbers also presented further in this chapter.

2.2.3.4 Convergence in Mean

$(X_n)_{n\in\mathbb{N}}$ converges in the r-th mean or in the L_r norm towards X, if and only if:

$$\forall n \in \mathbb{N},\ \mathrm{E|X_n|_r} < \infty,\ \lim_{n\to\infty} \mathrm{E}\left(|X_n - X|^r\right) = 0$$

Finally, if $(X_n)_{n\in\mathbb{N}}$ converges in r-th mean to X for $r = 1$, we say that $(X_n)_{n\in\mathbb{N}}$ converges in mean to X. If $(X_n)_{n\in\mathbb{N}}$ converges in r-th mean to X for $r = 2$, we say that $(X_n)_{n\in\mathbb{N}}$ converges in mean square to X. Convergence in the r-th mean for $r > 0$, implies convergence in probability (once again, we do not give formal proof). If $r > s \geq 1$, convergence in r-th mean also implies convergence in s-th mean and convergence in mean square therefore implies convergence in mean.

2.2.4 Laws of Large Numbers

We present here the two laws of large numbers (weak and strong) which are, with the central limit theorem, the core and the roots of Monte-Carlo integration. First, let S_n denote $\sum_{i=1}^{n} X_i$.

2.2.4.1 The Weak Law of Large Numbers

The weak law of large numbers states that if $(X_n)_{n\in\mathbb{N}}$ is a sequence of independent and identically distributed random variables with expected value $\mu < \infty$ and variance $\sigma^2 < \infty$ then $S_n/n = (X_1 + \cdots + X_n)/n$ converges in probability to μ. Therefore,

$$\forall \varepsilon \in \mathbb{R}^{+*},\ \lim_{n\to\infty} \mathrm{P}\left(|S_n - \mu| < \varepsilon\right) = 1$$

2.2.4.2 The Strong Law of Large Numbers

The strong law of large numbers states that if $(X_n)_{n\in\mathbb{N}}$ is a sequence of independent and identically distributed random variables and that $E[X_i] = \mu$ and $E[|X_i|] < \infty$, then:

$$\mathrm{P}\left(\lim_{n\to\infty} S_n/n = \mu\right) = 1$$

If we finally replace $\mathrm{E}[X_i] < \infty$ by $\mathrm{E}[X_i^2] < \infty$, we obtain a convergence in mean square and therefore an almost sure convergence. For this particular case, we have a simple manner to obtain some properties about the convergence rate.

Indeed, since the variance σ^2 exists:

$$V[S_n/n] = V\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = V\left[\frac{1}{n}\sum_{i=1}^{n} X_i\right] = \frac{1}{n^2}\sum_{i=1}^{n} V[X] = \frac{1}{n}V[X]$$

Therefore, the standard deviation decreases as the square root of the sample number. The strong law is actually the simple explanation of many intuitive phenomena and experiments: when the same sampling process is repeated many times, averaging all results leads to the expected average μ of the measured quantities.

2.2.5 Central Limit Theorem

While the laws of large numbers only give us the convergence of partial sums S_n/n, the central limit theorem also provides more accurate properties concerning the convergence speed. We first suppose that all X_n are independent and identically distributed and that $\forall n \in \mathbb{R}$, $\mu = \mathrm{E}[X_n]$ and $\sigma^2 = \mathrm{E}[X_n^2]$ exist and are finite. Whereas the strong law of large numbers only states that the expected value of S_n/n is μ, the central limit theorem also provides a convergence in distribution. Indeed, if Z_n is defined by $Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}}$, then, the distribution of Z_n converges in distribution towards the standard normal distribution $\mathcal{N}(0,1)$ when $n \to \infty$. In a sense, the distribution of S_n/n approaches the normal distribution $N(\mu, \sigma^2)$ when $n \to \infty$.

Finally, if the third moment $\mathrm{E}[(X_n - \mu)^3]$ exists and is finite, we have a more precise knowledge of the speed of convergence: it is on the order of $1/\sqrt{n}$. Formally, if f is a C^3 function with a bounded third-order derivative ($\exists b \in \mathbb{R}^+$, $\forall x \in \mathbb{R}$, $|f'''(x)| < b$) and N is a random variable which follows $\mathcal{N}(0, \sigma^2)$, then,

$$\mathrm{E}\left[f\left(\frac{S_n - \mu n}{\sqrt{n}}\right)\right] - \mathrm{E}(N) < \mathcal{O}(\frac{1}{\sqrt{n}})$$

This gives us the classical convergence rate of Monte-Carlo integration for smooth enough functions.

2.3 Monte-Carlo Integration

The goal of Monte-Carlo integration is to compute an integral of the form:

$$I \;=\; \int_{\Omega} f(x) d\mu(x)$$

where Ω is a given space and μ an associated measure. Thanks to the strong law of large numbers, if we generate N random variables $(X_n)_{n\in[1\ldots N]}$ with the same probability density function p, then:

$$\lim_{N\to\infty} I_N = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} \frac{f(X_n)}{p(X_n)} = I \text{ almost surely}$$

This convergence is the core of Monte-Carlo integration which therefore simply consists in sampling random variables to estimate integrals.

2.3.1 Advantages and Drawbacks of Monte-Carlo Integration

As indicated in the previous section, if variance σ^2 exists, the variance of I_n decreases in the order of $\frac{1}{\sqrt{n}}$. The central limit theorem gives us a more precise bound. Indeed, $Z_n = \frac{S_n - n\mu}{\sigma\sqrt{n}}$ converges in distribution towards the normal distribution $\mathcal{N}(0,1)$. Finally, if the third moment exists, the order of the speed of converge is exactly $\mathcal{O}(\frac{1}{\sqrt{N}})$.

This speed may seem to be quite slow if we compare Monte-Carlo integration with other common quadrature rules. For example, a very simple Simpson's rule will provide a speed of convergence of an order of $\mathcal{O}(\frac{1}{N^5})$. The other important drawback of Monte-Carlo methods are their intrinsic statistical nature which makes that one cannot be sure of the result. Only confidence intervals (which can be made however as tight as required) can be indeed computed. However, Monte-Carlo methods have several decisive advantages which make them very efficient and consequently, very popular.

- Compared to deterministic integration methods, Monte Carlo methods are better for high-dimensional integrals. Indeed, even if deterministic integration rules of order r have a convergence rate of $\mathcal{O}(N^{-r})$ in one dimension, as the regularly spaced samples have to be distributed over all dimensions, the convergence rate is only on the order of $\mathcal{O}(N^{-\frac{r}{d}})$ in d dimensions.

- Conversely to higher order integration rules which require smooth integrands, Monte-Carlo methods do not need such properties. This is particularly effective in the case of light transport problems where many discontinuities (often due to the geometry of the objects) may occur in the integrands.

- Monte-Carlo integration methods are simple. Indeed, the evaluation of random variables only requires to sample random points X and evaluate $f(X)$. These make the method easily applicable to many integration problems.

Monte Carlo methods are therefore very robust and this leads us to prefer them to classical quadrature rules which do not seem appropriate to the light transport problem which, as we show it in Chapter 4, is both a high-dimensional and "not-smooth integrand" problem.

2.3.2 Sampling Random Variables

There are a large variety of techniques for sampling random variables: the inversion method, the rejection method, and the Metropolis method that we will present further in this thesis are some of them.

2.3.2.1 The Inversion Method

The goal is to generate a random variable X with a given density function $p(x)$. Let $F(x)$ be the cumulative distribution of X. The inversion method consists in choosing a uniform random variable U on $[0 \dots 1]$ and in finding the value of X such that $F(X) = U$ and therefore such that $X = F^{-1}(U)$ where F^{-1} is the inverse function of F. It is then easy to check that the cumulative distribution function of X is F. Indeed,

$$P(X \leq x) = P(F(X) \leq F(x)) = P(U \leq F(x)) = F(x)$$

Uniformly Sampling a Disk. If we are uniformly sampling a disk of radius R, the density function of such a random variable is: $p(r, \theta) = \frac{1}{\pi R^2}$. The two-dimensional distribution function is therefore:

$$P_r(r < r_0 \text{ and } \theta < \theta_0) \;=\; \int_0^{\theta_0} \int_0^{r_0} \frac{r dr d\theta}{\pi R^2} = \frac{\theta_0 r_0}{2\pi R^2}$$

As the density function is here separable, we do need to involve the Jacobian and we can directly apply the inversion on both variables by first computing the marginal distributions of r and θ. Therefore, the uniform pair (u_0, u_1) on $[0,1]^2$ can be transformed to a uniform point on the disk by $(r, \theta) = (R\sqrt{u_0}, 2\pi\sqrt{u_1}))$

Sampling a Phong Lobe. In computer graphics, sampling direction random variables with density $p(\theta, \phi) = \frac{n+1}{2\pi}\cos^n(\theta)$ may be a common situation when glossy or shiny materials occur (n is said to be the Phong exponent, $\theta \in [0, \pi/2]$ is the angle from the surface normal and $\phi \in [0, 2\pi]$ is the azimuthal angle). Once again, by computing the marginal distributions of θ and ϕ, we find that a uniform pair (u_0, u_1) can be directly transformed to a direction by:

$$(\theta, \phi) = \left(\arccos\left[(1-u_0)^{\frac{1}{n+1}}\right], 2\pi u_1\right)$$

2.3.2.2 Rejection Sampling

If the transformation technique is not available (for example, when the cumulative distribution function cannot be analytically inverted), one can use rejection sampling. The algorithm, due to John von Neumann and presented in Algorithm 1, can be actually used with any density function, even those that cannot be integrated analytically. If it accepts the proposed candidate with a high probability, i.e. if $f(x)$ is close to $M \cdot g(x)$, it may be furthermore highly efficient. Unfortunately, if it is applied naively, it may be however very ineffective, since a large number of iterations will be needed to accept a proposed candidate. For example, a too large value for M will lead to a small acceptance rate and therefore a very poor efficiency of the resulting estimator. This drawback makes that Metropolis or Metropolis-Hastings algorithms are often preferred to rejection sampling for difficult cases (and therefore, in a light transport context). We however present a very simple and common example.

Uniformly Sampling a Disk. We aim at sampling a disk of radius R. First, generate a candidate point (x, y) where x and y are independent and uniformly distributed between $-R$ and R. Then, if $x^2 + y^2 \leq 1$ accept the candidate point within $[-R, R]$. Otherwise, reject it and repeat this process. This leads to a uniform distribution over the unit disk. We may also remark that this algorithm trivially provides an approximation of π since its acceptation rate is equal to π when R is equal to 1.

Algorithm 1 Rejection Sampling: Sample x with density f. $\exists M > 1, \forall x \in \Omega, f(x) < M \cdot g(x)$

```
1: while 1 do
2:     Sample $x$ from $g(x)$ and $u$ from $\mathcal{U}(0,1)$
3:     if $u < \frac{f(x)}{M \cdot g(x)}$ then
4:         return x
5:     end if
6: end while
```

2.3.3 Estimators

An estimator is a function of the observable or generated sample data that is used to estimate an unknown parameter from this sample set. An estimator I_N can be therefore defined as a function of several random variables by:

$$I_N = F(X_1 \ldots X_N)$$

Typically, in the remainder of this thesis, the quantity that we want to estimate (generally called *estimand*) will be the intensity of every screen pixel. I finally denotes the quantity we estimate. The estimators can have different properties:

2.3.3.1 Error and Bias

The error of an estimator is defined by:

$$Error(I_N) \;=\; I_N - I$$

The mean of the error is generally known as the bias β of the estimator and it is therefore defined by:

$$\beta[I_N] \;=\; E[I_N - I]$$

The estimator is unbiased if and only if $\forall N \in \mathbb{N}^*,\ \beta[I_N] \;=\; 0$

2.3.3.2 Consistency

The estimator is called consistent if and only if I_N almost surely converges towards I when $N \to \infty$. Therefore, I_N is consistent if and only if:

$$P_r\left(\lim_{N\to\infty} I_N = I\right) \;=\; 1$$

2.3.3.3 Asymptotic Normality

An estimator I_N is asymptotically normal if and only if it is consistent and its distribution around I converges in distribution towards a normal distribution with standard deviation shrinking in proportion to $1/\sqrt{N}$. Typically, if X_N are independent and identically distributed random variables, the central limit theorem directly proves the asymptotic normality of I_N.

2.3.3.4 Efficiency

While computing and evaluating estimators, it is often fundamental to decrease variance and maintaining running time small. This trade-off between variance and speed is generally called efficiency and it is defined by:

$$e[I_N] = \frac{1}{V[I_N]T[I_N]}$$

where $T[I_N]$ is the time needed for the evaluation of I_N. Therefore, an estimator is all the more efficient than the variance is small for a fixed computation time.

2.3.3.5 Miscellaneous Remarks

Generally, it is better to only consider *unbiased* estimators. Indeed, the common goal of a Monte-Carlo integrator is to decrease the Root Mean Square error (RMS error) given by:

$$RMS[I_N] \ = \ E[(I - I_N)^2]$$

In the general case (i.e. with a biased estimator), the RMS error can be expressed with the variance and the bias of the estimator. Indeed, we can easily show that:

$$RMS[I_N] \ = \ V[I_N] \ + \ \beta[I_N]^2$$

Therefore, if the estimator is unbiased, the RMS error is equal to the variance. For this particular case, the central limit theorem generally provides satisfactory error estimates and useful information concerning the speed of convergence if the random variables X_i are independent and identically distributed. However, there are many cases where unbiased estimators are worst than biased ones. First, it is often possible that the most efficient unbiased estimator is less efficient than one given biased estimator: Depending on the application, it may be preferable to use biased estimators. This is for example true in the computer graphics field. While using virtual point lights, as the variance of the resulting estimator does not even exist and is infinite, the resulting estimates have very poor properties. The common solution is to use a very efficient biased estimator and to evaluate the bias with a second unbiased estimator (see Section 4.4.2).

2.3.4 Variance Reduction

As specified by the central limit theorem, decreasing the variance of the computed estimators is fundamental: this actually increases the precision of the estimations that can be obtained for a fixed number of iterations. As there are a lot of variance reduction techniques and as it is a very active research field in computational statistics, we only present here the most famous and classical ones.

2.3.4.1 The Use of Expected Value

A first effective variance reduction technique is the use of expected values to reduce the dimensionality of the integral. Suppose the problem consists in computing the following integral:

$$I = \int_{\Omega_0} \int_{\Omega_1} f(x_0, x_1) d\mu(x_0) d\mu(x_1)$$

If the distribution of the random variable (X_0, X_1) is $p(x_0, x_1)$, the strong law of large numbers provides the following unbiased estimator:

$$I_N = \sum_{i=1}^{N} \frac{f(x_0, x_1)}{p(x_0, x_1)}$$

If $f_0(x_0) = \int_{\Omega_0} f(x_0, x_1) d\mu(x_1)$ and the marginal density of (X_0, X_1) $p_0(x_0) = \int_{\Omega_0} f(x_0, x_1) d\mu(x_1)$ are known, then, we can compute a new estimator J_N such that:

$$J_N = \sum_{i=0}^{N} \frac{f_0(x_0)}{p_0(x_0)}$$

It can actually be shown that J_N has a lower variance than I_N does. Indeed, since the dependence on X_1 is substituted by its expected value, the dimensionality of the integration problem is reduced and the integration problem becomes easier. However, reducing the number of dimensions is not always an appropriate solution, On the contrary, it is often practical to sample higher-dimensional random variables and use marginal densities to project the generated sample on the appropriate sub-space. In this thesis, we present such an example of application in Chapter 8.

2.3.4.2 Importance Sampling

Variance reduction techniques using importance sampling consists in using specific probability density functions for a given integrand. In fact, it is well known that the best probability function to integrate the integrand f is proportional to

f. Indeed, if X is a random variable with pdf $p(x) = \frac{f(x)}{c} = \frac{f(x)}{\int_\Omega f(x)d\mu(x)}$, then:

$$V[I] = E\left[\frac{f(X)}{p(X)}\right]^2 - E\left[\frac{f^2(X)}{p^2(X)}\right] = c^2 - c^2 = 0$$

Unfortunately, it is often impossible to generate such a random variable to compute the integral since the value of the integral has to be known. A common way to reduce the variance of the estimators is therefore to a sample random variable with probability functions g close to the integrand and to make $f(x)/g(x)$ as constant as possible. Many techniques can be used to generate such probability density functions. A common technique consists in discretizing the sample space and in generating a piecewise constant function close to $f(x)$. Other appropriate choices of probability density functions that can ensure efficient importance sampling can be found in [60].

We may finally notice that importance sampling is certainly the most used variance reduction techniques in Monte-Carlo rendering. It is for example, commonly used to sample environment maps or to deal with specular materials or area light sources.

2.3.4.3 Multiple Importance Sampling

Multiple Importance Sampling was introduced by Veach and Guibas in [87]. Their goal was to optimally combine several importance sampling strategies in order to get performance similar to those obtained with the best strategy. To achieve this goal, they propose effective ways to make strategies that compute weighted combinations of all generated estimators. Let $w = (w_i(x))_{i\in[1\ldots N]}$ be a family of real numbers such that $\forall i \in [1\ldots N]$, $\sum_{i=1}^{N} w_i(x) = 1$ and $w_i(x) \geq 0$ and let $I_{N,w}$ denote the following estimator:

$$I_{N,w} = \sum_{i=1}^{N} \frac{1}{N_i} \sum_{j=1}^{N_j} w_i(X_{i,j}) \frac{f(X_{i,j})}{p_i(X_{i,j})}$$

where $X_{i,j}$ are independent and identically distributed random variables with probability density functions p_i.

The Balance Heuristic. The balance heuristic is given by the following weighting functions:

$$w_i(x) = \frac{N_i p_i(x)}{\sum_{j=1}^{N_i} N_j p_j(x)} \qquad (2.1)$$

The authors proved that for all estimators I_N built with a given family $(w_i(x))_{i\in[1\dots N]}$, we have the following relation:

$$V[\hat{I}_N] \leq V[I_N] + \left(\frac{1}{\min_i n_i} - \frac{1}{\sum_i n_i}\right) \cdot I^2$$

where $\hat{I}_N$ is the estimator obtained by using weights proposed by Equation (2.1) and I is the quantity we want to estimate. This inequality actually tends to prove that, even if there is still room for improvement, the balance heuristics gives good results.

The Power Heuristic. The power heuristic is given by the following weighting functions:

$$w_i(x) \ = \ \frac{N_i^\beta p_i(x)^\beta}{\left(\sum_{j=1}^{N_i} N_j p_j(x)\right)^\beta}$$

An exponent $\beta = 2$ gives satisfactory results in practice. The exponentiation actually increases the weight of the samples for which one of the sampling techniques which have been used generates it with a high probability.

2.3.4.4 Control Variates

The control variate technique proposes to find a function g similar to the integrand and assumes that the integral $\int_\Omega g(x)d\mu(x)$ can be analytically computed. The technique then consists in computing a new estimator $\hat{I}$ by subtract g such that:

$$\hat{I} \ = \ \int_\Omega g(x)d\mu(x) + \sum_{i=1}^{N} \frac{f(X_i) - g(X_i)}{p(X_i)}$$

where X_i are independent and identically distributed random variables with probability density function p. Then, if we can find an appropriate density function p, we can ensure that the variance of the estimator is decreased.

In a sense, control variates are the orthogonal approaches of importance sampling techniques: The first ones try to minimize $f(x) - g(x)$ while the second ones aim at making $f(x)/g(x)$ as close to 1 as possible.

2.3.4.5 Stratified Sampling

Stratified sampling proposes to divide the domain Ω into different non-overlapping sub-domains (generally called strata) $\Omega_1 \dots \Omega_N$ such that:

$$\bigcup_{i=1}^{N} \Omega_i = \Omega$$

We then take n_i independent and identically distributed samples with probability density function p_i within each stratum. This gives us a new stratified estimator given by:

$$I_N^{(s)} = \sum_{i=1}^{N} \mathrm{P}(\Omega_i) \frac{1}{n_i} \sum_{j=1}^{n_i} f(X_{i,j})$$

For example, if $\Omega = [0,1]$, a typical stratified uniform sampler will choose Ω_i and n_i such that:

$$n_i = 1 \text{ and } \Omega_i = \left[\frac{i-1}{N}, \frac{i}{N}\right]$$

The resulting variance and convergence properties of stratified sampling have been studied in the literature. We may however make three important remarks:

- Stratified sampling cannot be worse than uniform sampling: it is therefore always better to use it if it is possible;
- It is better to increase the number of strata rather than to increase the number of samples per stratum. A good choice is therefore to use only one sample per stratum (i.e. $N = \sum_i n_i$ and $n_i = 1$);
- If the integrand is smooth and there are few discontinuities within each stratum, the variance can decrease on the order of $\mathcal{O}(N^{-2})$ i.e. as the square of the number of strata.

A major problem with stratified sampling is the difficulty to compute strata when the integration problem becomes high-dimensional. For example, for a d-dimensional problem, splitting once each dimension results in 2^d strata thereby limiting the choice of the sample count and actually the possibility to use only a small number of samples (which is relatively important in computer graphics and Monte-Carlo rendering). Several solutions such as Quasi Monte-Carlo methods however try to solve this problem. For more details about Quasi Monte-Carlo techniques, one may refer to Niederreiter [58], Beck and Chen [3] or Kuipers and Niederreiter [52]

2.3.4.6 Russian Roulette

Russian roulette does not actually reduce the variance of a given estimator but rather tends to increase its efficiency. It is actually a very commonly used technique in the field of Monte-Carlo rendering. It first consists of considering a first estimator I_N such that:

$$I_N = \sum_{i=1}^{N} J_i, \text{ where } J_i \text{ are a set of estimators.}$$

In an application like a light transport algorithm, it is probable that J_i is very small and that its contribution is negligible. Therefore, its evaluation will be very expensive and will make the estimator *inefficient*. The idea of Russian roulette is to try to neglect this contribution in an *unbiased* way. To achieve such a result, the Russian roulette technique proposes to build a new estimator $I_N^{(r)}$ such that:

$$I_N^{(r)} = \sum_{i=1}^{N} J_i^{(r)} \text{ where}$$

$$\forall j \in \mathbb{N}_N,\ J_i^{(r)} = \frac{B_i}{p_i} \cdot J_i$$

B_i is a Bernoulli random variable with probability $p_i \in [0,1]$ such that $B_i = p_i \cdot \delta_0 + (1-p_i) \cdot \delta_1$. Even if the new estimator $I_N^{(r)}$ has a higher variance than I_N does, it is possible to make it more efficient if we have some information about J_i. It is actually a very common situation in Monte-Carlo rendering. Suppose that we have N light sources: if we consider one geometric point on the surface of the scene, it is possible that only a small fraction of the light sources is able to bring a large contribution to this point *even if they are unoccluded*. Russian roulettes can therefore be applied to randomly discard light sources with small unoccluded contributions: the idea is to limit the expensive visibility computations by evaluating the energy transfer between one light and the point as if there were no occluder and randomly set the contribution to 0 if it is small. This leads to the following value for p_i:

$$p_i = \min(1, C_i/s)$$

where s is a user-defined threshold and C_i is the unoccluded contribution of light source i such that:

$$C_i = \begin{cases} J_i \text{ if there is no occluder} \\ 0 \text{ otherwise} \end{cases}$$

This technique can easily increase the efficiency of $I_N^{(r)}$.

2.4 Conclusion

The roots of Monte-Carlo rendering are the strong law of large numbers and the central limit theorem. As the convergence rate cannot be changed, decreasing the variance of the computed estimators or increasing their efficiency is therefore the core of computational statistics. In this thesis, we will try to focus on both strategies. We indeed want to combine low variance estimators with efficient implementations by proposing predictive and robust methods which can efficiently use current hardwares.

CHAPTER 3

Physics of Light Transport

In this chapter, we present the physics of light transport: we therefore explain how lights can interact with matter and most of the corresponding equations involved in Monte-Carlo rendering and more generally in realistic rendering.

3.1 Introduction

Simulating most of light transport phenomena requires a well defined formal approach. As other simulation problems, it can be summarized into three different points:

- We first have to identify the *physical* problem;
- Then, we must formalize a *mathematical* model to support this problem;
- Finally, we have to propose an algorithmic and numerical solution to our formalization.

In this chapter, we focus on the first two points. The third point will be analyzed in the next chapter where we make a state of the art of most of the Monte-Carlo methods proposed to solve the physical problem we describe here. Actually, the main part of this Ph.D. and of our contributions focuses on the third point i.e. the computational part of the light transport problem.

To be more precise, a model which completely satisfies the light transport problem will include the following four aspects:

- The description of the scene we have to render. This includes the geometry of the objects, the specification of the media (for example, taking into account atmospheric phenomena) and the specification of the light sources;
- The radiometric measures;
- The light transport model that specifies how the light will propagate in the scene and how it behaves on the surfaces of the objects;

- The quantities we want to measure. Typically, when we compute a virtual picture, these quantities are generally the responses (i.e. the transmitted colors) of each camera captor.

We describe here these four points. One may also refer to the excellent and very complete description in the Ph.D. dissertations of Eric Veach [85], Eric Lafortune [54], James Arvo [1] or Philip Dutré [20]. As Eric Veach did in his Ph.D. thesis, we finally think it is important to clearly formalize the problem by identifying the associated integration domains, measures and integrands. As we explained it in the previous section, it is a major concern when we want to use any Monte-Carlo strategies. Furthermore, sorting the rendering algorithms depending on the integration spaces, the computed integrands and so on is a great help when we will present most of the Monte-Carlo rendering techniques used in computer graphics in Section 4.

3.2 Geometric Quantities

3.2.1 Surfaces of the Scene

We first assume that the geometry of the scene consists of a finite set $\mathcal{M}$ of piecewise differentiable two-dimensional manifold surfaces in $\mathbb{R}^3$. As differentiable surfaces, they more particularly contains some extra information such as normal vectors and intrinsic parameterization (i.e. a mapping from the unit square to the surface that is used for example for texture mapping). A point on a surface is therefore given by a family of vectors such as its position, normal, tangent and binormals vectors.

The surfaces divide $\mathbb{R}^3$ into several connected cells: each of them is filled with a non-participating medium with a constant refractive index. We finally define an area measure A on $\mathcal{M}$ such that A is the Lebesgues (or uniform) area measure on $\mathcal{M}$.

3.2.2 Directions and Solid Angles

Directions are represented by unit-length vectors $\omega \in \mathbb{R}^3$. The set of all directions is denoted as the unit sphere S^2 in $\mathbb{R}^3$. The measure commonly associated to S^2 is the solid angle measure denoted by σ such that if $\Lambda \in S^2$ is a measurable subset of S^2, $\sigma(\Lambda)$ is the solid angle of Λ.

Another interesting measure for integration over directions is the projected solid angle $\sigma^{\perp}$ defined such that:

$$\text{if } \Lambda \text{ is a measurable subset of } S^2,\ \sigma^{\perp}(\Lambda) = \int_{\Lambda} \omega \cdot \mathbf{N}(\mathbf{x}) d\omega$$

where $\mathbf{x}$ is a point on $\mathcal{M}$ and $\mathbf{N}(\mathbf{x})$ is the normal vector at $\mathbf{x}$. The dot product $\omega \cdot \mathbf{N}(\mathbf{x})$ is generally denoted $cos(\theta)$ where θ is angle $\widehat{\mathbf{N}(\mathbf{x}), \omega}$. Actually, if we consider only one hemisphere (the one such as $cos(\theta) \geq 0$ or the one such as $cos(\theta) < 0$) [1], the projected solid angle of Λ is the area of the projection of Λ on the plane sustaining the chosen hemisphere. Figure 3.1 illustrates most of the notions described here. In the rest of the thesis, $\mathcal{H}^{+}(\mathbf{x})$ will denote the set of

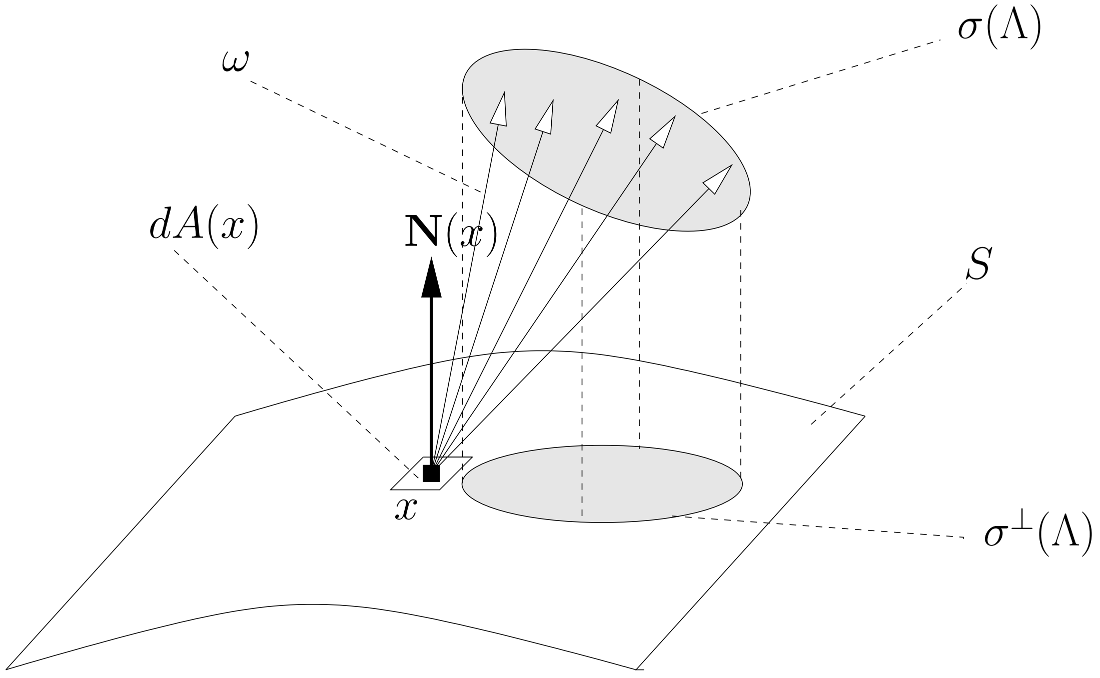


Figure 3.1: Surface S, Directions ω, Solid Angles $\sigma(\Lambda)$ and Projected Solid Angles $\sigma^{\perp}(\Lambda)$

directions ω such as $\omega \cdot \mathbf{N}(\mathbf{x}) \geq 0$ and $\mathcal{H}^{-}(\mathbf{x})$, the set of directions ω such as $\omega \cdot \mathbf{N}(\mathbf{x}) < 0$. Finally, $\overrightarrow{\mathbf{xy}}$ will denote the direction vector going from $\mathbf{x}$ to $\mathbf{y}$ and $\overrightarrow{\mathbf{xy}}^{(u)}$, the corresponding unit-length direction.

[1]In computer graphics, we often consider the upward hemisphere of a given set of direction (i.e. the hemisphere such that $cos(\theta) \geq 0$) for non-transparent objects.

3.3 Radiometric Quantities

The basic notions for the light transport problem are its intrinsic radiometric quantities. We present here the most important and commonly used ones.

3.3.1 Power or Flux

The power (also called "radiant power" or "flux "in the field of radiometrics) is the rate of flow of electromagnetic energy (or "radiant energy"). Radiant power is usually expressed in watts or joules per second $[W = J \cdot s^{-1}]$. Therefore, if W is the radiant energy, the radiant power ϕ is defined by:

$$\phi = \frac{dW}{dt}$$

3.3.2 Irradiance

Irradiance $E(x)$ at point $\mathbf{x}$ is the power per unit surface area at $\mathbf{x}$ and is therefore defined by:

$$E(\mathbf{x}) \;=\; \frac{d\phi(\mathbf{x})}{dA(\mathbf{x})}$$

with units $[W \cdot m^{-2}]$.

3.3.3 Radiance

Radiance is the most important radiometric quantity for the light transport problem since it is directly related to the rendering equation (see Section 3.5.1 for more details). Radiance $L(\mathbf{x}, \omega)$ at point $\mathbf{x}$ and for direction ω is the power per *projected* solid angle per unit area. It is therefore defined by:

$$L(\mathbf{x}, \omega) \;=\; \frac{d^2\phi(\mathbf{x}, \omega)}{dA(\mathbf{x})\ d\sigma^{\perp}(\omega)}$$

with units $[W \cdot m^{-2} \cdot sr^{-1}]$. Thus, if $\mathbf{N}(\mathbf{x})$ is the normal at point $\mathbf{x}$, then $L(\mathbf{x}, \omega)$ can be expressed as:

$$L(\mathbf{x}, \omega) \;=\; \frac{d^2\phi(\mathbf{x}, \omega)}{|\mathbf{N}(\mathbf{x}) \cdot \omega)| dA(\mathbf{x})\ d\sigma(\omega)}$$

Quantity	Symbol	Definition	Unit
Power	ϕ	$\frac{dW}{dt}$	$[W]$
Irradiance	E	$\frac{d\phi(\mathbf{x})}{dA(\mathbf{x})}$	$[W \cdot m^{-2}]$
Radiance	L	$\frac{d^2\phi(\mathbf{x},\omega)}{dA(\mathbf{x})\ d\sigma^{\perp}(\omega)}$	$[W \cdot m^{-2} \cdot sr^{-1}]$
Spectral Radiance	L_λ	$\frac{d^3\phi(\mathbf{x},\omega,\lambda)}{dA(\mathbf{x})\ d\sigma^{\perp}(\omega) d\lambda}$	$[W \cdot m^{-2} \cdot sr^{-1} \cdot m^{-1}]$

Table 3.1: Radiometric Quantities Commonly used in Computer Graphics

Therefore, if $A^{\perp}$ is the *projected* area measure, $L(\mathbf{x}, \omega)$ can be expressed by:

$$L(\mathbf{x}, \omega) \ = \ \frac{d^2\phi(\mathbf{x}, \omega)}{dA^{\perp}(\mathbf{x})\ d\sigma(\omega)}$$

The radiance function intuitively corresponds to the quantity of power passing through a small surface $dA^{\perp}(\mathbf{x})$ perpendicular to ω around ω in a small solid angle $d\sigma(\omega)$.

3.3.4 Spectral Radiance

Spectral radiance L_λ is finally defined as the radiance per wavelength and is therefore given by:

$$L_\lambda(\mathbf{x}, \omega, \lambda) = \frac{d^3\phi(\mathbf{x}, \omega, \lambda)}{dA(\mathbf{x})\ d\sigma^{\perp}(\omega)\ d\lambda}$$

with units $[W \cdot m^{-2} \cdot sr^{-1} \cdot m^{-1}]$

Spectral radiance is also a major concept even if the different wavelengths are often not explicitly distinguished. We generally consider the quantities as vectors L with a number of components dependent on the wavelength such that:

$$L = [L_{\lambda_1} \dots L_{\lambda_k}]$$

In most light transport solvers (i.e. renderers), only three components (red, green, and blue) are used. In this Ph.D. thesis, all algorithms will also only deal with these three components.

Table 3.1 finally summarizes all the quantities presented here.

3.4 Material Properties

Material properties describe the behavior, the response and the interaction of a given surface with light. It thus defines how light is reflected, refracted or emitted by a surface. In computer graphics, the most commonly used function for non-emitting surfaces is the bidirectional scattering distribution function or BSDF. For light sources, we generally use the emission distribution function (EDF) which provides the power emitted for each small surface area and each solid angle over the surface of the light source.

3.4.1 BSDF

For surfaces, it is generally interesting to dissociate incident and exitant radiance functions. The incident radiance function $L_i(\mathbf{x}, \omega)$ will give the quantity of power *arriving* at $\mathbf{x}$ per unit area and per projected solid angle at $\mathbf{x}$ around ω_i. Conversely, the exitant radiance function $L_o(\mathbf{x}, \omega_o)$ will give the quantity of radiance leaving $\mathbf{x}$. This physical distinction directly leads to the definition of the BSDF which describes the complex relation between L_o and L_i. BSDF $f_s(\mathbf{x}, \omega_o \to \omega_i)$ is actually expressed as the ratio of the radiance leaving in direction ω_o and the unit irradiance arriving from ω_i at point $\mathbf{x}$. Therefore:

$$f_s(\mathbf{x}, \omega_o \to \omega_i) = \frac{dL_o(\mathbf{x}, \omega_o)}{dE_i(\omega_i)} = \frac{dL_o(\mathbf{x}, \omega_o)}{dL_i(\mathbf{x}, \omega_i) d\sigma^{\perp}(\omega_i)}$$

3.4.2 BRDF and BTDF

Generally, the BSDF is decomposed into several components according to the considered hemisphere. Actually, the BRDF f_r is the restriction of the BSDF to the incoming and outgoing direction subset such that the incoming and outgoing directions respectively belong to the same hemisphere around the surface normal $\mathbf{N}(\mathbf{x})$. Conversely, the BTDF f_t is the restriction of the BSDF to the direction subset such that the incoming and outgoing directions respectively belong to two opposite hemispheres.

In computer graphics, we also generally classify the BSDFs according to the directionality of the light scattering. Therefore, three behaviors, diffuse, specular and glossy are informally specified:

Diffuse Reflection and Diffuse Transmission. The BSDF is constant and the light is therefore directly scattered proportionally to the projected solid angle.

Specular Reflection and Specular Transmission. The BSDF is equal to a Dirac or to a function close to a Dirac (this is the informal part of this classification).

Glossy Reflection and Glossy Transmission. This contains all other materials. This can describe, for example, a moderately diffuse plastic or painting.

The BRDF has finally a set of physical properties which make it physically plausible.

Helmotz Reciprocity principle. If a given surface is physically based, then:

$$f(\mathbf{x}, \omega_i \rightarrow \omega_o) \;=\; f(\mathbf{x}, \omega_o \rightarrow \omega_i)$$

Energy Conservation The directional hemispherical reflectance or albedo ρ_r is the fraction of the incoming radiance for a single direction that is reflected over the hemisphere. It is therefore defined by:

$$\rho_r(\mathbf{x}, \omega_i) \;=\; \int_{\mathcal{H}^+(\mathbf{x})} f_r(\mathbf{x}, \omega_o \rightarrow \omega_i)\; d\sigma^{\perp}(\omega_o)$$

A physically based BRDF must ensure that the albedo function $\rho_r(\mathbf{x}, \omega_i)$ satisfies:

$$\forall \omega_i \in \mathcal{H}^+(\mathbf{x}), \rho_r(\mathbf{x}, \omega_\mathbf{i}) \leq 1$$

3.4.3 Light Sources

While most of the properties presented above deal with passive material i.e. materials which do not emit light, another class of materials, the light sources, are materials which do emit light. They are generally specified by their emission distribution function (EDS) which defines the self emitted radiance $L_e(\mathbf{x}, \omega)$ at point $\mathbf{x}$ on the light source surface for direction $\omega \in \mathcal{H}^+$. Generally, two other quantities are often considered. The emittance $B_e(\mathbf{x})$ is defined as the self-emitted radiance integrated over $\mathcal{H}^+$ and is therefore given by:

$$B_e(\mathbf{x}) \;=\; \int_{\mathcal{H}^+} L_e(\mathbf{x}, \omega_o) d\sigma^{\perp}(\omega_o)$$

The self-emitted flux or power ϕ_e of the light source is defined as the self-emitted emittance integrated over the surface A of the light source and is thus given by:

$$\phi_e = \int_A B_e(\mathbf{x}) dA(\mathbf{x})$$

3.5 Light Transport Equations

While dealing with light transport problems, the first issue is to physically describe the behavior of the materials and the lights. This is the role of the *rendering equation*. The second issue is to describe how the measurements are done during the virtual experiments. This is the role of the *measurement equation*. Typically, the rendering equation will say how a given surface will scatter the incoming light flux while the measurement equation will provide the response of the camera film to the scattered light.

3.5.1 The Rendering Equation

The rendering equation or the radiance transfer equation is the most fundamental relation in the field of light transport. It actually combines into the following unique integral equation all the concepts and quantities previously described: the BSDF, the self-emitted radiance and the associated geometric quantities.

$$\forall(x, \omega_o) \in (\mathcal{M}, S^2),\ L_o(\mathbf{x}, \omega_o) = L_e(\mathbf{x}, \omega_o) + \int_{S^2} f_s(\mathbf{x}, \omega_o \rightarrow \omega_i)\ L_i(\mathbf{x}, \omega_i) d\sigma^{\perp}(\omega_i) \tag{3.1}$$

Therefore, the outgoing radiance at point $\mathbf{x}$ into direction ω is equal to the sum of the self-emitted radiance (equal to zero it the surface does not emit light) and the incoming radiance "filtered" by the BSDF and integrated over the complete direction unit sphere S^2.

The first form of the rendering equation given by Kajiya in [42] is slightly different but the form given in 3.1 is a bit more practical and the most commonly used today.

Several important remarks concerning the rendering equation have to be made.

3.5.1.1 The Rendering Equation as a Fredholm Equation of the Second Kind

We can reexpress $L_i(\mathbf{x}, \omega_i)$ in a more practical way. If $\mathbf{y} = \mathrm{vp}(\mathbf{x}, \omega_i) \in \mathcal{M}$ denotes the first intersection point seen by $\mathbf{x}$ into direction $-\omega_i$, then $L_i(\mathbf{x}, \omega_i) = L_o(\mathbf{y}, -\omega_i)$. Therefore, the rendering equation can be reexpressed in a recursive form:

$$L_o(\mathbf{x}, \omega_o) \;=\; L_e(\mathbf{x}, \omega_o) + \int f_s(\mathbf{x}, \omega_o \to \omega_i)\; L_o\left(\mathrm{vp}(\mathbf{x}, \omega_i), -\omega\right) d\sigma^{\perp}(\omega_i) \qquad (3.2)$$

This form is very close to a Fredholm equation of the second kind generally defined by: $f(x) = g(x) + \int_0^1 f(y)K(x,y)dy$. As we show it in the next chapters, it directly leads to specific numerical Monte-Carlo techniques using Markovian paths in $\mathcal{M}^{\mathbb{N}}$.

3.5.1.2 Extended Forms of the Rendering Equation

The rendering equation in its original form does not handle all different lighting phenomena. Several extensions have been therefore proposed to handle phenomena such as sub-surface scattering and participating media [19, 70], polarization [84], diffraction [77] and fluorescence [26].

3.5.2 The Measurement Equation

Whereas the rendering equation models the behavior of elements in a scene, the measurement equation models the response of real or virtual captors. For captor j, we first define the sensor responsivity $W^{(j)}(\mathbf{x}, \omega)$. The total response I_j is therefore expressed as the integration of $W^{(j)}(\mathbf{x}, \omega) \cdot L_i(\mathbf{x}, \omega)$ such that:

$$I_j \;=\; \int_{\mathcal{M}\times S^2} W^{(j)}(\mathbf{x}, \omega) L_i(\mathbf{x}, \omega) d\sigma^{\perp}(\omega)$$

Typically, if the surface points of sensor j are included into $\mathcal{M}^{(j)}$, $W^{(j)}(\mathbf{x}, \omega)$ can be classically decomposed into two components such that:

$$W(\mathbf{x}, \omega) \;=\; \mathbb{I}_{\mathcal{M}^{(j)}}(\mathbf{x}) \cdot W'^{(j)}(\mathbf{x}, \omega) \qquad (3.3)$$

where

$$\mathbb{I}_{\mathcal{M}^{(j)}}(\mathbf{x}) = \begin{cases} 1 \text{ if } \mathbf{x} \in \mathcal{M}^{(j)} \\ 0 \text{ otherwise} \end{cases}$$

3.6 Conclusion

In this part, we presented the most important physical quantities while dealing with light transport problems and the two most major equations. This gives us the goal of any renderer: computing the measurement equation for all sensors in the scene by using the rendering equation. In the next part, we present simple manners to make this problem suitable to Monte-Carlo techniques. Different classical Monte-Carlo solvers and our own techniques will be then exposed.

CHAPTER 4

Monte-Carlo Rendering

While the two previous parts introduced the physics of light transport (the *model* we want to study, i.e. the rendering and the measurement equations) and Monte-Carlo methods (the theoretical framework which helps us to design new *numerical schemes*), this chapter presents most of the algorithms which aim at computing virtual pictures by using this physical model and Monte-Carlo estimators. This class of methods are commonly named "Monte-Carlo Rendering" techniques.

4.1 An Appropriate Formalism for Monte-Carlo Rendering

The goal of any Monte-Carlo integrator is to compute an integral of the form:

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

Unfortunately, the rendering equation (see Section 3.5.1) does not have such a practical form. To make our problem suitable to Monte-Carlo methods, we therefore have to specify one sample space, the associated measure and the integrand. In his Ph.D. thesis, Veach proposes such a formalism [85], the path integral formulation, that we therefore expose in this section. However, for many sampling techniques used in the field of Monte-Carlo rendering, the path integral formulation is not the most appropriate one since many samplers specifically deal with the path length: as specifying only one unified path space is not necessarily practical, we also introduce in this chapter the formalism commonly needed while dealing with many Monte-Carlo rendering techniques.

4.1.1 The Path Integral Formulation

The path integral formulation aims at unifying the sampling processes while using Monte-Carlo rendering techniques. The goal is to make the light transport problem directly suitable to a Monte-Carlo approach which consists in sampling a random variable X with density function p and in evaluating $f(X)/p(X)$.

4.1.1.1 The Light Transport Equation

Veach first developed the "three point form" formulation which describes the local lighting behavior of materials. This formulation replaces the classical rendering equation form by removing all occurrences of directions ω. First, we redefine the radiance function (for either its outgoing or incoming form) such that:

$$L(\mathbf{x}, \omega) = L(\mathbf{x} \to \mathbf{x}')$$

where ω is the unit-vector pointing out from $\mathbf{x}$ to $\mathbf{x}'$. In the same way, we can redefine the BSDF f_s such that:

$$f_s(\mathbf{x}', \omega_o \to \omega_i) \;\; = \;\; f_s(\mathbf{x} {\to} \mathbf{x}' {\to} \mathbf{x}'')$$

where $\omega_i = \overrightarrow{\mathbf{x}\mathbf{x}'}^{(u)}$ and $\omega_o = \overrightarrow{\mathbf{x}'\mathbf{x}''}^{(u)}$. Finally, we replace measure $\sigma^\perp$ by measure A with a change of variables in the rendering equation 3.1 such that:

$$d\sigma^\perp(\omega_i) = d\sigma^\perp(\overrightarrow{\mathbf{x}\mathbf{x}'}^{(u)}) = G(\mathbf{x} \leftrightarrow \mathbf{x}')dA(\mathbf{x})$$

$G(\mathbf{x} \leftrightarrow \mathbf{x}')$ is the geometric term between $\mathbf{x}$ and $\mathbf{x}'$. It represents the differential beam between the two differential surfaces and is given by:

$$G(\mathbf{x} \leftrightarrow \mathbf{x}') = V(\mathbf{x} \leftrightarrow \mathbf{x}') \frac{\cos(\theta_0)\cos(\theta_i')}{||\mathbf{x} - \mathbf{x}'||^2}$$

where $V(x \leftrightarrow x')$ is the visibility term between $\mathbf{x}$ and $\mathbf{x}'$ which is equal to 1 if $\mathbf{x}$ sees $\mathbf{x}'$ and 0 otherwise. θ_0 (resp. θ_i') is the angle between $\mathbf{x} \to \mathbf{x}'$ and the surface normal at $\mathbf{x}$ (resp. $\mathbf{x}'$). By respectively replacing $d\sigma^\perp(\omega_i)$, $f_s(\mathbf{x}', \omega_o \to \omega_i)$ and $L(\mathbf{x}, \omega_i)$ by their respective three-point form expressions described above, we obtain the three point form of the rendering equation:

$$L(\mathbf{x}' {\to} \mathbf{x}'') \;\; = \;\; L_e(\mathbf{x}' {\to} \mathbf{x}'') + \int_{\mathcal{M}} L(\mathbf{x} {\to} \mathbf{x}') f_s(\mathbf{x} {\to} \mathbf{x}' {\to} \mathbf{x}'') G(\mathbf{x} {\leftrightarrow} \mathbf{x}') dA(\mathbf{x}) \quad (4.1)$$

4.1.1.2 The Measurement Equation

Similarly, we can reexpress the measurement equation given by Equation 3.3 by replacing the radiance expression and the sensor responsivity by their three-point forms. All sensor responses can be thus defined by:

$$I_j = \int_{\mathcal{M} \times \mathcal{M}} W_e^{(j)}(\mathbf{x} {\to} \mathbf{x}') L(\mathbf{x} {\to} \mathbf{x}') \; G(\mathbf{x} {\leftrightarrow} \mathbf{x}') \; dA(\mathbf{x}) \; dA(\mathbf{x}') \quad (4.2)$$

4.1.1.3 The Path Integral Formulation

Using the light transport equation, the measurement equation can be recursively expanded to be expressed in an iterative way:

$$\begin{aligned} I_j \;=\; & \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} \Big[L_e(\mathbf{x_k} \rightarrow \mathbf{x_{k-1}}) G(\mathbf{x_0} \leftrightarrow \mathbf{x_1})\; W_e^{(j)}(\mathbf{x_1} \rightarrow \mathbf{x_0}) \\ & (\prod_{i=1}^{k-1} f_s(\mathbf{x_{i+1}} \rightarrow \mathbf{x_i} \rightarrow \mathbf{x_{i-1}}) G(\mathbf{x_i} \leftrightarrow \mathbf{x_{i+1}})) dA(\mathbf{x_0}) \;\ldots\; dA(\mathbf{x_k}) \Big] \end{aligned} \tag{4.3}$$

The measurement equation can be finally reformulated as:

$$I_j = \int_{\Omega} f^{(j)}(\overline{x}) d\mu(\overline{x}) \tag{4.4}$$

$f^{(j)}$ is defined for each path length k by extracting the appropriate term from expansion (4.3), Ω is the set of all finite length paths and μ the natural associated measure given by: $\mu(D) = \sum_{k=1}^{\infty} \mu_k(D \cap \Omega_k)$ where Ω_k is the set of all length k paths and μ_k the associated product measure given by $d\mu_k(\mathbf{x_0} \ldots \mathbf{x_k}) = dA(\mathbf{x_0}) \ldots dA(\mathbf{x_k})$. With this formalism, the global illumination problem is now an integration problem we can solve by a Monte-Carlo algorithm.

As indicated by Veach, using the path integral formulation has several advantages.

- As this is a standard formulation for Monte-Carlo techniques, it becomes much easier to use and re-use most of the variance reduction methods described in the literature;
- This formulation allows us to directly sample paths of any length. As it is not needed anymore to make some distinction about path length, samplers can become much simpler and generic;
- By explicitly dealing with light paths, we can design more efficient samplers. Indeed, as we will show in the next sections, until Veach introduced this formulation (and the associated Metropolis Light Transport algorithm [88]), most of the algorithms only dealt with local sampling strategies: it was therefore difficult to find optimal random variables, i.e. random variables with distributions proportional to $f^{(j)}$.

Finally, building Monte-Carlo estimators with such a formalism is straightforward. First, sample a path $\overline{X}$ with density $p(\overline{X})$. Then, simply consider the estimator $f^{(j)}(\overline{X})/p(\overline{X})$. If we can ensure sufficient conditions on $f^{(j)}(\overline{X})/p(\overline{X})$, this directly leads to an unbiased estimator I_j such that:

$$\mathrm{E}\left[\frac{f^{(j)}(\overline{X})}{p(\overline{X})}\right] = I_j$$

In the next section, we present some simple strategies to sample such light paths $\overline{x} \in \Omega$.

4.1.2 Sampling the Path Space

With Equation 4.4, we have the appropriate formalism to tackle the light transport problem with Monte-Carlo techniques. We present here the basic notions to perform sampling processes with the path integral formulation.

- The first technique consists in sampling points across the set $\mathcal{M}$ of the surfaces of the scene. It is actually a common strategy while we have to sample area light sources;
- The second technique consists in sampling directions and finding the nearest intersection point along this direction. For example, while dealing with glossy surfaces, it is common to sample direction ω from point $\mathbf{x}$ with a distribution close to the BSDF $f_s(\mathbf{x}, \cdot)$.

The first sampler is the most related to the path integral formulation since this formulation explicitly deals with surfaces. For example, *uniformly* sampling a point on surface $\mathcal{M}$ with a total area equal to $A_{\mathcal{M}}$ can simply be achieved with a density equal to $1/A_{\mathcal{M}}$.

The second sampler is a bit more complicated since it requires a change of variables (see Figure 4.1). Actually, if ω is generated from $\mathbf{x}$ with probability $p_{\mathbf{x}}(\omega)$, then $\mathbf{x}'$ along direction ω has a density equal to:

$$p(\mathbf{x}') = p(\omega) \cdot \frac{|\cos(\theta_i')|}{||\mathbf{x} - \mathbf{x}'||^2}$$

With these remarks, it becomes easy to sample point over the set of surfaces $\mathcal{M}$. However, generating a *complete* path and evaluating the corresponding density is a bit more complicated.

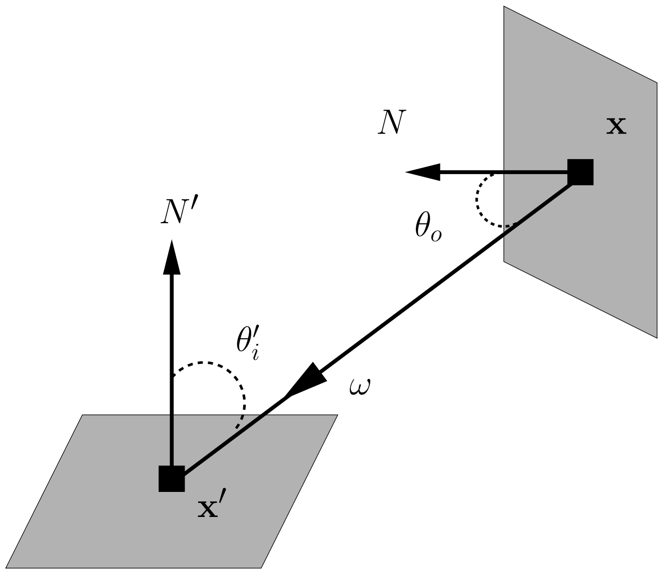


Figure 4.1: **Geometry and Notations to Generate $\mathbf{x'}$ from $\mathbf{x}$.**

4.1.2.1 Randomly Sampling Points over the Surface Set $\mathcal{M}$

The first obvious method would be to select points over the surfaces of the scene in a naive way. We can for example design the following sampler:

- Randomly choose the length l of the path $\overline{x}$ with probability $P(l)$;
- Randomly sample l *independent* points $(\mathbf{x_k})_{k\in[1\ldots l]}$ with respective density $p_k(\mathbf{x_k})$.

This naive sampler provides a path $\overline{x} = (\mathbf{x_k})_{k\in[1\ldots l]}$ with a density equal to:

$$p(\overline{x}) = P(l)\prod_{k=1}^{l} p_k(\mathbf{x_k})$$

To increase the probability that $f(\overline{x}) > 0$, we may ensure that:

- $\mathbf{x_1}$ is on camera sensor j.
- $\mathbf{x_l}$ is on a light source.

Unfortunately, even if we ensure the two above properties, this sampler will provide very poor results. Indeed, for most common scenes, since the probability that segment $(\mathbf{x_i}, \mathbf{x_{i+1}})$ is occluded may be large, the probability that $f(\overline{x}) = 0$ is high and the resulting estimator $f(\overline{X})/p(\overline{X})$ will have a high variance. This leads to another sampling strategies mostly based on truncated "Markov-Chained" light paths.

4.1.2.2 Chaining Sampled Points

In computer graphics (and in other scientific fields such as neutron transport), it is much efficient to choose points $\mathbf{x_k}$ such that:

$$\forall k \in [1 \ldots l-1],\ vp(\mathbf{x_k}, \overrightarrow{\mathbf{x_k x_{k+1}}}^{(u)}) = \mathbf{x_{k+1}}$$

We therefore just have to ensure that $\mathbf{x_{k+1}}$ is the first visible point by $\mathbf{x_k}$ along direction $\overrightarrow{\mathbf{x_k x_{k+1}}}^{(u)}$. To achieve such a result, the common solution is to chain the sampled points $\mathbf{x_k}$. This is actually the core of all "path tracing" technique we present further in this chapter. The idea is to generate $\mathbf{x_{k+1}}$ *conditionally* to $\mathbf{x_k}$. Imagine that we have already sampled point $\mathbf{x_k}$. A practical way to find a point over the surface set $\mathcal{M}$ is to sample direction ω_k with density $p(\omega_k)$, to compute $vp(\mathbf{x_k}, \omega_k)$ and to finally set $\mathbf{x_{k+1}} = vp(\mathbf{x_k}, \omega_k)$. With this sampling technique, we do not actually know the density of points $\mathbf{x_k}$ but only the *conditional* density $p(\mathbf{x_k} \to \mathbf{x_{k+1}})$ given by:

$$p(\mathbf{x_k} \to \mathbf{x_{k+1}}) = p(\omega_k) \cdot \frac{|\cos(\theta_i')|}{||\mathbf{x_k} - \mathbf{x_{k+1}}||^2}$$

Fortunately, we do not need to know the density of $\mathbf{x_k}$ but only the density of the complete path $\overline{x}$. Actually, such a path is simply a truncated Markov-Chain since point $\mathbf{x_{k+1}}$ only depends on point $\mathbf{x_k}$. Therefore, the density of $\overline{x}$ can be expressed by:

$$p(\overline{x}) = p(x_1) \cdot \prod_{k=1}^{l-1} p(\mathbf{x_k} \to \mathbf{x_{k+1}})$$

Sampling such paths is the most common technique in computer graphics. We give here two examples of samplers directly using both path integral formulation and Markov-Chain path sampling strategies. One may argue that the following techniques are *not* used in the field of computer graphics. It is actually true but our goal is to show how to use the path integral formulation in a direct manner. Actually, the only algorithms which require and use it are those which are based on Metropolis Light Transport samplers [88]. For all other ones, the path integral formulation is used in a more restrictive way. This common approach will be presented and introduced in the next section.

4.1.3 A "Path Integral Formulated" (but *Inefficient*) Path Tracer

We consider in this section the sensor j and as precised in Section 4.1.1.3, we therefore want to compute:

$$I_j = \int_\Omega f^{(j)}(\overline{x}) d\mu(\overline{x})$$

$\mathcal{M}_s$ will denote the light source surfaces in the scene. Our path tracer will sample paths $\overline{x}$ as presented by Algorithm 2.

Algorithm 2 "Path Integral Formulated" Path Tracer: sample path $\overline{x}$

```
 1: Sample point $\mathbf{x_1}$ over the surface $\mathcal{M}^{(j)}$ of sensor $j$ with density $p^{(j)}(\mathbf{x_1})$
 2: Sample direction $\omega_1$ from $\mathbf{x_1}$ such that $W^{(j)}(\mathbf{x_1}, \omega_1) > 0$ (see Section 3.5.2).
    Find the nearest point along direction $\omega_1$ from $\mathbf{x_1}$, $vp(\mathbf{x_1}, \omega_1)$
 3: $\mathbf{x_2} \leftarrow vp(\mathbf{x_1}, \omega_1)$
 4: $l \leftarrow 2$
 5: while $\mathbf{x_l} \notin \mathcal{M}_s$ do
 6:     Sample direction $\omega_l$ with density $p_{\mathbf{x_l}}(\omega)$ such that $f_s(\mathbf{x_l}, \omega_{l-1} \to \omega_l) > 0$.
        Find the nearest point along direction $\omega_l$ from $\mathbf{x_l}$, $vp(\mathbf{x_l}, \omega_l)$,
 7:     $\mathbf{x_{l+1}} \leftarrow vp(\mathbf{x_l}, \omega_l)$
 8:     $l \leftarrow l + 1$
 9: end while
10: $\overline{x} = (\mathbf{x_1} \dots \mathbf{x_l})$
```

This sampler simply tracks a light source in the scene by randomly casting paths from the camera. The density of such a sampled path is trivial and is given by:

$$p(\overline{x}) \; = \; p^{(j)}(\mathbf{x_1}) \cdot \prod_{k=1}^{l-1} p(\mathbf{x_k} \to \mathbf{x_{k+1}}) \; = \; p^{(j)}(\mathbf{x_1}) \cdot \prod_{k=1}^{l-1} p_{\mathbf{x_k}}(\omega_k) \cdot \frac{|cos(\theta'_{i,k})|}{||\mathbf{x_k} - \mathbf{x_{k+1}}||^2}$$

Unfortunately, this sampler is totally inefficient. For the very simple and common case where light sources have only small areas, the probability of hitting a light source is small and finding them in a fast way is thus almost impossible with this technique. One may argue that arbitrarily casting a ray (often called "shadow" ray) provides a better solution. It is actually the first step towards a bidirectional path tracer. We present such a sampler in the next section. Once again, this sampler is not the solution commonly implemented in renderers. However, it is a practical example to understand bidirectional samplers in the computer graphics literature.

4.1.4 A "Path Integral Formulated" (but *inefficient*) Bidirectional Path Tracer

In this section, we first assume that we set a function P_s defined on $\mathcal{M}$ such that:

$$\forall \mathbf{x} \in \mathcal{M},\ P_s(\mathbf{x}) \in [0 \dots 1] \text{ and } P_s(\mathcal{M}) \ = \ 1$$

P_s actually provides the probability to stop the sampling of the current path. This leads to the sampler given by Algorithm 3.

Algorithm 3 "Path Integral Formulated" Bidirectional Path Tracer: sample path $\overline{x}$

```
1: Sample point $\mathbf{x_1}$ over the surface $\mathcal{M}^{(j)}$ of sensor $j$ with density $p^{(j)}(\mathbf{x_1})$
2: $l_c \leftarrow 1$
3: while $uniform\_random\_0\_1() < P_s(\mathbf{x_{l_c}})$ do
4:     Sample direction $\omega_{l_c}$ with density $p_{\mathbf{x_{l_c}}}(\omega_{l_c})$ such that:
       if $l_c = 1$ then $W^{(j)}(\mathbf{x_{l_c}}, \omega_{l_c}) > 0$
       else $f_s(\mathbf{x_{l_c}}, \omega_{l_c-1} \rightarrow \omega_{l_c}) > 0$.
5:     Find the nearest point along direction $\omega_{l_c}$ from $\mathbf{x_{l_c}}$ $vp(\mathbf{x_{l_c}}, \omega_{l_c})$
6:     $\mathbf{x_{l_c+1}} \leftarrow vp(\mathbf{x_{l_c}}, \omega_{l_c})$
7:     $l_c \leftarrow l_c + 1$
8: end while
9: Sample point $\mathbf{y_1}$ over the light source surface set $\mathcal{M}_s$ with density $p_s(\mathbf{y_1})$
10: $l_s \leftarrow 1$
11: while $uniform\_random\_0\_1() < P_s(\mathbf{x_{l_s}})$ do
12:     Sample direction $\omega_{l_s}$ with density $p_{\mathbf{y_{l_s}}}(\omega_{l_s})$ such that:
        if $l_s = 1$ then $L_e(\mathbf{y_{l_s}}, \omega_{l_s}) > 0$
        else $f_s(\mathbf{y_{l_s}}, \omega_{l_s-1} \rightarrow \omega_{l_s}) > 0$.
13:     Find the nearest point along direction $\omega_{l_s}$ from $\mathbf{y_{l_s}}$ $vp(\mathbf{y_{l_s}}, \omega_{l_s})$
14:     $\mathbf{y_{l_s+1}} \leftarrow vp(\mathbf{y_{l_s}}, \omega_{l_s})$
15:     $l_s \leftarrow l_s + 1$
16: end while
17: $\overline{x} = (\mathbf{x_1} \dots \mathbf{x_{l_c}}, \mathbf{y_{l_s}} \dots \mathbf{y_1}) = (\mathbf{x_1} \dots \mathbf{x_{l_c+l_s}})$
```

The density of the resulting path is given by:

$$\begin{aligned}
p(\overline{x}) \quad = \quad & p(\mathbf{x_1}) \cdot p_s(\mathbf{x_{l_c+l_s}}) \sum_{i=1}^{l_c+l_s-2} \prod_{k=1}^{i-1} p_{\mathbf{x_k}}(\overrightarrow{\mathbf{x_k x_{k+1}}}^{(u)}) \cdot \frac{|\mathbf{N}(\mathbf{x_{k+1}}) \cdot \overrightarrow{\mathbf{x_k x_{k+1}}}^{(u)}|}{||\mathbf{x_k} - \mathbf{x_{k+1}}||^2} \\
\times \quad & \prod_{k=l_s+l_c}^{i+2} p_{\mathbf{x_k}}(\overrightarrow{\mathbf{x_k x_{k-1}}}^{(u)}) \cdot \frac{|\mathbf{N}(\mathbf{x_{k-1}}) \cdot \overrightarrow{\mathbf{x_k x_{k-1}}}^{(u)}|}{||\mathbf{x_k} - \mathbf{x_{k-1}}||^2} \\
\times \quad & \underbrace{\prod_{k=1}^{i-1} P_s(\mathbf{x_k}) \cdot \prod_{k=l_s+l_c}^{i+2} P_s(\mathbf{x_k}) \cdot [1 - P_s(\mathbf{x_i})] \cdot [1 - P_s(\mathbf{x_{i+1}})]}_{\substack{\text{Probability to have sampled } i \text{ points from the camera} \\ \text{and } l_c + l_s - i \text{ points from the light}}} \qquad (4.5)
\end{aligned}$$

where $\prod_{i\in\emptyset} \mathbf{x_i} = 1$

This sampler therefore proposes to build two independent paths and to connect their respective ending points. Thus, $f^{(j)}(\overline{x}) \neq 0$ if and only if $vp(\mathbf{x_{l_c}}, \overrightarrow{\mathbf{x_{l_c} y_{l_s}}}^{(u)}) = \mathbf{y_{l_s}}$. Therefore, if there is any occluder between $\mathbf{x_{l_c}}$ and $\mathbf{y_{l_s}}$, path $\overline{x}$ does not bring any contribution to sensor j.

The expression given by Equation 4.5 may seem a bit complicated but it is actually tightly related to the sampling process. The question (which is the most common one while using non-trivial samplers) is: what is the density of the sample? To answer this question, we have to determine *all the possible manners* to generate this path with *this* sampler. Here, Algorithm 3 provides two paths $(\mathbf{x_k})_{k\in[1\ldots l_c]}$ and $(\mathbf{y_k})_{k\in[1\ldots l_s]}$ which have been concatenated in one path $\overline{x} = (\mathbf{x_k})_{k\in[1\ldots l_c+l_s]}$. The *same* sampler could have generated this path with any other combination of camera path length i and light path length j as long as $i + j = l_c + l_s$. Therefore, density of path $\overline{x}$ can be expressed as the weighted sum of all densities provided by all possible manners to sample the path. We may finally notice that the density of path $\overline{x}$ is very close to the conditional densities of bidirectional mutations of the Metropolis Light Transport algorithm [88].

Unfortunately, this elegant bidirectional path tracer is once again very inefficient. The main reason is that casting rays in the scene is very expensive and it becomes fundamental to intensively reuse the paths we are building. The other (and in fact, commonly used) technique would consist of casting shadow rays between all possible $\mathbf{x_p}$ and $\mathbf{y_q}$ point pairs. It however seems difficult to formalize such a sampler with the path integral formulation: indeed, it would not generate one path at a time but, on the contrary, a large family of strongly correlated paths.

In the literature of Monte-Carlo rendering, all these techniques are based on the distinction among path lengths, the intensive use of Russian roulettes and the explicit combination of several estimators. We present these approaches in the next section.

4.2 Monte-Carlo Rendering by Explicitly Handling Path Lengths

In this section, we present the most common way to build samplers in computer graphics. In the previous section, we showed that directly designing samplers using the path integral formulation is not necessarily an *efficient* strategy. On the contrary, specifically sampling paths with fixed lengths is generally more effective.

4.2.1 Subdividing the Path Space with Path Lengths

Equation 4.3 can be reformulated in the following manner (see Section 4.1.1.3):

$$\begin{aligned} I_j &= \sum_{k=1}^{\infty} \int_{\mathcal{M}^{k+1}} \Big[L_e(\mathbf{x_k} \rightarrow \mathbf{x_{k-1}}) G(\mathbf{x_0} \leftrightarrow \mathbf{x_1})\ W_e^{(j)}(\mathbf{x_1} \rightarrow \mathbf{x_0}) \\ & \quad (\prod_{i=1}^{k-1} f_s(\mathbf{x_{i+1}} \rightarrow \mathbf{x_i} \rightarrow \mathbf{x_{i-1}}) G(\mathbf{x_i} \leftrightarrow \mathbf{x_{i+1}})) dA(\mathbf{x_0})\ \ldots\ dA(\mathbf{x_k}) \Big] \\ &= \sum_{k=1}^{\infty} \int_{\Omega_k} f_k^{(j)}(\overline{x}_k) d\mu_k(\overline{x}_k) \ = \ \sum_{k=1}^{\infty} I_{j,k} \end{aligned} \tag{4.6}$$

Instead of explicitly gathering all paths in only one path space, an alternative would be to evaluate all the terms of the sum. To achieve such a result in an efficient way, the following strategies are currently used in all "path tracing systems":

- If $\overline{X_k}$ is a random variable used to sample Ω_k, define $\overline{X_{k+1}}$ such that $\overline{X_{k+1}}$ and $\overline{X_k}$ are correlated and at least a sub-path of $\overline{X_k}$ is reused for $\overline{X_{k+1}}$;
- As sampling an infinity of random variables is impossible, use a Russian roulette to avoid any further sampling after some random rank p.

Many path samplers actually use those two techniques to compute an unbiased solution of the measurement equation.

4.2.2 Standard Path Tracing

Standard Path Tracing proposed by Kajiya [42] was the first algorithm attempting to tackle the light transport problem in its global form. This algorithm is actually not really different from the modified path tracing algorithm we presented in Section 4.1.3. Similarly to the approach we detailed, a light path is first generated backward from the camera for each camera sensor. However, to make the approach more efficient, Kajiya proposed to cast a visibility ray towards one light source for each light bounce made by the current path. As indicated by the previous sub-section, this technique therefore consists of reusing the camera path sampled for path space Ω_k while sampling path space Ω_{k+1}. Conceptually, at each sampling step, an infinity of paths is sampled (one for each path length). Practically, a Russian roulette is used to set that after some random rank, all contributions due to the "non-sampled" paths are equal to zero (see Algorithm 4).

Algorithm 4 Standard Path Tracer: set maximum sampling rank p and sample path family $(\overline{x_l})_{l\in[1\ldots p]}$

```
1:  Sample point $\mathbf{x_1}$ over the surface $\mathcal{M}^{(j)}$ of sensor $j$ with density $p^{(j)}(\mathbf{x_1})$
2:  Sample direction $\omega_1$ from $\mathbf{x_1}$ such that $W^{(j)}(\mathbf{x_1}, \omega_1) > 0$ (see Section 3.5.2).
    Find the nearest point along direction $\omega_1$ from $\mathbf{x_1}$, $vp(\mathbf{x_1}, \omega_1)$
3:  $x_2 \leftarrow vp(\mathbf{x_1}, \omega_1)$
4:  Sample a point $\mathbf{x_s}$ across the surface set $\mathcal{M}_s$ of the light sources with density
    $p_s(\mathbf{x_s})$.
5:  $\overline{x_1} \leftarrow (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_s})$
6:  $l \leftarrow 2$
7:  **while** $uniform\_random\_0\_1() < P_s(\mathbf{x_l})$ **do**
8:      Sample direction $\omega_l$ with density $p_{\mathbf{x_l}}(\omega)$ such that $f_s(\mathbf{x_l}, \omega_{l-1} \to \omega_l) > 0$.
        Find the nearest point along direction $\omega_l$ from $\mathbf{x_l}$, $vp(\mathbf{x_l}, \omega_l)$
9:      $x_{l+1} \leftarrow vp(\mathbf{x_l}, \omega_l)$
10:     Sample point $\mathbf{x_s}$ from $\mathcal{M}_s$ with density $p_s(\mathbf{x_s})$.
11:     $\overline{x_l} \leftarrow (\mathbf{x_1} \ldots \mathbf{x_{l+1}}, \mathbf{x_s})$
12:     $l \leftarrow l + 1$
13: **end while**
14: $p \leftarrow l$
```

As shown by Algorithm 4, a maximum rank p is randomly set. It simply depends on the sequence of sampled points $(\mathbf{x_k})_{k\in[1\ldots p]}$. The density of each path $\overline{x_l}$ can then be expressed as the product of the probability to reach rank l and the product of the successive conditional densities of each point (since each path

$\overline{x_k}$ is once again a truncated Markov-Chain). Therefore:

$$\forall l \in [1 \dots p],\ p(\overline{x}) \ = \ \prod_{k=2}^{l} P_s(\mathbf{x_k}) \cdot p^{(j)}(\mathbf{x_1}) \cdot \prod_{k=1}^{l-1} p_{\mathbf{x_k}}(\omega_k) \cdot \frac{|cos(\theta'_{i,k})|}{||\mathbf{x_k} - \mathbf{x_{k+1}}||^2}$$

Thus, if $l \leq p$, this sampler provides the estimator $f_l^{(j)}(\overline{x_l})/p(\overline{x_l})$. If $l > p$, the paths are supposed to bring no contribution to sensor j. We may finally remark that this sampler is a bit simplified since it does not handle light sources directly visible from the camera or specific paths with specular materials which cannot be sampled from the camera. The common solution is to specifically handle several estimators depending on the kinds of paths which are generated. This is actually one of the goals of bidirectional path tracers.

4.2.3 Bidirectional Path Tracing

Even if the standard path tracing algorithm remains unconditionally robust and unbiased, variance problems lead to implement more predictive algorithms. In fact, since generating light paths only from the camera is a bit arbitrary, sampling them also from the light sources may be a good idea: this is the core of bidirectional samplers initially proposed by Lafortune and Willems [55] and Veach and Guibas [86]. As the standard path tracer presented above, bidirectional path tracers propose to sample path spaces $(\Omega_k)_{k\in\mathbb{N}}$ and to build several correlated estimators for each path length. They are actually efficient implementations of the "Path Integral Formulated" bidirectional path tracer exposed in Section 4.1.4.

Algorithm 5 presents a form of bidirectional sampler. Two paths are first built, one from the camera sensor and one from the light source surface set. This sampling process then provides $p_c \times p_s$ correlated light paths. Similarly to the standard path tracing algorithm, the density of path x_{l_c,l_s} can be expressed as the product of the probabilities of reaching ranks l_c and l_s, the densities of point

Algorithm 5 Standard Bidirectional Path Tracer: set maximum sampling ranks p_c and p_s and sample path family $(\overline{x_{l_c,l_s}})_{(l_c,l_s)\in[1\ldots p_c]\times[1\ldots p_s]}$

```
 1: Sample point $\mathbf{x_1}$ over the surface $\mathcal{M}^{(j)}$ of sensor $j$ with density $p^{(j)}(\mathbf{x_1})$
 2: $l_c \leftarrow 1$
 3: **while** $uniform\_random\_0\_1() < P_s(\mathbf{x_{l_c}})$ **do**
 4:    Sample direction $\omega_{l_c}$ with density $p_{\mathbf{x_{l_c}}}(\omega_{l_c})$ such that:
       **if** $l_c = 1$ **then** $W^{(j)}(\mathbf{x_{l_c}}, \omega_{l_c}) > 0$
       **else** $f_s(\mathbf{x_{l_c}}, \omega_{l_c-1} \rightarrow \omega_{l_c}) > 0$.
 5:    Find the nearest point along direction $\omega_{l_c}$ from $\mathbf{x_{l_c}}$, $vp(\mathbf{x_{l_c}}, \omega_{l_c})$
 6:    $\mathbf{x_{l_c+1}} \leftarrow vp(\mathbf{x_{l_c}}, \omega_{l_c})$
 7:    $l_c \leftarrow l_c + 1$
 8: **end while**
 9: Sample point $\mathbf{y_1}$ over the light source surface set $\mathcal{M}_s$ with density $p_s(\mathbf{y_1})$
10: $l_s \leftarrow 1$
11: **while** $uniform\_random\_0\_1() < P_s(\mathbf{x_{l_s}})$ **do**
12:    Sample direction $\omega_{l_s}$ with density $p_{\mathbf{y_{l_s}}}(\omega_{l_s})$ such that:
       **if** $l_s = 1$ **then** $L_e(\mathbf{y_{l_s}}, \omega_{l_s}) > 0$
       **else** $f_s(\mathbf{y_{l_s}}, \omega_{l_s-1} \rightarrow \omega_{l_s}) > 0$.
13:    Find the nearest point along direction $\omega_{l_s}$ from $\mathbf{y_{l_s}}$, $vp(\mathbf{y_{l_s}}, \omega_{l_s})$
14:    $\mathbf{y_{l_s+1}} \leftarrow vp(\mathbf{y_{l_s}}, \omega_{l_s})$
15:    $l_s \leftarrow l_s + 1$
16: **end while**
17: $p_c \leftarrow l_c$
18: $p_s \leftarrow l_s$
19: **for** $l_c = 1$ to $p_c$ **do**
20:    **for** $l_s = 1$ to $p_s$ **do**
21:       $\overline{x_{l_c,l_s}} \leftarrow (\mathbf{x_1} \ldots \mathbf{x_{l_c}}, \mathbf{y_{l_s}} \ldots \mathbf{y_1})$
22:    **end for**
23: **end for**
```

$\mathbf{x_1}$ and $\mathbf{y_1}$ and the conditional densities of each other sampled points. Therefore:

$$\begin{aligned} p(\overline{x}) &= p(\mathbf{x_1}) \cdot \prod_{k=1}^{l_c-1} p_{\mathbf{x_k}}(\overrightarrow{\mathbf{x_k x_{k+1}}}) \cdot \frac{|\mathbf{N}(\mathbf{x_{k+1}}) \cdot \overrightarrow{\mathbf{x_k x_{k+1}}}|}{||\mathbf{x_k} - \mathbf{x_{k+1}}||^2} \\ &\times \; p_s(\mathbf{y_1}) \cdot \prod_{k=1}^{l_s-1} p_{\mathbf{y_k}}(\overrightarrow{\mathbf{y_k y_{k+1}}}) \cdot \frac{|\mathbf{N}(\mathbf{y_{k+1}}) \cdot \overrightarrow{\mathbf{y_k y_{k+1}}}|}{||\mathbf{y_k} - \mathbf{y_{k+1}}||^2} \\ &\times \; \underbrace{\prod_{k=1}^{l_c-1} P_s(\mathbf{x_k}) \cdot \prod_{k=1}^{l_c-1} P_s(\mathbf{y_k})}_{\text{Russian roulette}} \end{aligned} \tag{4.7}$$

The further idea is to build several estimators for each path length l. We first consider all paths $(\overline{x_{l_c,l_s}})$ such as $l_c + l_s = l$ and combine all their contributions to design estimator $J_{j,l}$ thus defined by:

$$J_{j,l} \;=\; \sum_{l_c+l_s=\;l} w_{l_c,s}(\overline{X_{l_c,l_s}}) \cdot \frac{f_l^{(j)}(\overline{X_{l_c,l_s}})}{p(\overline{X_{l_c,l_s}})}$$

where $\sum_{l_c+l_s=\;l} w_{l_c,s}(\overline{X_{l_c,l_s}}) = 1$ and $w_{l_c,s}(\overline{X_{l_c,l_s}}) \in [0\ldots1]$. As with standard path tracer:

$$E\left[\sum_{l=2}^{\infty} J_{j,l}\right] = I_j$$

The choice of the weight families $w_{i,j}$ is fundamental: the most common technique is to use multiple importance sampling as presented in Section 2.3.4.3. Once again, the algorithm presented here is a bit simplified. For example, in the more sophisticated sampler proposed by Veach and Guibas, length 1 paths are also handled.

4.2.4 Light Tracing

Light tracing (or "particle tracing") algorithms [21] are not really different from path tracing ones. As they consist of sampling the light paths from the light sources instead of the camera, they are actually a simpler form of bidirectional path tracers as presented in the previous section.

4.2.5 Metropolis Light Transport

Metropolis Light Transport [88] is a successful algorithm presented by Veach and Guibas. As Metropolis sampling strategies are used in two of our contributions (Metropolis Instant Radiosity and Coherent Metropolis Light Transport with Multiple-Try Mutations), this technique will be detailed in Chapters 8 and 9. To sum up the approach, we may however say that Metropolis Light Transport consists of locally exploring the sampled space Ω with a set of path *mutations*.

4.3 Designing Efficient Monte-Carlo Renderers

While bidirectional paths tracers are the cores of most of the samplers used in Monte-Carlo rendering, it exists many numerical techniques which aim at making the computations more effective. They either propose to *bias* the estimators to make them more efficient or to factorize and reexpress all the computations to make the algorithms faster. In this section, we present the most classical examples.

4.3.1 Photon Mapping

The photon mapping technique [38–40] is the most famous example of trade-off between rendering speed and bias in the field of computer graphics. It is actually a two-pass method that is able to handle many illumination features in an efficient manner. The technique actually consists in sampling light paths from the light sources and in storing path information with the compact representation given by photons. Once the photons have been stored, the incoming radiance value is computed by using a non-parametric estimation pass. The estimation pass which consists in averaging and weighting neighbor photon contributions by using kernel functions, makes the estimators biased but *consistent* since the error due to the estimation tends to 0 when the number of photons increases.

In fact, we do not detail more particularly the algorithm but one may find a very detailed explanations in Chapter 16 of the book by Pharr and Humphreys, Physically Based Rendering [64]: the authors furthermore expose the formalism of photon mapping techniques in a manner closed to ours. An interesting and complete survey of photon mapping methods can also be found in Suykens Ph.D. [79]. Some interesting work has also been achieved by Dmitriev et al. [18], Guenther et al. [32] and Wald et al. [94] to make photon mapping suitable to interactive rendering.

4.3.2 Environment Mapping

Environment mapping techniques [27] consist in replacing the whole incoming radiance field by a radiance picture which provides the incoming radiance for each possible direction ω. This simplification leads to consider that the parts of the scene seen by the camera are only illuminated by a set of objects located at an infinite distance. The method is commonly used to represent sky dome or simply, to make the computations easier while illuminating one given object.

There is an important amount of recent work aiming at tackling variance problems and at proposing interactive rendering while using this kind of methods.

To achieve interactive rendering, most of the researchers propose to use expensive pre-computations [36, 44, 45] or to project the illumination and/or the BRDFs of the objects into finite bases such as spherical harmonics [65, 66, 75] and wavelets [57]. These techniques therefore involve a second source of bias since the environment maps (and sometimes the BRDFs) are compressed.

Other classical Monte-Carlo approaches consist of reducing variance. Most of these techniques are based on the notion of bidirectional sampling [6] and take into account the energy of incident illumination as well as the BRDF in the sampling process by either using a sampling / resampling step [82,83], sequential sampling [25] or wavelet representations of the BRDF and the environment map [10].

4.3.3 Irradiance and Radiance Caching

Irradiance caching proposed by Ward et al. [101,102] is, with photon mapping, one of the most efficient trade-offs between speed and bias. It exploits the smoothness of indirect illumination by sampling the irradiance sparsely over surfaces, caching the results and interpolating them. For each ray hitting a surface, the irradiance cache is queried. If one or more irradiance records are available, the irradiance is interpolated. Otherwise a new irradiance record is computed by sampling the hemisphere and is added to the cache. In this way, the cache is progressively updated in a view dependent way. More recently, Krivanek [53] proposed to store the incoming radiance function over the hemisphere thereby allowing interpolation to be applied to glossy surfaces.

In the next section, we present with more details, Instant Radiosity [47]. As photon mapping, Instant Radiosity is a Monte-Carlo rendering technique which focuses on rendering speed and efficiency.

4.4 Instant Radiosity

4.4.1 Principle

Instant Radiosity [47] (IR) is a powerful method to compute global illumination for diffuse or not-too-shiny materials. It actually consists in virtually splitting each path $\overline{x} = \{\mathbf{x_0}, \mathbf{x_1}, \ldots, \mathbf{x_n}\}$ into three parts:

- $\overline{x}_c = \{\mathbf{x_0}, \mathbf{x_1}\}$ where $\mathbf{x_0}$ is a location on a sensor and $\mathbf{x_1}$ is a point seen by this sensor;
- x_v is a point located just after $\mathbf{x_1}$: in a sense, $\mathbf{x_v}$ "illuminates" $\mathbf{x_1}$ and it is the Virtual Point Light (VPL) location we are looking for;
- $\overline{x}_s$ is the remainder of the path. Its end is connected to a light source while its first point is connected to $\mathbf{x_v}$. We may notice that $\overline{x}_s$ can be void if $\mathbf{x_v}$ is located on a light source.

Using the path integration formulation, we then generate and store N random paths $\{\mathbf{x_v}, \overline{x}_s\}$ from the light sources and reuse these sub-paths for all camera pixels (i.e. the camera sub-paths $\{\mathbf{x_0}, \mathbf{x_1}\}$). Therefore, we have for each pixel j:

$$I_j = E\left(\frac{f^{(j)}(\overline{x})}{p(\overline{x})}\right) = E\left(\frac{f^{(j)}(\{\overline{x}_c, \mathbf{x_v}, \overline{x}_s\})}{p(\{\overline{x}_c, \mathbf{x_v}, \overline{x}_s\})}\right)$$

As $\overline{x}_c$ and $\{\mathbf{x_v}, \overline{x}_s\}$ are independent:

$$p\left(\{\overline{x}_c, \mathbf{x_v}, \overline{x}_s\}\right) = p\left(\overline{x}_c\right)p(\{\mathbf{x_v}, \overline{x}_s\}))$$

Once again, two sampling strategies can be designed:

- In the original presentation of the idea by Keller, the path lengths were actually distinguished. Therefore, in the most common implementation of Instant Radiosity, different length but correlated paths are generated during the same sampling process and one or several estimators are built for each path length;
- An alternative implementation can directly generate paths of any length and therefore sample the whole path space Ω.

The efficiency of Instant Radiosity relies on the fact that the $VPLs$ are reused for all the sensors in the scene and that the global illumination problem comes down to visibility requests between the $VPLs$ and a point of the scene.

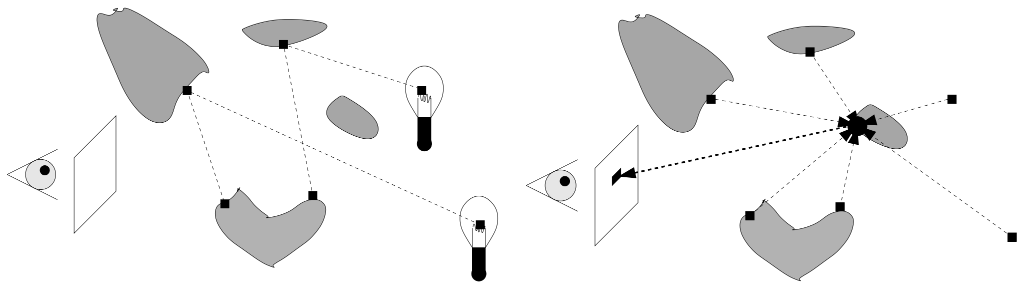


Figure 4.2: **Instant Radiosity.** (a) First, light paths are propagated from the physical light sources. Each bounce is stored as a Virtual Point Light or VPL. (b) Then, all lighting contributions (direct and indirect) are replaced by the set of VPLs. Evaluating the incoming radiance field on sensor j becomes equivalent to accumulating all VPL contributions.

4.4.2 The Overmodulation Problem

Instant Radiosity suffers from a serious overmodulation problem. In the simple form described above, the estimators produced while using VPLs *do not have bounded variance*. Indeed, while evaluating the VPL contribution, visibility term G has to be evaluated and as the $1/r^2$ term is not bounded, the VPL contribution tends to ∞ when the distance between the VPL and the illuminated point tends to 0. This property makes that Instant Radiosity does not provide Monte-Carlo estimators since the Central Limit Theorem (see Chapter 2) does not apply. We therefore have no information concerning the convergence speed since variance does not exist.

The most common solution is to bound the visibility term G so that the resulting estimators become biased and remain *efficient*. Furthermore, inspired by the work by Heinrich [37] who proposed an optimal method for the Monte Carlo approximation of weakly singular operators in the context of parametric integration, Kollig and Keller [51] propose to compute in an *unbiased* manner the bias introduced when clamping visibility term G.

4.4.3 Variance Reduction Techniques

The location of the VPLs is a crucial problem. Indeed, it is fundamental to place VPLs only on surfaces which are able to illuminate parts of the scene seen by the camera. Indeed, the "gathering" step which consists in computing the sensor response to the VPLs (i.e. illuminating the screen pixels) is very expensive. Surprisingly, there are few researches explicitly dealing with the placement of VPLs. The "Power Sampling" technique proposed by Wald et al. [98] proposes to compute the power brought by each *physical* light source to the camera through direct or indirect contributions and build the corresponding discrete cumulative density function. They finally sample light paths or VPLs by choosing the light sources proportionally to the power they bring to the camera. Even it is, in the literature, the only example of such variance reduction for the VPL generation, there are a lot of researches which have been recently made for the photon sampling step in photon mapping. All these techniques are generally said to be "importance driven". Importance has been used by many researchers to speed up the convergence rates of off-line rendering in radiosity or Monte-Carlo methods. A detailed discussion is available in [7]. More particularly, a subsequent work has been achieved with photon map techniques [40]. Even if photon map algorithms are numerically quite different from Instant Radiosity (the first ones are non-parametric estimation methods while the second one is an analytical MC approach), the VPL generation is quite similar to the photon generation. Peter and Pietrek propose to trace importance particles from the eye point [62]. The resulting distribution is then used to guide the photon tracing step. The method was extended by Keller and Wald [49] and Suykens and Willems [80] to provide more robust numerical results. All these methods can be used to guide the VPL generation. However, as all of them require to sample a large number of importons (particles generated from the camera), they unfortunately seem unsuitable to real-time rendering. That is why we propose in Chapters 7 and 8 efficient samplers for the VPL generation.

4.4.4 Rendering Systems Using Instant Radiosity

There are several rendering systems which have been proposed and combined with Instant Radiosity. The most famous one is certainly "Instant Global Illumination" designed by Wald et al. [97], and improved by Benthin et al. [5]. It aims at computing realistic outputs with a VPL set: they both distribute screen tiles over clustered commodity PCs and decrease the required fillrate by using interleaved sampling invented by Keller and Heidrich [48]: instead of computing the contributions of every VPL for *all* pixels, interleaved sampling suggests distributing uncorrelated sub-sets of VPLs inside $n \times m$ sample patterns (see Chapters 5 and 6 for a detailed description of this technique). By using this approximation, they easily achieved interactive and even real-time frame rates with a cluster of about 20 PCs for both simple and complex scenes (counting millions triangles).

More recently, Bruce Walter et al. [99,100] proposed to use a binary light tree containing all VPLs and a perceptual metric to adaptively partition the lights into groups. This method introduces a small bias due to the "cuts" introduced during the rendering process but offers a good trade-off between speed and quality.

In this thesis, we propose new systems using GPUs to render the contributions of a VPL set: Chapters 6 and 5 present them in details. In Chapter 8, we will also use a rendering system close to Instant Global Illumination.

4.5 Conclusion

In this chapter, we presented the roots of any Monte-Carlo renderer attempting at tackling the light transport problem: the path integral formulation and the path tracing sampling techniques. Of course, this is only a very small part of Monte-Carlo rendering and more generally a very small part of all existing rendering techniques. The book by Pharr and Humphreys [64] is an excellent and complete introduction to physically based rendering. More directly related to *Monte-Carlo* rendering, the Ph.D. thesis of Veach [85] is certainly today the most precise and formalized work and will help any person wanting to design a Monte-Carlo renderer.

CHAPTER 5

Non-Interleaved Deferred Shading of Interleaved Sample Patterns

In this chapter, we present a novel and fast technique to combine interleaved sampling and deferred shading on a GPU. The goal is to offer a simple and fast sampling scheme which aims at speeding up the gathering pass while using Instant Radiosity and the set of Virtual Point Lights (VPLs) provided by this method (see Chapter 4). In this chapter, we do not therefore propose a new sampling scheme but only an efficient implementation of the final pass of the algorithm. For new ways to sample VPLs, one may refer to Chapters 7 and 8.

The core idea of the method is quite simple. Interleaved sample patterns are computed in a non-interleaved deferred shading process. The geometric buffer (G-buffer) which contains all of the pixel information is actually split into several separate and distinct sub-buffers. Formally, the texels inside a $n \times m$ regular pattern are dispatched over different regions of the screen, the separate sub-buffers. Hence, texel (x, y) will go to texel $(x/n, y/m)$ belonging to sub-buffer $(i,\ j)$ with $i\ =\ x\ mod\ n$ and $j\ =\ y\ mod\ m$. To achieve such a result in a fast way, a massive two-pass swizzling copy is used to convert between these two buffer organizations. Once split, the sub-buffers can then be accessed to perform any fragment operation as it is done with a standard deferred shading rendering pipeline. By combining interleaved sampling and deferred shading, real time rendering of global illumination effects can be therefore easily achieved. Instead of evaluating each light contribution on the whole geometric buffer, each shading computation is coherently restricted to a smaller subset fragments using the sub-buffers. Therefore, each screen pixel in a regular $n \times m$ pattern will have its own small set of light contributions. Doing so, the consumed fillrate is considerably decreased and the provided rendering quality remains close to the quality obtained with a non-interleaved approach. The implementation of this rendering pipeline is finally straightforward and it can be easily integrated in any existing real-time rendering package already using deferred shading. As an application of this technique, we propose to efficiently handle a set of virtual point lights and therefore to deal with indirect illumination effects. Figure 5.1 presents some results obtained with the approach.

The chapter is organized as follows. Section 5.1 sums up the related techniques, deferred shading and interleaved sampling. Section 5.2 presents our core technique i.e. the buffer splitting strategy used to perform interleaved sampling and all the following shading operations. Section 5.3 shows how the standard deferred shading rendering pipeline can be greatly accelerated by combining it with the buffer splitting technique. Sections 5.4 and 5.5 are devoted to the results obtained for every step of the pipeline and two different kinds of applications. A conclusion is finally given in Section 5.6.

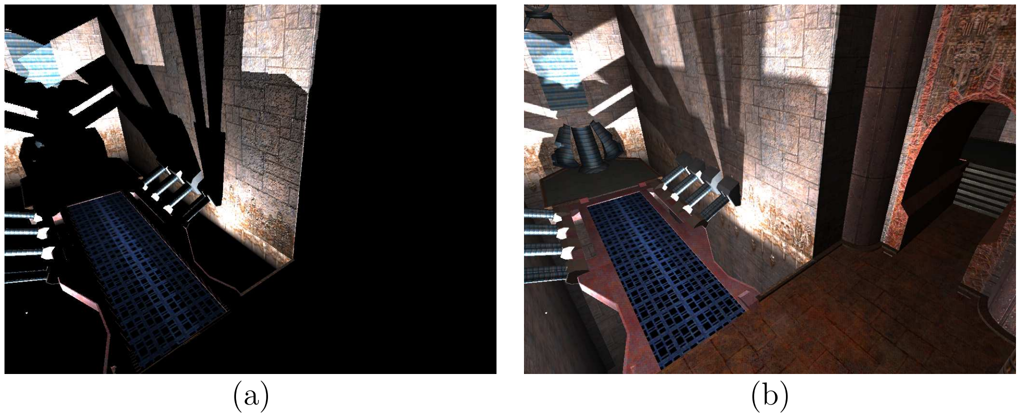


Figure 5.1: **The Indirect Lighting Impact.** (a) presents a snapshot of Q3tourney2 (courtesy of Id Software) with direct lighting only (two point light sources). The screen resolution is 1024 × 768. (b) The same scene with 512 secondary point light sources. Using interleaved deferred shading, the scene is rendered at 17 f/s on a GeForce 6800 GT and 33 f/s on a NVidia GeForce 7800 GT.

5.1 Overview of Related Techniques

The method proposed in this chapter deals with two pre-existing techniques: interleaved sampling and deferred shading. We present them in details in this section.

5.1.1 Deferred Shading

The first technique we use here is deferred shading. Deferred shading first introduced by Deering et al. [16] then developed by Saito and Takahashi [71] suggests storing the geometric and material information in buffers (generally called ”G-buffer”) and reusing them to perform the lighting computations. It greatly simplifies the rendering pipeline and it also prevents the geometry from being reprojected each time a shading pass is performed.

5.1.2 Interleaved Sampling

The second technique we use is interleaved sampling. Since Cook et al. [13] [12] presented several Monte-Carlo strategies using camera path samples and incoherent sampling from one pixel to the other, many sampling methods have been proposed. Recently, Keller and Heidrich presented a simple interleaved sampling strategy [48] quite suitable to graphics hardware. As shown by Figure

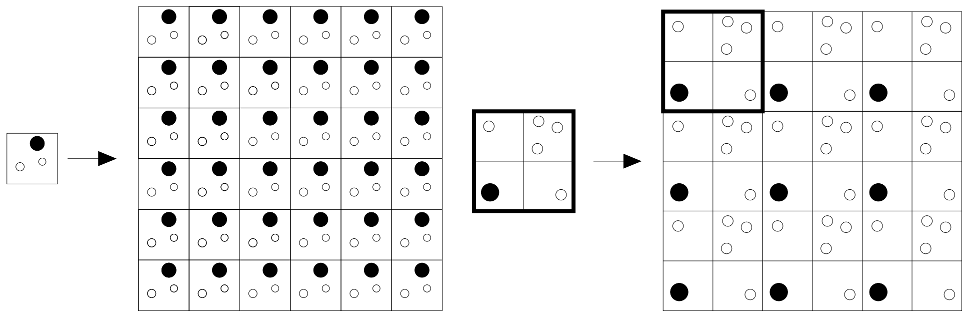

(a) Standard Sampling (b) Interleaved Sampling

Figure 5.2: **Standard and Interleaved Samplings.** (a) presents a regular sample pattern typically used with accumulation buffers while (b) shows an example of 2×2 interleaved sample pattern.

5.2, interleaved sample pattern families are an extension of regular sample pattern ones. Indeed, instead of sampling the same random variable families for *each* pixel, the interleaved sampling technique proposes to sample these families across $n \times m$ pixels. We may also say that interleaved samples are taken from a number of independent regular grids which are then merged into a single sampling pattern. The samples of all regular grids are thus interleaved such that in the final high-quality image, adjacent pixels are not correlated. In a sense, interleaved sampling is a trade-off between completely uncorrelated sample sets as used by Cook et al. with distributed ray tracing and completely correlated sample set as used with accumulation buffers.

This sampling technique was successfully used by Ingo Wald et al. to perform interactive global illumination [97]. First, the whole incoming radiance field is replaced and represented by a set of hemispherical virtual point lights (commonly called VPLs) computed with the Instant Radiosity algorithm [47]. Then, all visibility requests are computed with an efficient and parallelized raytracer [92] [98].

To perform the integration with limited computation capabilities, the VPL contributions are interleaved. Therefore, not every pixel computes every lighting sample since each pixel in a regular pattern (3×3 for example) uses a different light sample set. The VPL contributions are then filtered using a discontinuity buffer. With a cluster of commodity computers, they achieved interactive frame rates on highly complex scenes. Unfortunately, raytracing is not currently supported by specific graphics hardware and even if significant advances were recently achieved in this domain [72] [81], commodity "RPUs" are not currently available. For these reasons, we propose to efficiently combine interleaved sampling and deferred shading by using today GPUs.

5.2 GPU-friendly Interleaved Sampling

The goal of this section is to limit computations to separate and distinct fragment subsets. In a standard deferred shading application, since a light source will illuminate *all* screen pixels, a large fillrate will be consumed if many light sources are present in the scene. With interleaved sampling and a $n \times m$ interleaved pattern, each pixel inside a $n \times m$ rectangle will have its own set of light contributions and no correlation between adjacent pixels occurs. By exploiting the coherence of neighbor pixels and blending their irradiance, the rendering quality can therefore remain close to the quality provided by non-interleaved methods with a performance equivalent to the performance obtained with a sub-sampling technique. This section introduces the shortcomings of several naive ideas to perform interleaved sampling and also presents a more GPU-friendly approach. To compare all the approaches, we will assume that a geometric buffer containing the normal, the position and the material information of each pixel has been first created and stored.

5.2.1 Naive Approaches

The obvious way to accomplish interleaved sampling is to render $n \times m$ passes by using a stencil buffer for each of them so that only one pixel in each $n \times m$ cell should be rendered. This approach was tested and as shown in Table 5.1, it gave very poor results. Indeed, even if early fragment culling is activated, the incoherence of the input data prevents the GPU from being efficient. Several variants were also explored. First, stencil tests were replaced by depth tests in order to exploit early-z culling capabilities. However, as the hierarchical Z-buffer algorithm [28] commonly used by the GPUs also requires the coherence of the depth data, the method was inefficient too. We secondly tested dynamic branching but the incoherence of the data once again caused bad results. We

	4×4	2×2
No Interleaving	75 ms	75 ms
Stencil Culling	73 ms	72 ms
Depth Culling	72 ms	74 ms
Dynamic Branching	78 ms	82 ms

Table 5.1: **Time to Accumulate 16 Point Light Contributions.** The G-buffer resolution is equal to 1024×1024. Stencil or depth culling and dynamic branching give very poor results. The incoherence of the data inside a 2×2 or 4×4 pattern strongly limits the performance.

finally tried to directly create the tiled and separate sub-buffers presented in Figure 5.4.c by accessing the G-buffer data with an indirection during *every* shading pass. Since texture accesses are very incoherent, the application becomes utterly bound by the memory latency and considerably slows down. These failures motivated more sophisticated techniques.

5.2.2 One-Pass G-Buffer Splitting

To provide the coherence needed by the GPU, explicitly splitting the G-buffer $\mathcal{G}$ into $n \times m$ smaller tiled sub-buffers $\mathcal{G}_{i,j}$ seems a good idea since operations for each light source can be performed on contiguous groups of pixels. To understand this simple concept, Figure 5.4.b gives an example of a split normal buffer. Hence, texel (x, y) from $\mathcal{G}$ goes to texel $(x/n, y/m)$ belonging to sub-buffer $\mathcal{G}_{i,\ j}$ with $i\ =\ x\ mod\ n$ and $j\ =\ y\ mod\ m$. The split can be done in a single pass with a specific fragment program. A look-up texture is first precomputed and used to move a texel of the initial G-buffer to the associated texel of the split G-buffer. Using one pass to split the buffer is unfortunately slow since memory accesses remain strongly incoherent during the splitting pass. Fortunately, the approach can be easily accelerated.

5.2.3 Two-Pass G-Buffer Splitting

The memory organization on GPUs requires special care. Indeed, like CPUs, GPUs are all the more efficient that memory and cache accesses are coherent. To provide coherence during any mapping operation, the textures are 2D block allocated by the graphics card drivers. It motivated a coherent two-pass approach.

Block Splitting. The memory and cache access incoherence during the one-pass splitting is mainly caused by the size of the manipulated data. A simple idea is therefore to limit splitting operations to small 2D blocks to enhance data locality. The initial G-buffer $\mathcal{G}$ is thus subdivided into $p \times q$ blocks and each block is split into $n \times m$ separate sub-blocks. If the blocks are small enough, memory accesses remain coherent during the pass. After this pass, each sub-buffer is subdivided into $p \times q$ sub-blocks spread across the whole buffer as shown by Figure 5.3.c.

Block Translation. To rebuild each sub-buffer, another pass performs the translation of the interleaved sub-blocks (See Figure 5.3.c and 5.3.d). Once again, the memory accesses remain coherent since entire blocks are moved.

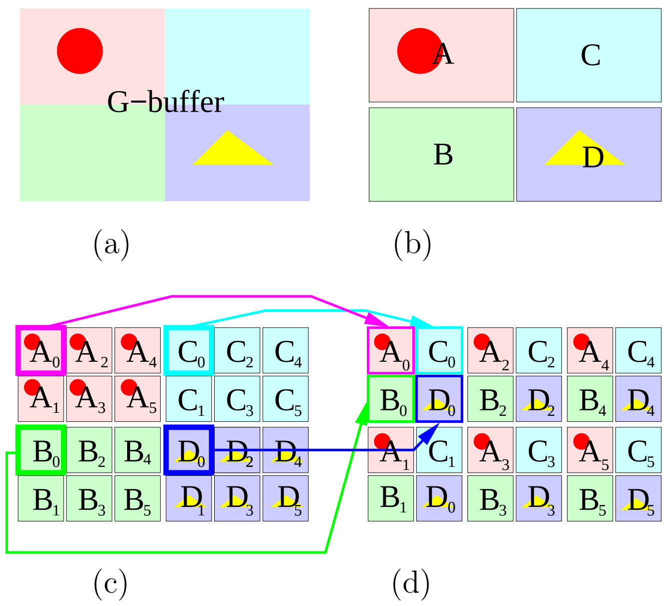


Figure 5.3: **Two-Pass G-Buffer Splitting.** The desired interleaved pattern is 3×2. (a) presents the initial G-buffer. (b) The G-buffer is subdivided in 4 (2×2) blocks. The interleaved pattern size is 3×2. (c) shows the G-buffer after the block splitting. As the interleaved pattern is 3×2, each block has been split in 6 (3×2) sub-blocks. The wanted sub-buffers $(A_i B_i C_i D_i)$ are therefore spread across the whole buffer. (d) shows how the sub-buffers $(A_i B_i C_i D_i)$ are retrieved by the block translation.

As shown in Section 5.4, this approach gives satisfactory results. In most of the cases, performing block manipulations is much more efficient than the one-pass naive approach.

5.2.4 Buffer Gathering

Once the shading operations have been achieved (See Section 5.3.3), the interleaved pattern is reconstructed by gathering the sub-buffers. This pass is the opposite of the buffer splitting one (see Figure 5.4.d) and for the same reasons, it is performed in two passes, the block translation pass and the block gathering one.

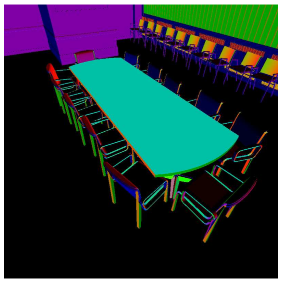
With the buffer splitting approach, uncorrelated light contributions or more generally uncorrelated computations can be made for nearby pixels in a way as generic as the one proposed by a standard deferred shading approach. In the next sections, we will show that computing and accumulating hundreds of light contributions is now possible with a decent framerate and a good rendering quality by combining buffer splitting with deferred shading and fast filtering techniques.

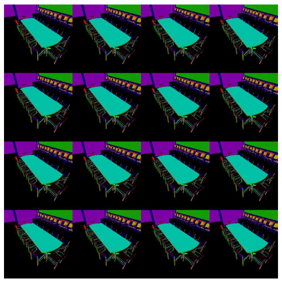
5.3 Non-Interleaved Deferred Shading of Interleaved Sample Patterns

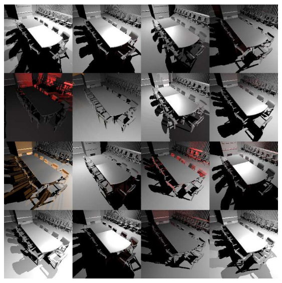
We now present an extension of deferred shading using the buffer splitting / gathering techniques. Instead of performing the shading operations on the whole G-buffer, they are restricted to low-resolution tiled sub-buffers. In comparison with standard deferred shading, three passes are added. The first one splits the G-buffer in several sub-buffers. The second one reconstructs the interleaved sampling pattern after the shading passes and the third one exploits the spatial coherence of neighbor pixels to blend uncorrelated lighting contributions. Therefore, the rendering pipeline is now decomposed into 5 steps all presented in Figure 5.4.

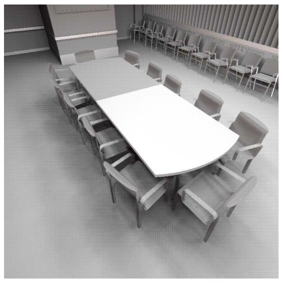
5.3.1 G-Buffer Creation (BC)

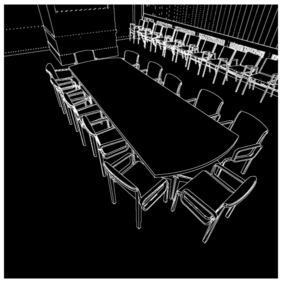
Before the shading operations, three float buffers (the G-buffer) which respectively contain positions, normals and colors are first created (see Figure 5.4.a). The material information such as material identifiers is also packed in the remaining components. For bandwidth reasons, precision is limited to 16 bits and the scene is therefore bounded.

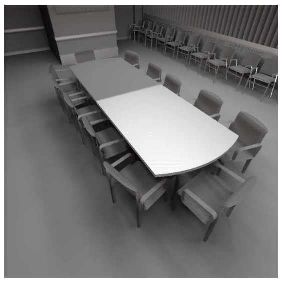

(a)


(b)


(c)


(d)


(e)


(f)

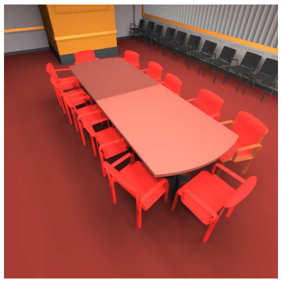

(g)

Figure 5.4: **Non-Interleaved Deferred Shading of Interleaved Sample Patterns.** (a) G-buffer Creation. (b) G-buffer Splitting. The G-buffer is subdivided into $n \times m$ separate smaller sub-buffers. Each sub-buffer contains a distinct texel subset. (c) Different shading computations are performed on every sub-buffer. (d) G-buffer Gathering. The irradiance buffer is computed by gathering all the irradiance sub-buffers. The resulting interleaved pattern may be noticed. (e) A discontinuity buffer is computed. (f) A box blur is applied on the irradiance buffer and the discontinuity buffer is used to prevent non-neighbor texels from being filtered. (g) The self-colors of the objects and the irradiance buffer are finally blended.

5.3.2 G-buffer Splitting (BS)

The initial G-buffer is split into separate sub-buffers. Two look-up textures are first computed. The first one stores the block splitting function presented in Section 5.3.2 while the second one stores the block translation function. Then, two fragment programs successively execute the two functions. Once it is done, the sub-buffers are tiled in a buffer with the same size than the initial one (see Figure 5.4.b).

5.3.3 Shading Computations (SSM / SNSM)

Different lighting contributions are computed for each sub-buffer. Any operation possible with deferred shading is still available. Indeed, as the former G-buffer is explicitly split into smaller sub-buffers, any deferred shader can also be used by focusing the viewport or drawing a quad on a given sub-buffer. A tile of small irradiance sub-buffers is then obtained (see Figure 5.4.c). It may be noticed that the visibility of hemispherical (resp. spherical) point light sources is solved by unrolling the hemicube (resp. the cube) in a standard shadow map [103] and reindexing it with a small cube map as described in [50]. Depending on the application (see Section 5.5), two shading techniques can finally be used: SSM (Shading with a Shadow Map) is a shading pass with a shadow map reprojection and SNSM (Shading with No Shadow Map) is a shading pass with no shadow map reprojection (visibility is ignored). In both cases, glossy and diffuse BRDFs are handled.

5.3.4 Buffer Gathering (BG)

Once all light contributions have been accumulated, the irradiance buffer is rebuilt by two fragment programs which successively perform the two passes of the buffer gathering technique described in Section 5.2.4.

5.3.5 Filtering (F)

To maintain interactive or real time rendering, few light contributions per pixel can be computed. If a filter is applied on continuous zones of the screen, the geometric coherence of the scene can be exploited to virtually compute many light contributions per pixel.

Discontinuity Buffering (DB). A discontinuity buffer is first computed. Two discontinuity thresholds (respectively on normals and positions) are initially fixed.

Then, a fragment program reads the normal and the position buffers and decides if the current pixel is on a discontinuity or not by evaluating an arbitrary measure between the current fragment (x_0, y_0) and its three "positive" neighbors (x_0+1, y_0), (x_0, y_0+1) and (x_0+1, y_0+1). Similar techniques are described with more details by Simmons and Séquin in [74].

Box (Uniform) Blurring (B). The discontinuity buffer is used to apply a two-pass separable box blur on continuous zones of the screen. Two cases can occur.

- The current fragment (x_0, y_0) is not on a discontinuity. The +x (resp. +y) and -x (resp -y) directions are explored one after the other. Neighbor pixels are blended to the current pixel until a discontinuity is encountered in the given direction or the filter kernel size is reached.
- The current fragment (x_0, y_0) is on a discontinuity. Since either (x_0+1, y_0) or (x_0, y_0+1) or (x_0+1, y_0+1) is not similar to (x_0, y_0), only the -x (resp. -y) direction is explored. A neighbor pixel is once again blended to the current pixel until a discontinuity is encountered.

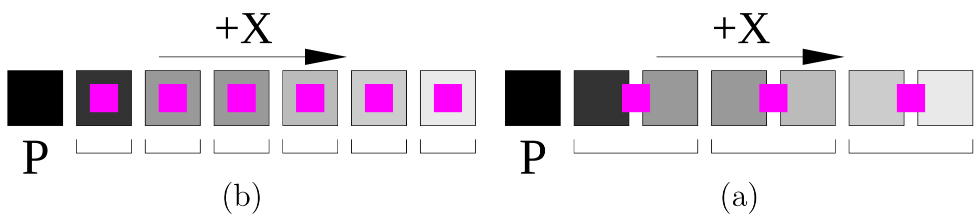


Figure 5.5: **Box Blur.** (a) The standard approach. For a given direction, all texture accesses are made at texel center. (b) With hardware-supported bilinear filtering. All texture accesses are made between two texels. Bilinear filtering blends their contributions. One texture access is therefore sufficient for two texels.

To speed up the filtering pass, we modified the standard box blur by using hardware-supported filtering (see Figure 5.5). Instead of fetching data at the center of each texel, a 0.5 bias is applied and combined with hardware-supported bilinear filtering. As extra artifacts can occur on discontinuities with the fast filtering approach, the accesses to the discontinuity buffer are also modified by activating the 0.5 bias and the bilinear filtering. The texel contributions are then ignored as soon as the discontinuity texel value is greater than 0. In comparison with the first approach, this criteria is *sufficient*.

To prevent object colors from being filtered, the self-colors of the objects are blended only after applying the filter. As shown in Section 5.4, combining discontinuity buffering and interleaved sampling provides high quality filters with very few visible artifacts.

5.3.6 Remarks

Two important remarks may be made. First, the discontinuity buffer is already computed by many deferred shading engines since it helps to perform antialiasing. Secondly, the technique presented in this section can also be considered as a Level-Of-Detail algorithm. If a $2^n \times 2^m$ (low details) interleaved pattern has been computed, $2^i \times 2^j$ interleaved patterns (higher details with $i \leq n$ and $j \leq m$) can also be performed by clustering several sub-buffers. Depending on the application, it is therefore easy to interleave few direct point lights and many secondary ones with only one interleaved pattern.

5.4 Results

We present in this section the results obtained for every rendering pass and the impact of all technical choices presented in the previous sections.

5.4.1 Buffer Splitting Results

We analyzed the performance of the buffer splitting approach with or without block manipulations. Several sizes of blocks were tested. Figure 5.6 presents some results obtained on a GeForce 6800 GT. The performance greatly varies with the size of the blocks and the interleaving sampling pattern chosen. For a 2×2 interleaved sampling, buffer splitting with 32×32 blocks is the most efficient method (only 5% faster than the naive approach). For a 4×4 interleaved sampling, buffer splitting with 16×16 blocks is 70% faster than the one-pass approach. Other tests were made. For larger resolutions such as 1280×1024 and large interleaved sampling patterns, a finely tuned two-pass approach is more than three times as fast as the naive one. Unfortunately, the choice of the block size strongly depends on the organization of the memory and caches on the card. That is why the optimal configuration must often be empirically chosen for each screen resolution and each regular pattern size.

We may finally notice that a recent work using our technique (i.e. buffer splitting and buffer gathering) with GeForce 8 GPUs [56] shows that a direct one-pass splitting was more efficient than the two-pass one. Indeed, the advances recently achieved with graphics hardware make incoherent texture accesses a bit less sensitive to cache problems.

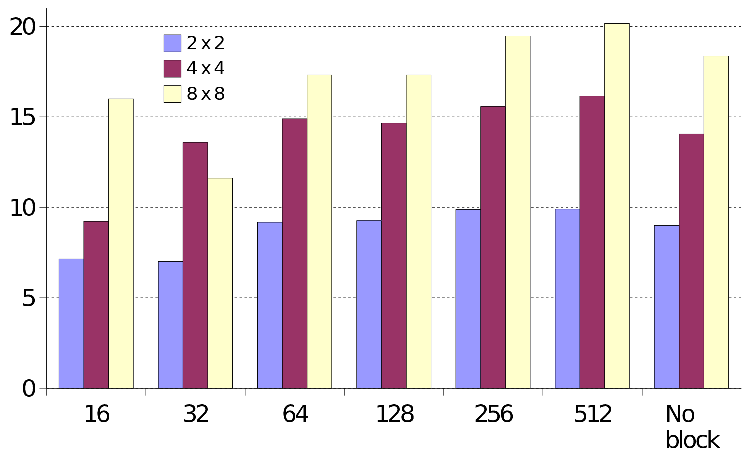


Figure 5.6: **Buffer Splitting Results.** The size of the G-buffer is equal to 1024×1024. The x axis gives the size of the blocks and the y axis gives the computation time in milliseconds. Different interleaved sampling patterns are presented (2×2, 4×4 and 8×8)

5.4.2 Shading Results

The shading performance depends on the type of applications but it is identical to the one obtained with standard deferred shading methods. All details are given in Section 5.5.

5.4.3 Buffer Gathering Results

The performance is quite similar to the one presented in Section 5.4.1 but it may be noticed that only one three-component 16 bit float buffer has to be gathered since only the irradiance buffer is rebuilt. This pass is therefore about 40% faster than the splitting one.

5.4.4 Discontinuity Buffering and Filtering Results

Discontinuity Buffer. Computing discontinuities in a fragment program does not strongly limit our application since it is much less expensive than shading computations. Nevertheless, handling discontinuities during the box blurring is a bit more difficult and expensive.

Filtering. To stop blending texels beyond a discontinuity, accesses to the irra-

	Normal box Blur	Fast box Blur
7×7	16.9 ms	10.9 ms
11×11	29.7 ms	16.9 ms
17×17	51.8 ms	32.7 ms

Table 5.2: **Filtering Results.** Times to perform a box blur on the picture. Two techniques are presented, with and without hardware supported bilinear filtering. The screen resolution is equal to 1280×1024. The GPU used for the tests is a NVidia GeForce 6800GT.

diance buffer are sequentially done along a direction. As soon as a discontinuity is encountered, a flag is set to 0 to ignore the next texel contributions. It may be noticed that dynamic branching can also be used to stop blending texels beyond a discontinuity. This approach was tested and does not provide any significant performance enhancement (because the blurring operation is most of the time performed on the whole kernel).

Table 5.2 shows results obtained with both standard and hardware-supported approaches. Using float buffer filtering capabilities provides a speed-up equal to 60 % without visible differences.

Quality. Figure 5.7 presents some results obtained with sub-sampling and interleaved sampling methods (self-colors of objects are not blended). Even with high-variance estimators due to glossy reflections, interleaved sampling combined with a discontinuity buffer and our fast box blur provides good results. Sub-sampling does not handle high-frequencies details and it is much more difficult to deal with discontinuities. As shown in picture 5.8, details brought by a normal map or a finely tessellated model disappear with sub-sampling whereas interleaved sampling and fast box filtering still provide satisfactory results.

5.5 Applications

After analyzing each computation pass of the algorithm, we now present the overall performance obtained with two different kinds of applications. For both applications, a set of virtual point light sources is first generated with the sampling strategy provided by Instant Radiosity [47]. For more details about this technique, one may refer to Chapter 4. We also present two effective sampling techniques in Chapter 7 and 8.

	BC	BS	SSM	SNSM	BG	B
6800GT	10.9	10.7	4.9	4.7	4.5	7.3
7800GT	5.2	5.8	2.7	2.6	2.3	3.9

Table 5.3: **Overview of the Performance.** The results are given in milliseconds per million of fragments. Times for the G-buffer creation pass are given with a scene of Quake 3 (courtesy of Id Software) counting 250000 triangles. All acronyms are given in Section 5.3. The interleaved pattern size is 4×4 and the kernel size of the box filter is 5×5.

5.5.1 Implementation Details

To handle visibility problems, several classical algorithms were implemented. To speed up the visibility requests, the geometry is first segmented with a kd-tree. Then, frustum culling is performed to eliminate the unseen leaves of the tree. A *PVS* or Potential Visible Set is finally computed for the whole tree to cull leaves not seen by a given leaf. The rasterization code was written using the OpenGL 2.0 API and the ray tracing requests are done by using a finely tuned kd-tree.

5.5.2 Fully Interactive Applications

Using our extension of the deferred shading pipeline with real time applications (like video games) is straightforward but requires specific approximations. An interactive application needs a decent frame rate and computing all visibility requests for every point light is very expensive. The first approximation which has to be done is thus to ignore visibility for secondary light sources. Secondly, the variance is all the larger that the number of bounces increases. To efficiently handle this problem, only one-bounce indirect illumination will be taken into account. It is generally visually sufficient.

Table 5.3 sums up the computation times of all passes with two GPUs of different generations, a NVidia GeForce 6800 GT and a NVidia GeForce 7800 GT. With standard deferred shading, BS, BG and F passes are not executed. Only one or several shading passes are thus performed. It is therefore easy to analyze the extra costs due to interleaved sampling. For a NVidia 6800GT, the extra time taken by the interleaved sampling extension for one million of pixels is equal to 22.5 ms. It is roughly equivalent to 4.5 shading passes. Therefore, if interleaved sampling saves more than 4.5 million shading computations, it becomes competitive. For example, with our implementation and a 4×4 interleaved pattern, computing the contributions of 16 or more lights is already more efficient with interleaved deferred shading. Furthermore, our approach seems to be adapted

	IS	no IS	speed-up
Q3dm11	36 f/s	1.2 f/s	$\times 31$
Dragon in Q3t1	29 f/s	1.2 f/s	$\times 25$
Buddha in Q3dm12	36 f/s	1.2 f/s	$\times 30$

Table 5.4: **Performance with or without Interleaved Sampling on a 7800 GT.** The screen resolution is 1024×768. The interleaved pattern size is 8×6. 480 point light sources are accumulated. With standard deferred shading, the situation is intractable. If it is combined with interleaved sampling, real time frame rate is achieved without too noticeable artifacts.

to current graphics hardware improvements. With roughly the same memory bandwidth, the GeForce 7800 GT is roughly twice faster than the GeForce 6800 GT. Therefore, even with large memory manipulations, the memory latency and bandwidth do not strongly limit our approach and block splitting remains efficient. Table 5.4 gives frame rates obtained with typical scenes used in video games. Without interleaved sampling, real time frame rate cannot be achieved. By extending the rendering pipeline with buffer splitting, the performance is improved by more than an order of magnitude without major visible artifacts as shown in Figure 5.9.

5.5.3 Physically Based Rendering

Physically based rendering requires unbiased approaches. The Instant Radiosity sampling strategy first gives a set of virtual point lights (VPLs). For every VPL, a high quality shadow map is computed. It is then reprojected into a given sub-buffer. As visibility is solved for every VPL, the bottleneck of the algorithm can be either the shading computations or the shadow map computations. Table 5.5 gives numerical results obtained for scenes with different geometric complexities. Hence, due to their linear algorithmic complexities, the shadow maps become very expensive for large scenes. With interleaved sampling, the shading problem can be however efficiently handled. If the point of view and the scene do not change, the G-buffer (split or not) does not have to be recomputed. Therefore, if many VPL contributions are accumulated, the G-buffer creation, the manipulation passes and the filters become free. That is why our approach remains most of the time much faster than deferred shading with a speed-up roughly varying from the interleaved pattern size like 2^2, 4^2 or 8^2 (for very simple scenes) to 1 (for very large scenes). Figure 5.10 gives several images obtained with this approach.

	Shadow Map	No IS	2 x 2 IS	Speed Up
Office	2.2 ms	2.4 s	0.7 s	×3.4
Theater	4 ms	2.8 s	1.0 s	×2.8
Conference	8 ms	3.8 s	2.2 s	×1.7
Cruiser	15 ms	5.9 s	4.0 s	×1.5

Table 5.5: **Times to Obtain less than 1% RMS Error with a NVidia GeForce 6800 GT.** The office contains 31000 triangles, the Candlestick theater 100000 triangles, the conference room 228000 triangles and the cruiser one million triangles.

5.6 Conclusion

In this chapter, we presented an efficient and new way to perform interleaved sampling with today's graphics hardware and a novel and conservative extension of deferred shading. Instead of performing the shading computations with the whole geometric buffer, they are made with small, separate and interleaved sub-buffers. Doing so, the necessary fillrate is strongly limited and hundreds of light sources can be accumulated in real time. By exploiting the pixel coherence, the rendering quality remains very close to the quality obtained with a brute-force approach. Furthermore, all operations available with deferred shading remain available with our method and the algorithm can easily be integrated in any existing real-time rendering package already using a deferred shading technique.

The major step in improving the method is certainly to solve the visibility problems in a better way. Shadow maps like shadow volumes have a linear complexity in relation to the number of triangles. An interesting method would be to perform the shadowing computations by ignoring the visibility during the shading passes but using a set of positive and *negative* virtual point lights. Recently, Laine et al. proposed in [56] to use a shadow map cache system and a clever flickering reduction technique to make the combination of the buffer splitting approach and the use of VPLs physically based and real-time.

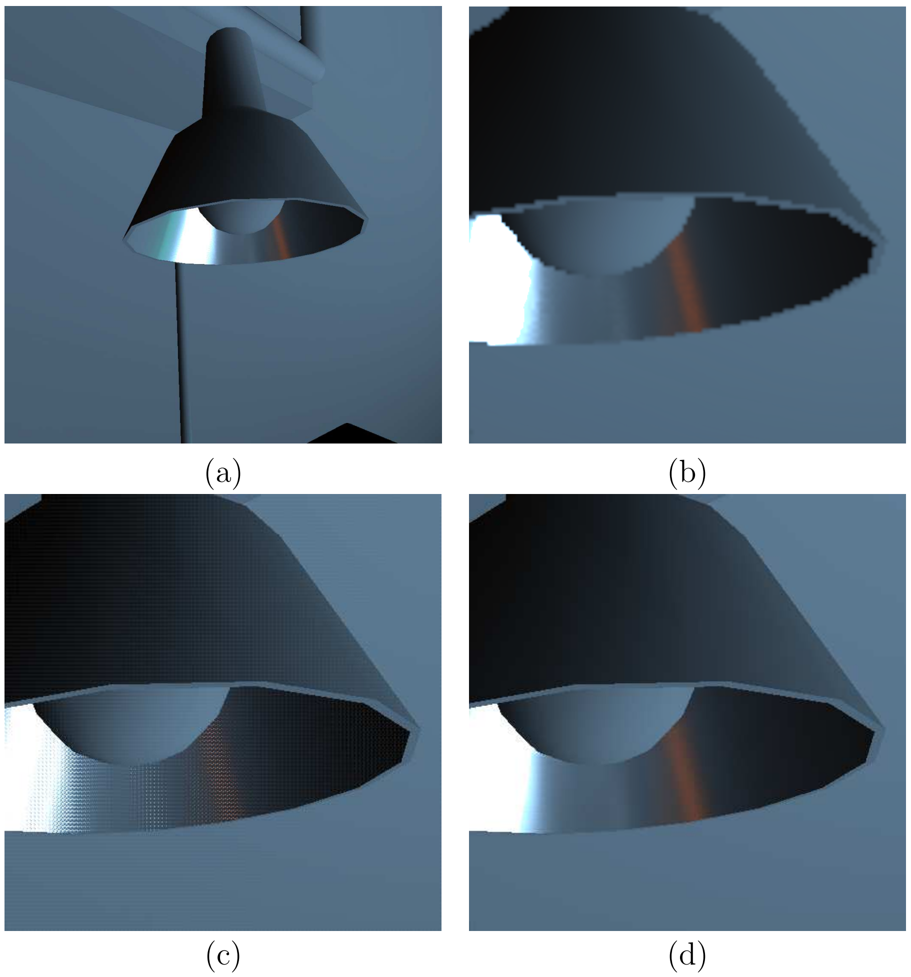

(a) (b) (c) (d)

Figure 5.7: **Rendering Quality.** (a) the 1024×1024 reference image computed with 512 point lights and no interleaved sampling. (b) close-up on a lamp with a simple sub-sampling approach. (c) 4×4 interleaved sampling with no filter. (d) 4×4 interleaved sampling + filtering

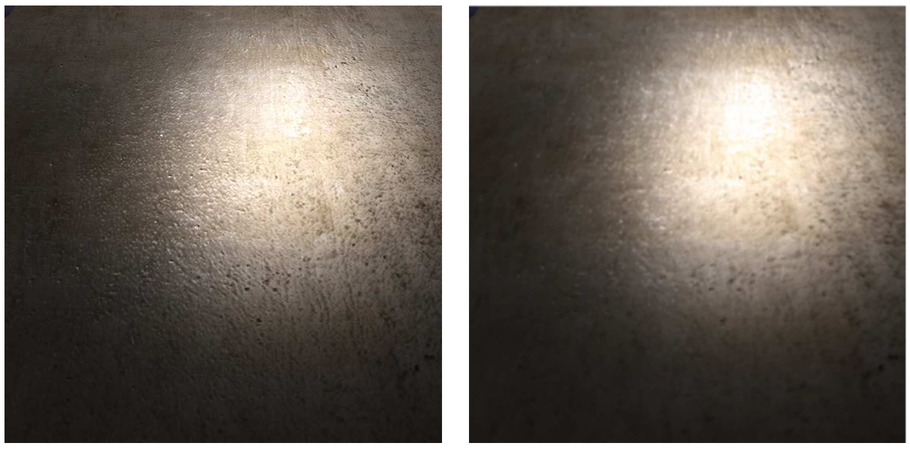
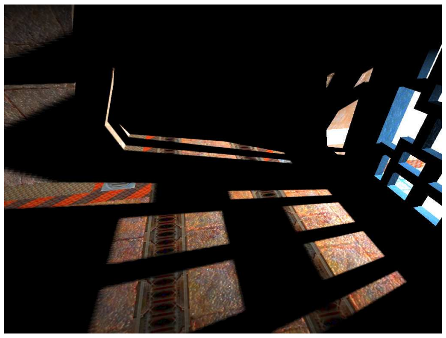
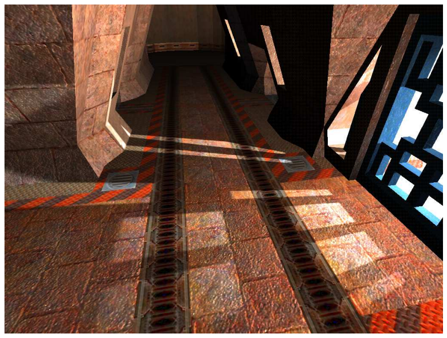
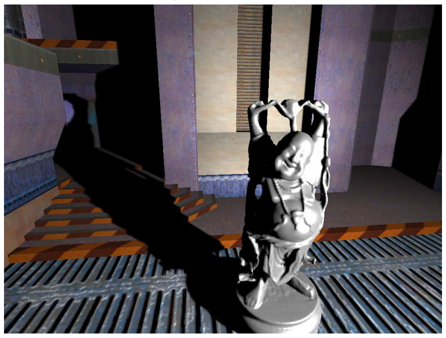
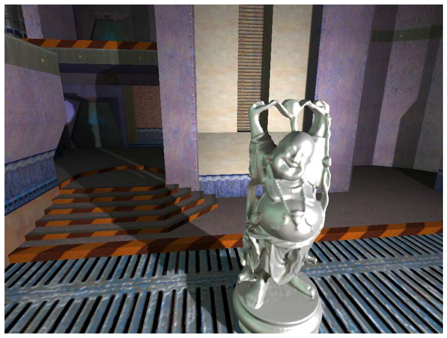

(a) (b)

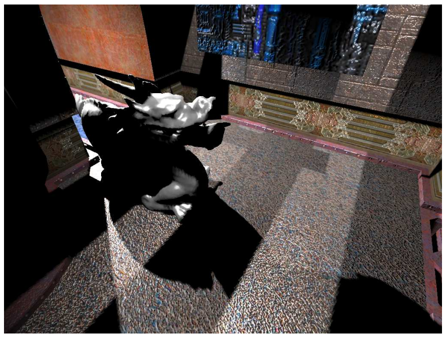
Figure 5.8: The surface details brought by a normal map are conserved with interleaved sampling and our fast box blur as shown in (a). They disappear with sub-sampling as shown in (b).


(a.1) direct only
6800 GT: 35 f/s — 7800 GT: 69 f/s


(a.2) direct + indirect
6800 GT: 19 f/s — 7800 GT: 36 f/s


(b.1) direct only
6800 GT: 32 f/s — 7800 GT: 64 f/s


(b.2) direct + indirect
6800 GT: 15 f/s — 7800 GT: 29 f/s


(c.1) direct only
6800 GT: 31 f/s — 7800 GT: 58 f/s

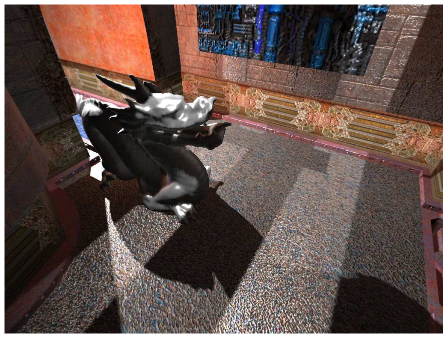
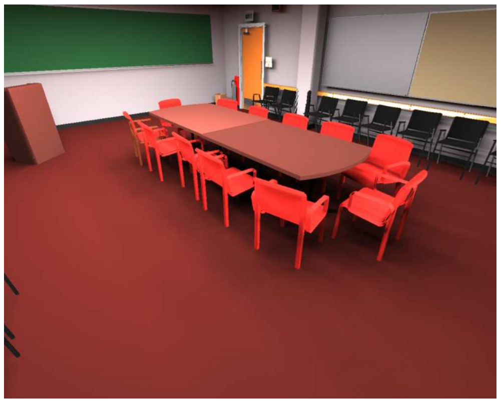

(c.2) direct + indirect
6800 GT: 15 f/s — 7800 GT: 29 f/s

Figure 5.9: **Results with Fully Interactive Applications.** The screen resolution is 1024×768. The interleaved pattern size is 8×6. Visibility for secondary light sources is ignored. All scenes count 480 secondary VPLs. Frame rates obtained for direct contributions only (with a standard deferred shading method) are also given. (a) Q3dm11 (about 80 000 triangles) lit by one hemispherical point light source. (b) Buddha in Q3dm12 lit by one hemispherical point light source. About 200 000 triangles are rendered. (c) Dragon in Q3tourney1 lit by two hemispherical point light sources. About 150 000 triangles are rendered in this scene.

(a) - IS: 0.7 s / No IS: 2.4 s

(b) - IS: 1.0 s / No IS: 2.8 s


(c) - IS: 2.2 s / No IS: 3.8 s

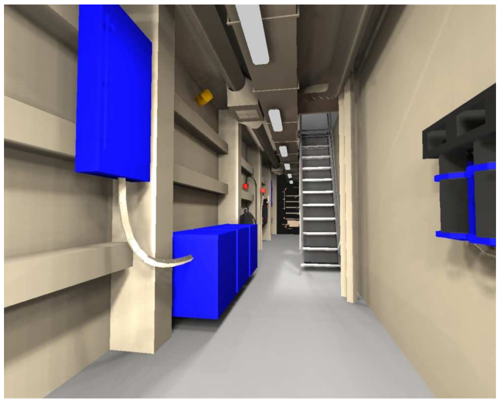

(d) - IS: 4.0 s / No IS: 5.9 s

Figure 5.10: **Physically Based Results.** Pictures and convergence times obtained with less than 1% RMS error. Results are given with or without interleaved sampling. The screen resolution is 1280×1024 and the interleaved sampling pattern is 2×2. (The theoretical maximum speed-up is therefore equal to 4) (a) The office. 35 000 triangles are rendered. (b) The Candlestick Theater with 100 000 triangles. (c) The conference room. 200 000 triangles are rendered. (d) The cruiser. One million triangles are displayed.

CHAPTER 6

Interleaved Deferred Shading

In this chapter, we present a new approach to combine deferred shading and interleaved sampling on modern graphics hardware. The goal is once again to propose a rendering technique using GPUs which aims at speeding up the gathering pass while using Instant Radiosity Algorithm (see Section 4.4 for more details about the method).

The main idea is simply to speed up any rendering application requiring an important use of sampling strategies by computing uncorrelated samples for nearby pixels. To achieve such a result, we simply interleave the sample information in look-up textures and we use these textures during every deferred shading pass. For example, we can compute soft shadows by first storing different sample information inside a $n \times m$ texture and then read this texture to build a different subset of shading results for each pixel inside a $n \times m$ rectangle. To illustrate the efficiency of the method, we propose to speed up two common applications in video games and interactive applications, both using a set of virtual point lights. The first one proposes to use a set of primary virtual point light sources to compute soft shadows due to area light sources in real time while the second one consists in real time rendering of global illumination effects for completely dynamic scenes. For these two applications, our method remains very simple to implement, considerably decreases the consumed fill rate and speeds up the whole rendering process without any noticeable artifacts.

This chapter is organized as follows. Section 6.1 presents our core technique i.e. the generic way to speed up a rendering technique by combining interleaved sampling and deferred shading. Section 6.2 shows how the method can be applied to speed up the generation of approximate soft shadows. Section 6.3 is devoted to the simulation of global illumination effects. A conclusion is finally given in Section 6.4.

6.1 The Interleaved Deferred Shading Technique

Like the previous chapter, the two techniques upon which we base the strategies presented here are interleaved sampling [48] and deferred shading [71]. One may

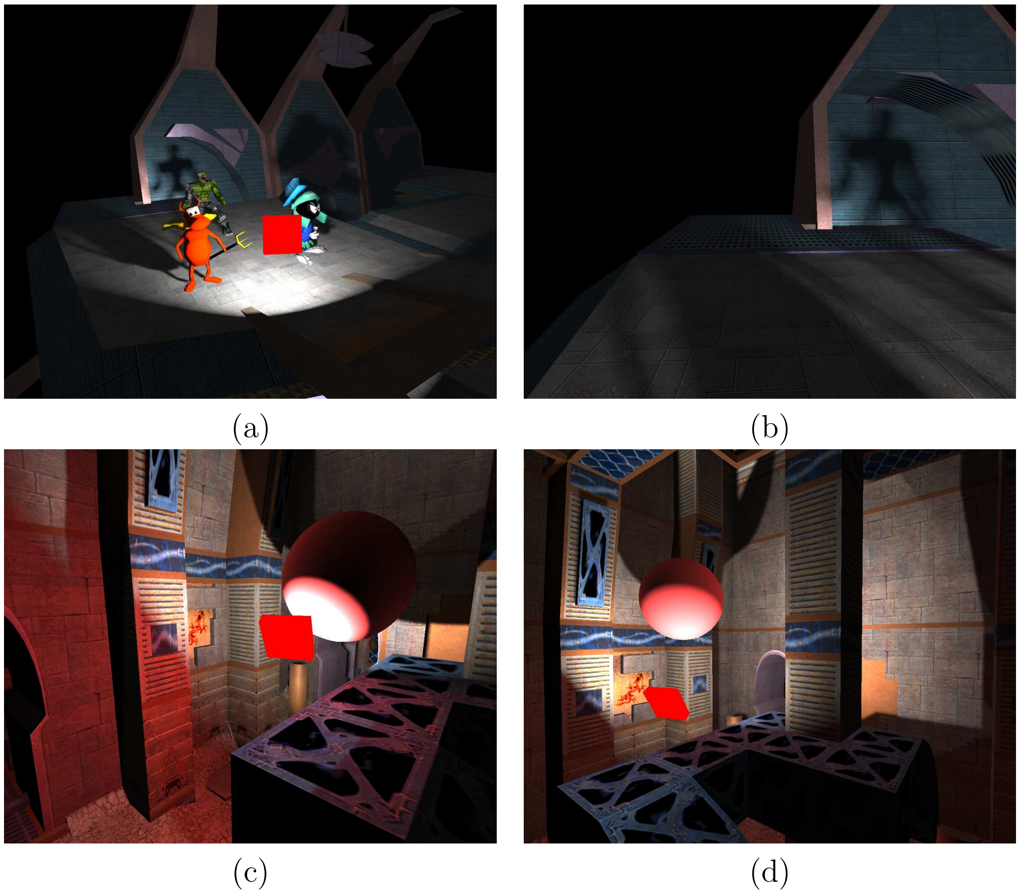


Figure 6.1: **Two Different Applications using Interleaved Deferred Shading.**(a) and (b) show how interleaved deferred shading can speed up the rendering of soft shadows. Thanks to the method, a speed up of 3 and a 35 f/s frame rate can be easily achieved with 256 samples per pixel. (c) and (d) present the results obtained with interleaved deferred shading and a Monte-Carlo global illumination algorithm. The indirect incoming radiance field is first replaced by 500 virtual point light sources. Interleaved deferred shading is then used to display a completely dynamic environment which is both directly and indirectly lit at 40 f/s.

therefore refer to Chapter 5 of this Ph.D. thesis. The goal of our technique is to limit the fill rate consumed by a rendering application. We can for example imagine a Monte-Carlo soft shadow algorithm where 256 samples per pixel are taken to compute the penumbra due to a large light source. To limit the computation requirements, sub-sampling is a common technique: instead of achieving the estimations for all pixels, the calculations can be made for one pixel in 16. Unfortunately, sub-sampling leads to visible artifacts. Conversely, interleaved sampling will propose to take 16 samples for *all* pixels in the screen so that every pixel will have its sample set and filtering the picture will be much easier.

Interleaved deferred shading directly computes interleaved samples without any buffer splitting / gathering passes. The principle is extremely simple since it consists in storing and interleaving the information needed by the shading passes in look-up textures. Therefore, for a $n \times m$ interleaved sample pattern:

Step 1. Store the needed information in one or several $n \times m$ textures;

Step 2. Perform the shading passes by reading the information textures and repeating them across the screen;

Step 3. With the G-buffer, compute a discontinuity buffer and use it to filter the results from Step 2.

The technique is less flexible than the buffer splitting one: interleaving several shadow maps can be for instance difficult since it would require to create a large texture containing all the shadow maps and another one containing the transformation data (such as the Model View / Projection Matrix). Nevertheless, for many applications, it is much simpler, faster and it gives the same results in term of quality.

6.2 Application to Soft Shadows

The first application of interleaved deferred shading is a simple method to generate approximate soft shadows with only one shadow map per light source. For more details about shadow mapping, one may refer to [103].

6.2.1 Principle of the Technique

Our algorithm aims at accelerating Fernando's technique, Percentage Closer Soft Shadows [23], which performs an adaptive percentage closer filter during the shadow mapping pass. The initial Fernando's algorithm is a three-step approach.

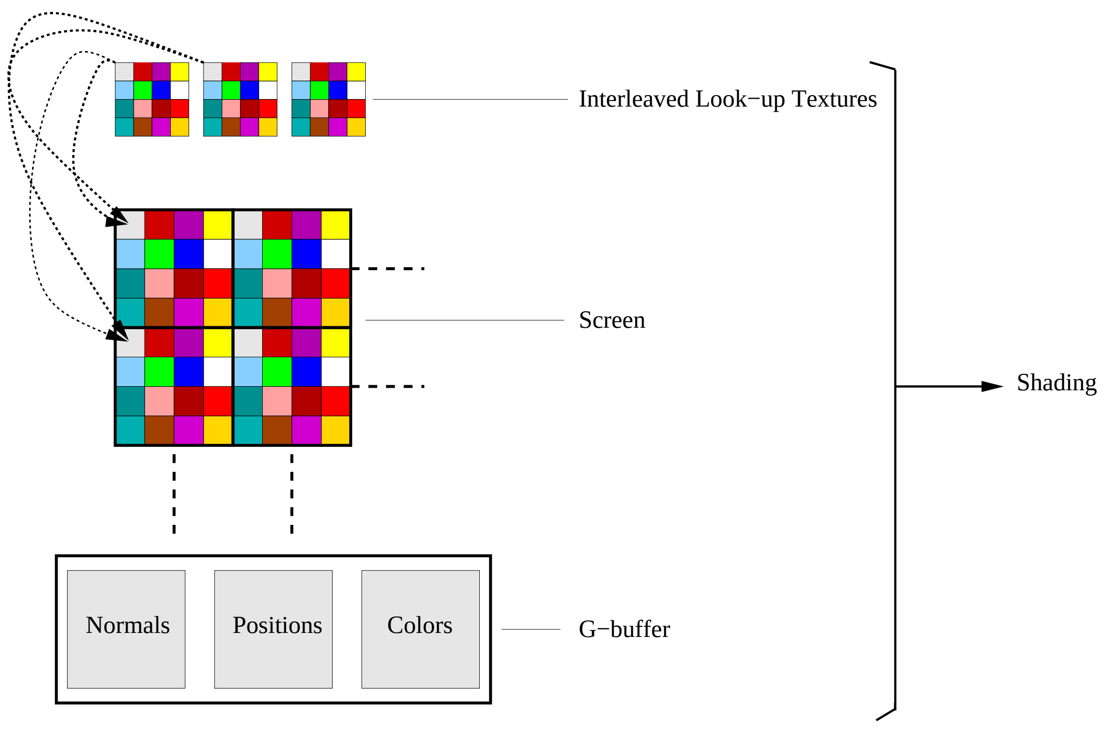


Figure 6.2: **Interleaved Deferred Shading.** Interleaved deferred shading first consists in storing uncorrelated sample information inside small $n \times m$ textures. Then, these samples and the G-buffer are used to perform the shading passes.

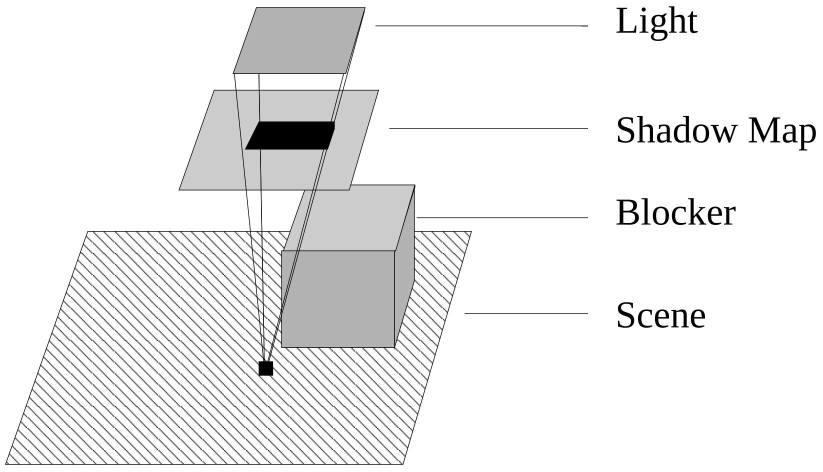


Figure 6.3: **Blocker Search Step.** During the blocker step, we average depth values which are smaller than the light to receiver depth. The search area is shown in black.

Step 1 - Compute a shadow map. Compute one shadow map for an area light source. The projection center is located at the center of the light;

Step 2 - Blocker Search and Penumbra Estimation. Estimate the blocker distance. Search in the shadow map and average the depths that are closer to the point being shaded (see Figure 6.3). Then, using a rough but common parallel plane approximation, evaluate the penumbra width thanks to the estimated blocker distance. Size penumbra w_p is simply given by:

$$w_p \;=\; (d_{receiver} - d_{blocker}) \cdot w_{light}/d_{blocker}$$

Step 3 - Filtering. Perform a standard Percentage Closer Filtering or PCF step (see [67] for more details) using a kernel size proportional to the penumbra estimate from Step 2.

This quite simple technique provides non-physically based soft shadows but visually satisfactory results. Furthermore, even if it is a brute force approach and the resulting frame rate may be low, it is a perfect candidate for interleaved deferred shading since it relies heavily on an important use of sampling strategies.

6.2.2 Percentage Closer Soft Shadows with Interleaved Deferred Shading

The technique is accelerated in a direct way by distributing all the samples inside a $n \times m$ sample pattern. To achieve such a result, we just compute a $n \times m \times l$ 3D sample texture which stores the sample information needed by each pixel inside any $n \times m$ rectangle. By using this texture, we therefore have l samples per pixel and all pixels inside any $n \times m$ rectangle will have their own sets of uncorrelated estimators.

We now have a modified soft shadow algorithm.

Step 0 - Discontinuity Buffer. Compute a discontinuity buffer with the normal and depth information given by the G-buffer;

Step 1 - Blocker Search and Penumbra Estimation. Estimate the penumbra size but only with a smaller sample number. To fetch the sample information needed by pixel (x_i, y_i), read and use the l texels located at the positions going from $(x_i\ mod\ n, y_i\ mod\ m, 0)$ to $(x_i\ mod\ n, y_i\ mod\ m, l-1)$ from the 3D texture;

Step 2 - Penumbra Size Filtering. Filter the penumbra size buffer computed in Step 1 with the discontinuity buffer. Details about discontinuity buffering and fast and accurate filters have been presented in the previous chapter;

Step 3 - Shadow Buffer. Apply a standard PCF filter thanks to the penumbra size buffer computed in Step 2. Once again, use the 3D texture to fetch the sample information so that only l samples are used per pixel;

Step 4 - Shadow Buffer Filtering. Filter the shadow buffer obtained in Step 3 by using the discontinuity buffer.

Once the soft shadows have been computed, the shading passes can be executed to compute the final image.

6.2.3 Results and Discussion

We implemented three versions of the technique using OpenGL 2.0.

- The first one directly uses the algorithm as described by Fernando;
- The second one is the implementation of the method described in Section 6.2.2;

	IDS	Buffer Splitting	Brute Force
A Quake 3 map + 3 Quake 2 models	31 f/s	11 f/s	10 f/s

Table 6.1: **Soft Shadows with several techniques on a NVIDIA 7800 GT.** The screen resolution is 1024×768. The interleaved pattern size is 4×4. 256 samples are used per pixel so that only 16 texture reads are performed with the buffer splitting and the IDS techniques.

- The last one implements the buffer splitting approach. Therefore, we first split the G-buffer and we estimate the blocker distance using only a subset of the needed samples. Then, we gather the resulting sub-buffers, filter them to estimate the blocker distance. We finally split the resulting buffer and repeat the same process one more time to perform the PCF.

As shown by Table 6.1, interleaved deferred shading speeds up the application by a factor of 3 whereas the buffer splitting only provides a 10% speed up. Indeed, the application requires two different sampling steps so that the buffer splitting method needs two extra buffer gathering / splitting passes. Moreover, the quality is quite similar to that of the initial method (see Figure 6.4). Indeed, the discontinuity buffer prevents the colors from bleeding and all high-frequency details provided by bump maps, height maps or a finely tessellated mesh are preserved by decoupling the shading and the shadowing steps in several passes. One might notice that the maximum theoretical speed up (equal in our example to 16) is not achieved due to the overhead in our technique: interleaved deferred shading requires to filter the results of the sampling passes two times and to compute the discontinuity buffer and the G-buffer. Finally, the more physically-based methods presented in [2], [29] and [30] may certainly be accelerated with our technique. Even if they are not Monte-Carlo numerical schemes, it may be quite interesting to reformulate the computations done as sampling strategies and therefore use interleaved deferred shading.

6.3 Application to Global Illumination

The second application of interleaved deferred shading is a simple method to generate global illumination effects in a real time application.

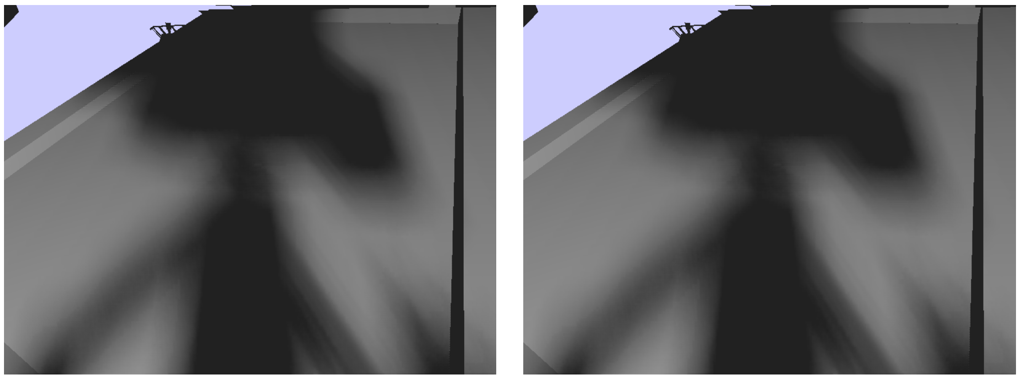

(a) (b)

Figure 6.4: **Quality Differences between the Brute Force and the Interleaved Deferred Shading Approaches.** The achieved quality between the two methods remains close: interleaved deferred shading is suitable for filtering soft shadows.

6.3.1 Principle of the Technique

To efficiently perform global illumination effects with interleaved deferred shading, we first use the Instant Radiosity sampling strategy [47] to find a set of virtual point lights (VPLs) replacing the indirect incoming radiance field. We remind that a complete description of this sampling method can be found in Chapter 4. As solving the visibility requests for every VPL is too expensive, we simply ignore the occlusions between secondary VPLs and a pixel. Therefore, only direct contributions are correctly handled with shadow maps. Finally, as the variance gets larger as the number of bounces increases, only one-bounce indirect illumination will be taken into account.

6.3.2 Instant Radiosity and Interleaved Deferred Shading

Combining interleaved deferred shading and Instant Radiosity is straightforward. We simply store and interleave the VPL normals, positions and powers inside one or more $n \times m$ textures. We then use them during the shading passes so that pixel $(x,\ y)$ will be illuminated by the VPLs located at $(x\ mod\ n, y\ mod\ m)$.

6.3.3 Results and Discussion

We now analyze the performance of our implementation and compare it with similar approaches. We implemented a complete rendering pipeline with the G-buffer creation, the shadow map computations, the shading passes and a filter using a

	IDS	Buffer Splitting	Brute Force
Quake 3 Map - ”Druel” (120K triangles)	44 f/s	36 f/s	1.2 f/s
Quake 3 Map - ”Gon” (30K triangles)	42 f/s	35 f/s	1.2 f/s

Table 6.2: **Global Illumination Effects with different Techniques on a NVIDIA 7800 GT.** The screen resolution is 1024×768. The interleaved pattern size is 8×6. 480 point light sources are accumulated. As interleaved deferred shading (IDS) does not require any massive copy between different buffer organizations, it is more than 20% faster than the previous method. With no interleaved sampling, the situation is intractable

discontinuity buffer. The visibility requests are finally performed through a software ray tracer. As shown by Table 6.2, interleaved deferred shading is about 20% faster than deferred shading with buffer splitting since it does not require any preprocessing or post-processing steps to reorganize the data. Furthermore, the visual quality is strictly equivalent and the implementation is much easier.

Finally, the method can be used with many other typical implementations of global illumination simulation. For example, Dachsbacher and Stamminger use an alternative method to find relevant VPLs from an extended shadow map [15]. As they also use deferred shading, directly interleaving the secondary light sources contributions can be quite appropriate to their technique and will provide a better quality than sub-sampling does.

6.4 Conclusion

In this chapter, we presented an efficient and straightforward way to combine interleaved sampling and deferred shading. The technique consists in storing and interleaving the needed data in textures and using them during the shading passes. We presented two typical rendering applications using it and showed how they can be accelerated without noticeable artifacts. Finally, we believe that the interleaved deferred shading technique can more generally offer a good trade-off between quality and speed when the computing capabilities are insufficient.

CHAPTER 7

Bidirectional Instant Radiosity

This chapter presents a new sampling strategy to achieve interactive global illumination on one commodity computer by proposing an efficient numerical stochastic scheme which can be well adapted to a fast rendering algorithm.

In fact, we want to make Instant Radiosity (IR) [47] more robust and more predictive. As presented in Chapter 4, Instant Radiosity proposes to sample a set of virtual point light sources (VPLs) to describe the incoming radiance field (this is the *sampling* pass) and to use it to illuminate the areas of the scene seen by the camera (this is the *gathering* pass). As we want to provide an efficient sampling strategy to handle difficult settings without sacrificing performance in common cases, we developed an extension of Instant Radiosity in the same way bidirectional path tracing is an extension of path or light tracing. Our idea is to build several estimators and to efficiently combine them to find a set of virtual point light sources which are relevant for the areas of the scene seen by the camera. The resulting algorithm is faster than classical solutions to global illumination. Using today graphics hardware, an interactive frame rate and the convergence of the scheme can be easily obtained in scenes with many light sources, glossy materials or difficult visibility problems. We may notice that this chapter does not explicitly present efficient implementations of the gathering pass: one may actually refer to the presentations of fast GPU techniques in Chapters 5 and 6.

The remainder of the chapter is organized as follows. Section 7.1 presents an overview of the approach. Section 7.2 suggests in details the new sampling strategies we chose to compute global illumination. Section 7.3 details the GPU implementation of our method and all numerical issues which can be encountered. Section 7.4 presents some results. A conclusion is finally given in Section 7.5.

7.1 Overview of the Approach

Bidirectional Instant Radiosity we present here is based on classical Monte-Carlo rendering strategies. Its goal is to compute the measurement integral using Monte-Carlo numerical schemes. Therefore, for sensor j in the scene, Bidirectional Instant Radiosity aims at evaluating Equation 7.1 (as specified in Chapter

4):

$$\begin{aligned} I_j &= \int_{\mathcal{M}\times\mathcal{M}} W_e^{(j)}(\mathbf{x}\rightarrow\mathbf{x}')L(\mathbf{x}\rightarrow\mathbf{x}')\ G(\mathbf{x}\leftrightarrow\mathbf{x}')\ dA(\mathbf{x})\ dA(\mathbf{x}') \\ &= \int_{\Omega} f^{(j)}(\overline{x})d\mu(\overline{x}) \end{aligned} \tag{7.1}$$

Similarly to Instant Radiosity [47], Bidirectional Instant Radiosity proposes to compute global illumination for diffuse or not-too-shiny materials using correlated random variables and factorized computations. Like Instant Radiosity, our method will generate a path random variable $\overline{X}_s$ generated from either the light sources or the camera. Then, if $\mathbf{X_0}$ a random variable defined on $\mathcal{M}_j$ (the surface of sensor j) and $\mathbf{X_1}$ a set of points on the surface of the scene $\mathcal{M}$ (see Figure 7.1.a):

$$I_j = E\left[\frac{f^{(j)}(\{\overline{X}_s, \mathbf{X_1}, \mathbf{X_0}\})}{p(\{\overline{X}_s, \mathbf{X_1}, \mathbf{X_0}\})}\right]$$

As $\{\mathbf{X_1}, \mathbf{X_0}\}$ and $\overline{X}_s$ are independent:

$$p\left[\{\overline{X}_s, \mathbf{X_1}, \mathbf{X_0}\}\right] = p(\overline{X}_s)p\left[\{\mathbf{X_1}, \mathbf{X_0}\}\right]$$

As we show it in the next section, Bidirectional Instant Radiosity builds bidirectional estimators and generates VPLs from the camera and the light sources with a sampling / resampling strategy: a VPL distribution with a density almost proportional to the power they bring to the camera is built. Actually, this kind of approach is not new and similar techniques close to ours, commonly called "importance-driven" sampling strategies, were already used in off-line renderers. We present in Chapter 4 a discussion concerning importance-driven techniques and why their applications to photon sampling are conceptually close to the variance reduction strategies applied to the generation of VPLs. Unfortunately, they do not seem suitable for interactive rendering since many importons (i.e. the particles generated from the camera) have to be generated and stored in a first pass and costly density estimations must be performed for each bounce made by the light paths. Finally, starting from a light source can be simply inappropriate for specific difficult integration cases since a very high number of particles may have to be generated before finding the relevant VPLs. This will be discussed with more details in Section 7.4.

7.2 Bidirectional Sampling of the VPLs

Inspired by the subsequent work achieved by Veach with path integration methods (one may refer to Chapter 4 for more details), we extend Instant Radiosity in the same way bidirectional path tracing is an extension of path or light tracing. Instead of generating VPLs only from the light sources, we propose to generate them from the camera, too. The goal is to provide an efficient sampling strategy suitable for fast rendering engines as those proposed by Wald et al. (see Section 4.4.4) or those we present in Chapters 5 and 6.

7.2.1 Reverse Instant Radiosity

Only generating VPLs from the light sources is as arbitrary as only generating paths from the camera or the lights. Indeed, one strategy can be better or worse than the other according to the integrand: for example, finding caustics is most of the time much easier with light tracing than with camera path tracing. That is why we propose here an adjoint approach to Instant Radiosity: instead of sampling VPLs only from the light sources, they are also generated from the camera.

Sampling Reverse VPLs. Our idea comes from a very simple observation. The only points which can illuminate an area of the scene seen by the camera are the points which can see this area. Therefore, the goal is to sample the VPL locations $\mathbf{x_2}$ along the surface of the scene $\mathcal{M}$ to solve:

$$\begin{aligned} I'_j \;=\; & \int_{\mathcal{M}\times\mathcal{M}\times\mathcal{M}} W_e^{(j)}(\mathbf{x_0}\!\to\!\mathbf{x_1})\ L(\mathbf{x_2}\!\to\!\mathbf{x_1})\ f_r(\mathbf{x_2}\!\to\!\mathbf{x_1}\!\to\!\mathbf{x_0}) \\ & G(\mathbf{x_0}\!\leftrightarrow\!\mathbf{x_1})\ G(\mathbf{x_1}\!\leftrightarrow\!\mathbf{x_2})\ dA(\mathbf{x_0})\ dA(\mathbf{x_1})\ dA(\mathbf{x_2}) \end{aligned} \tag{7.2}$$

With a simple camera model (just a pinhole without lens), the VPLs can be algorithmically found by randomly generating length 2 paths from the camera (see Figure 7.1.b). The end of such a path gives the location of a VPL which can bring light to the camera after one bounce. Therefore, we have a new and simple sampling strategy which consists in finding the ending points of length 2 camera paths. This set will be denoted $\mathcal{M}_r$ (where r stands for "reverse").

We may make an important remark: The density p_c chosen to sample $\mathcal{M}_c$ can be arbitrarily determined as long as we are sure that for all subsets $\Lambda \in \mathcal{M}_c$:

$$\int_\Lambda p_c(\mathbf{x})dA(\mathbf{x}) \neq 0 \text{ if } \Lambda \text{ brings some energy to the camera} \tag{7.3}$$

The simplest sampler to find reverse VPLs may consist in:

- Uniformly sampling the screen (the uniform distribution is not a necessary condition and other ones can be chosen as long as the condition given by Equation 7.3 is satisfied);
- Tracing the associated ray and find the nearest intersection along the ray;
- Sampling a new direction ω from the intersected surface with density $p(\omega)$ (which generally depends on the nature of the surface);
- Finding the nearest intersection along direction ω and considering the resulting point as a reverse VPL.

Once a VPL has been sampled, its properties which are needed to perform the gathering pass have to be computed.

Estimation of VPL Density $p(\mathbf{x_2})$. Each VPL is the end of a length 2 camera path $\overline{x}$. As $\overline{x}$ is a homogeneous Markov chain, its density p is expressed as $p(\overline{x}) = p(\mathbf{x_1})p(\mathbf{x_1} \rightarrow \mathbf{x_2})$ where $\mathbf{x_1}$ is a point directly visible by the camera and $\mathbf{x_2}$ is the location of the VPL. $p(\mathbf{x_1} \rightarrow \mathbf{x_2})$ is therefore the density of $\mathbf{x_2}$ seen by $\mathbf{x_1}$. Hence, computing the density of $\mathbf{x_2}$ consists in evaluating the mean of $p(\mathbf{x_1})p(\mathbf{x_1}\rightarrow\mathbf{x_2})$ by "integrating over $\mathbf{x_1}$".

In other terms, $p(\mathbf{x_2})$ is the probability of reaching this point by a length 2 path. Formally, it is the second marginal law of $\overline{x} = (\mathbf{x_1}, \mathbf{x_2})$ and therefore, if $\mathcal{M}_c$ is the set of the points visible from the camera:

$$p(\mathbf{x_2}) = \int_{\mathcal{M}_c} p(\mathbf{x_1})p(\mathbf{x_1}\rightarrow\mathbf{x_2})dA(\mathbf{x_1})$$

As we can see, evaluating the density of reverse VPLs is an integration problem. Many techniques can be used to solve it and we present here a simple Monte-Carlo numerical scheme. First, it is important to notice that the densities *used to sample the VPLs* (i.e. $p(\mathbf{x_1})$ and $p(\mathbf{x_1}\rightarrow\mathbf{x_2})$) and the densities *used during the Monte-Carlo estimation* may be different.

It is finally crucial to specifically design an estimator. Here, our estimation problem is quite simple since it only consists in sampling points $\mathbf{x_1}$. Therefore, if we sample $\mathbf{X_1}$ with density $p(\mathbf{X_1})$ (i.e. the *same* density that the one used during the reverse VPL sampling step), we have the simple following estimator:

$$U \ = \ p(\mathbf{X_1} \rightarrow \mathbf{x_2})$$

U can finally be reexpressed with directions. If direction ω is equal to the normalized direction $\overrightarrow{\mathbf{X_1 x_2}}^{(u)}$[1], we have:

$$U \;=\; p(\mathbf{X_1} \rightarrow \mathbf{x_2}) \;=\; p(\omega) \cdot \frac{|\omega \cdot \mathbf{N}(\mathbf{x_2})|}{||\mathbf{X_1} - \mathbf{x_2}||^2} \cdot V(\mathbf{X_1}, \mathbf{x_2})$$

where:

$$V(\mathbf{x}, \mathbf{y}) = \begin{cases} 1 \text{ if } \mathbf{x} \text{ is visible from } \mathbf{y} \\ 0 \text{ otherwise} \end{cases}$$

Estimation of VPL Exiting Radiance Field $L(x_2 \rightarrow x_1)$. Once the density of the VPLs has been estimated, their outgoing radiance field must be computed. We can do a rough but common approximation by considering the surface of the VPLs diffuse and their outgoing radiance field constant for all projected solid angles. Otherwise, representing the outgoing radiance field would be much more difficult since it may vary along the hemisphere according to the incoming radiance. To perform the radiance field estimation, we implemented a standard bidirectional sampler as described by Veach and Guibas [86] and Lafortune and Willems [55]. We may also notice that the "inefficient" bidirectional path tracer presented in Chapter 4 can be also used since the efficiency of the VPL sampling pass is less fundamental than the efficiency of the gathering pass.

Once the density and the outgoing radiance field of the generated VPLs have been estimated, we can use them to evaluate Equation 7.2. Hence, we provide a sampling strategy which finds a VPL distribution proportional to the importance brought by the camera.

7.2.2 Bidirectional Instant Radiosity

The major problem with path tracing algorithms is to find relevant paths. With Metropolis Light Transport [88], the ideal random variable with a density proportional to the integrand is directly found by a Metropolis sampling. The integration is finally performed thanks to the useful ergodic properties of the generated Markov chain. Unfortunately, with Instant Radiosity methods, generating an ideal random variable per pixel seems difficult since a part of the paths is already fixed by the VPLs. An easy way to do it would be to have a set of VPLs for each pixel. Unfortunately, all the nice algorithmic properties of the method and its efficiency would be lost.

That is why we propose another sampling strategy. Its goal is to generate a VPL family such as the power brought to the camera by each of them is constant.

[1]We remind that $\overrightarrow{\mathbf{xy}}^{(u)} = \frac{\overrightarrow{\mathbf{xy}}}{||\overrightarrow{\mathbf{xy}}||}$

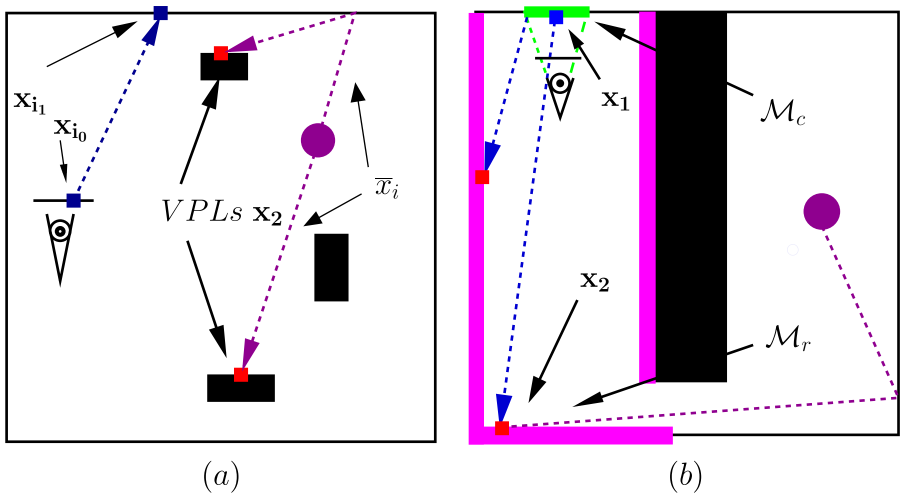


Figure 7.1: **Standard and Reverse Instant Radiosity.** (a) Standard IR. (b) Reverse IR. Green points x_1 ($\mathcal{M}_c$) are the points visible from the camera. Magenta points x_2 ($\mathcal{M}_r$) are those visible from $\mathcal{M}_c$ and the only possible VPL locations for the given point of view. Sampling length 2 paths gives $\mathcal{M}_r$. Connecting $\mathcal{M}_r$ to the light sources gives the outgoing radiance field of the sampled VPLs.

In other words, we want to sample a random variable with a density proportional to the power brought to the camera. In the case where the integrand is the responsivity of the camera to the incoming radiance (i.e. a camera with one pixel), it is the best sampling strategy.

Standard and Reverse IR sampling methods can now provide the location of VPLs. As shown on Figure 7.2, Standard IR is not always better or worse than Reverse IR. According to the radiance field in the scene and the point of view, a method has to be preferred to the other. Therefore, we have to find an efficient way to combine the two sampling strategies.

We propose a Monte-Carlo sampling / resampling approach. In path tracing methods, sampling / resampling is often useless since the most expensive step is the generation of paths. Conversely, IR is a two-pass algorithm with an integration pass generally much more expensive than the propagation one. Therefore, a sampling / resampling method can be quite suitable to generate the VPLs.

First, a set $\mathcal{V}_N$ of $N/2$ standard VPLs and $N/2$ reverse VPLs is generated. For each of them, we evaluate the power brought to the camera (more exactly the camera response to the incident radiance field created by the VPLs). Then, the associated cumulative distribution function (CDF) is built. To estimate the power brought by every VPL, M length 2 paths are cast from the camera to the

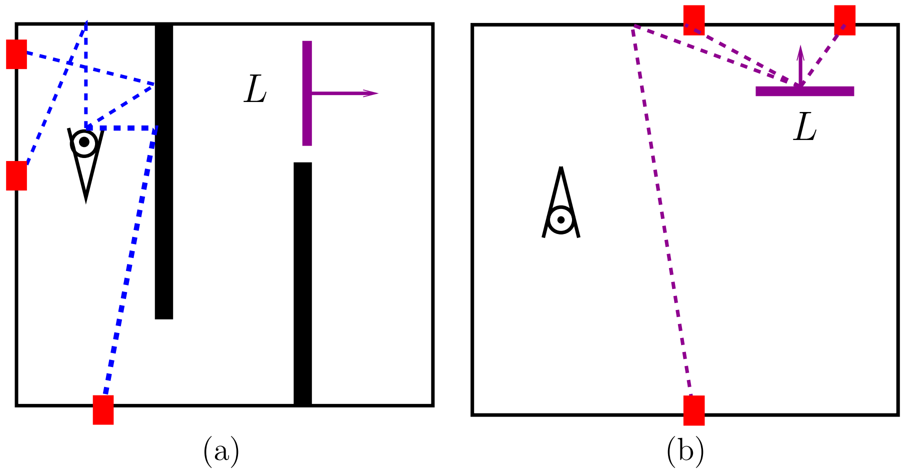


Figure 7.2: **Comparison of Standard Instant Radiosity and Reverse Instant Radiosity.** (a) Here, Reverse IR is more efficient to find relevant VPLs. The scene is indirectly lit and the relevant paths are difficult to find from the lights. (b) Standard IR is more efficient. A light source directly illuminates a very small area of the scene. Finding it from the camera is difficult.

VPL and the energy transfered along these paths is evaluated and accumulated to build each element of the CDF. Then, we use the CDF to resample $\mathcal{V}_N$ and to determine a relevant subset $\mathcal{V}_{N'}$ $(N' \leq N)$. The densities of the resampled VPLs are finally subsequently modified by multiplying their densities by their weights in the CDF and their energies by the number of times they have been chosen.

This method does not call for important storage capacity and remains very simple to implement since it does not require any extra sophisticated structures or non-parametric estimations. It may be noticed that a sampling / resampling strategy has been recently used by Talbot et al. [83] to efficiently sample BRDFs and direct lighting.

7.3 Implementation

The implementation we propose must deal with sampling and performance issues. Two distinct passes are made. The first one generates VPLs while the other one performs the final integration.

7.3.1 Handling Many Light Sources

Before generating VPLs, the problem of sampling many light sources must be solved. Several papers made attempts to deal with this issue. As Dietrich et al. did [17], we build a cumulative density function (CDF) depending on the light source contributions. Paths are first traced from the camera and the contribution of every light source is computed along them. Then, the CDF is built and used to choose without bias which light sources must start a light path. This method simply discards (or likely discards) the sources which barely light the areas seen by the camera.

7.3.2 VPL Sampling

The VPL sampling pass is done with a raytracer. To achieve efficient raytracing with a $\mathcal{O}(\log n)$ complexity, a kd-tree is classically built (with a $\mathcal{O}(n \log n)$ complexity). However, even if our raytracer is carefully written in C and many of the useful heuristics proposed by Havran in his Ph.D. thesis [34] were used, we do not use any low-level optimization for our implementation.

We may notice three important numerical issues during the propagation pass described in Section 7.2.

- The first important issue is the power estimation done for each VPL. M sample paths are generated from the camera. The power brought by each VPL is then estimated using these paths. To prevent a rough estimation from discarding lights with a small contribution, a small uniform quantity is added to each element of the CDF (see Figure 7.3). We may notice that doing this keeps the estimator perfectly unbiased since no element of the CDF which can bring some power to the camera is nil. However, the resampled VPL density is no more proportional to the power they bring to the camera (but it remains close to it). During all our tests, $M = 10$ provided satisfactory results.

- The second numerical issue is to estimate the density of probability of every VPL sampled from the camera. Once again, M sample camera paths are used to perform the estimation. If M is too small, the final result can be biased. If it is too big, the estimation becomes too expensive. $M = 50$ provided good results without noticeable bias. To speed up the computations, we also use a very simple trick. When the power brought to the camera by a reverse VPL is evaluated, only 5 samples are taken into account. Then, the precise density estimation with 50 camera samples is only done for the VPLs which are resampled. Doing so, the quality of the power estimation is a bit degraded but the resampling rate can be much larger for the same computation time. In our tests, generating more VPLs always provides more satisfactory results than performing a precise power estimation with fewer VPLs.

- The last numerical issue is the resampling rate. As the best sampling method is not known, $\frac{N}{2}$ reverse VPLs and $\frac{N}{2}$ standard VPLs are first sampled. Then, N' are resampled and kept. We produced several tests to estimate an efficient value for $r = \frac{N}{N'}$. If r is too large, two many sources are sampled / resampled and the propagation pass becomes too expensive. If it is too small, the integration domain is not sufficiently explored and the convergence will be slow. On most of the scenes, $r = 10$ produces very satisfactory results. With our GPU integrator, the propagation pass remains cheap compared to the integration one. Nevertheless, the value of r can be more accurately tuned depending on the respective performance of the propagation and the integration algorithms. It may be noted that it is not necessary to store all VPLs which will be resampled if the sampling / resampling is done incrementally several times. Therefore, Bidirectional Instant Radiosity does not require an important storage.

7.3.3 GPU Final Integration

To perform the final integration pass, we set up a simple GPU integrator close to the implementation we propose in Chapters 6 and 5. Since we have to accumulate many light sources, it is fundamental to prevent the geometry from being rasterized many times; we thus propose to use deferred shading [71]. We give here some of the main steps of our rendering pipeline:

G-buffer Creation. The normal, the position and the color of each pixel are stored in a G-buffer. To build it, three float buffers are first created. They respectively contain positions, normals and colors. Material information such as material identifiers is also packed in the remaining components. For bandwidth reasons, precision is limited to 16 bits. It may be noticed that the G-buffer does not have to be recomputed if the point of view remains still. It is therefore well adapted to progressive rendering.

Shadow Map Computation. To perform all visibility requests, a shadow map is computed for each VPL. With the approximation that VPLs lie on diffuse surfaces, they can be considered as uniform hemispherical sources. To compute the visibility requests due to a hemispherical light source, the five shadow maps of the hemicube are unrolled in a standard 2D shadow map and reindexed with a small cube map. The method is described in [50]. Due to shadow map aliasing issues, it may be noticed that the GPU implementation is not unbiased. Nevertheless, with high resolution shadow maps and the interior scenes we tested, aliasing is most of the time not noticeable for hemispherical shadow light sources. For exterior scenes and the illumination due to a sky, the parallel projections can be easily reparameterized to avoid aliasing (see, for example, [78] [104]).

Shadowing Operations. Once the shadow map is computed, all the VPL contributions are accumulated using the standard technique of ping-pong buffers. Two buffers are simply created and used. During a computation pass, one buffer is written while the other one is read to fetch and accumulate the previous contributions. Their respective roles are then sequentially switched.

7.4 Results

We now present the results obtained with the Bidirectional IR sampling technique and the GPU implementation presented in Section 7.3.3. Furthermore, we will compare our method to the closest related work.

7.4.1 Reverse and Standard IR

To illustrate the behavior of the sampling strategies, a "U" office only indirectly lit by another room through a small corridor was designed (see Figure 7.4). For this kind of scene, Reverse IR is more efficient since finding the VPLs from the lights is quite difficult. Figure 7.4 presents some results obtained with the two approaches *without* resampling. A second office indirectly lit by a halogen lamp was also built. It is a scene where Standard IR sampling is better than Reverse IR (see Figure 7.6b).

Therefore, depending on the scene, one of the method can be better than the other. To handle all kind of scenes, the two sampling strategies are combined. Table 7.1 gives the percentage of standard and reverse VPLs kept after resampling. It gives a good idea of the quality of the generated samples for the given point of view. Hence, for U-office and the point of view given on Figure 7.4, 100 % of the resampled VPLs were reverse VPLs. Conversely, on the conference room mainly directly lit by many small light sources, Standard IR provides much better light sources than Reverse IR does.

	Standard IR VPLs	Reverse IR VPLs
Conference	8.7 %	1.3 %
Cruiser	6.2 %	3.8 %
U-Office	0%	10 %
Q3tourney1	2.1 %	7.9 %

Table 7.1: **Percentage of Resampled VPLs.** The resampling rate is equal to 1:10. According to the scene and the point of view, one method or the other provides the more relevant VPLs. For the conference room scene, most resampled VPLs are standard VPLs. For U-Office, the Reverse IR strategy is more efficient.

7.4.2 Sampling / Resampling Performance

To handle all visibility difficulties, the VPLs are resampled. Figure 7.5 gives an image of the "U" office obtained by combining and resampling the two estimators. 1250 VPLs are created with Standard IR sampling strategies and 1250 other ones are generated with Reverse IR technique. Then, all of them are resampled with a resampling rate r equal to 1:10.

Sampling / resampling gives satisfactory results in all the tested cases. Figure 7.6 presents other images computed with VPLs obtained using the sampling strategies described in Section 7.2. Scene Q3tourney1 is particularly awkward. Light must bounce at least three times before reaching the central room through a small corridor.

7.4.3 GPU and Overall Performance

We now analyze the performance of our GPU integrator. The goal is to present the performance impact of our method compared to Standard Instant Radiosity and other techniques dealing with importance and bidirectional sampling. We only want to show that Bidirectional Instant Radiosity remains suitable for interactive rendering. Table 7.2 sums up the performance of our raytracer and the different GPU passes. An interactive frame rate is thus obtained with few VPLs for scenes with variable complexities (Office contains 35,000 triangles whereas Cruiser contains one million triangles). Compared to coherent raytracing approaches [92], our implementation suffers from the linear algorithmic complexity of shadow map computations. Nevertheless, with a commodity GPU and a simple frustum culling on kd-tree leaves, interactive global illumination can be easily performed. Several pictures with the scenes we tested are presented on Figure 7.7.

We now analyze the impact of Bidirectional Instant Radiosity on the frame

	Shadow Map	G-Buffer	Shading	ray/s	fps
Office	2.2 ms	8.3 ms	5.1 ms	1000K	15
Theater	4 ms	9.2 ms	5.1 ms	900K	6
Conf	8 ms	12.2 ms	5.1 ms	800K	4.5
Cruiser	15 ms	31 ms	5.1 ms	500K	2.2

Table 7.2: **Performance of our Implementation.** The given times for the GPU passes are obtained with $512 \times 512 \times 4$ unrolled hemicube shadow maps and a 1024×1024 screen resolution. Frames per second given in the Table are obtained with 30 VPLs / frame. The GPU is an Nvidia GeForce 6800 GT and the CPU an AMD Athlon 2400+.

rate in comparison with Standard Instant Radiosity. For a 1:10 resampling rate, each VPL requires about 10 rays (by using 5 rays for the power estimation, 5 rays for the fast reverse estimation of the VPL density of probability and 50 rays for the precise density estimation). With the standard Instant Radiosity method and a mean albedo equal to 0.5, one VPL requires about 1 ray. Therefore, our sampling strategy is ten times slower than Instant Radiosity. However, even with 1000 VPLs per frame and 100 frame per second, an optimized raytracer can almost provide ten million rays as required by our method. Furthermore, VPLs can be reused from one frame to the next one. Therefore, Bidirectional IR makes Instant Radiosity suitable for a wider variety of cases without sacrificing much of its fine properties (fast sampling strategies and fast integration).

7.4.4 Comparison with Importance-Driven Techniques

Another way to extend Instant Radiosity would be to use the importance-driven photon sampling strategy designed by [62]. With this technique, light sources and directions are chosen by estimating the importance brought by the camera. Therefore, it would be easy to sample VPLs proportionally to the power they brought to the camera. Unfortunately, this method does not seem suitable for interactive or real time rendering. First, large scenes can require a large number of importons (more than 20000). Secondly, choosing a new shooting direction is very expensive since it requires the construction of a hemispherical CDF.

Keller's and Wald's method [49] is faster since no CDF is computed every time a new direction is chosen. The importance estimation is only used to determine if a photon is stored or not. Nevertheless, a lot of importons have to be stored and the technique does not provide a new direction sampling strategy as Peter's technique does. The method was nevertheless tested with fast estimations with a regular grid. The quality of the resulted images was equivalent to Standard IR + sampling / resampling but provides much less satisfactory results than Bidirectional IR. This can be explained by the completely new sampling strategy brought by Reverse IR. The difference was particularly noticeable for the U-office scene where a large number of particles had to be generated from the physical light sources before finding relevant VPLs.

Therefore, even if standard importance-driven techniques provide very good results for off-line rendering, Bidirectional Instant Radiosity outperforms them when relevant VPLs have to be found with the constraint of interactivity.

7.5 Conclusion

We presented in this chapter new sampling strategies to handle difficult scenes for global illumination. Our method extends Instant Radiosity and proposes a bidirectional sampler to find relevant VPLs for a given point of view in a fast and efficient way. Using GPUs and a simple and brute force implementation, we already perform interactive rendering with not-too-shiny materials and all kinds of visibility layouts. In the next chapter, we will explore other sampling strategies using Metropolis VPL samplers. The idea consists in building an ergodic homogeneous Markov Chain of VPLs with an invariant law proportional to the power brought to the camera by the VPLs.

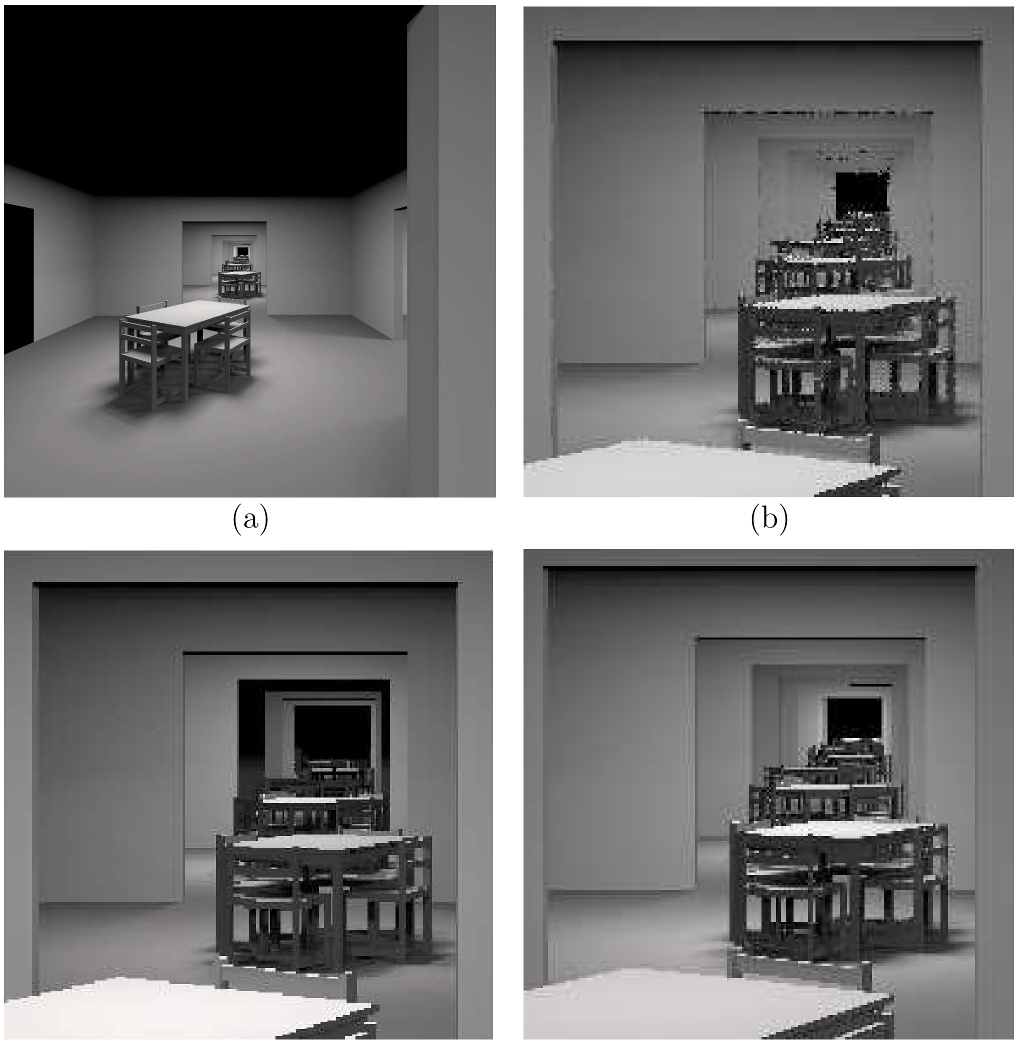
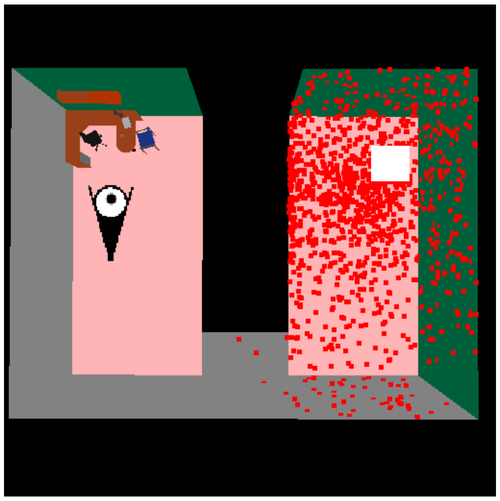
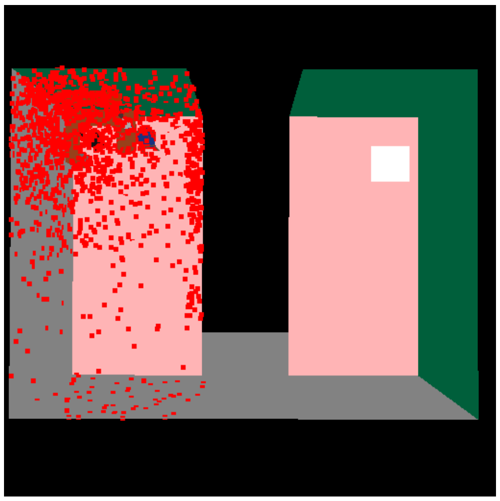
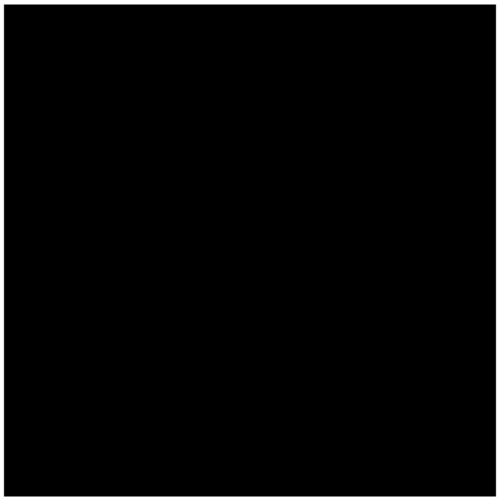

(a) (b) (c) (d)

Figure 7.3: **Handling the Bias Problem with Sampling / Resampling.** (a) shows direct lighting for the 10th Shirley's test scene. (b) is a zoom on a small part of the screen computed by Radiance. (c) The same zoom done with a too rough CDF and our GPU integrator. Power brought by remote VPLs is underestimated. (d) A small quantity is added to all the elements of the CDF. The estimator is now unbiased.


(a)


(b)


(c)

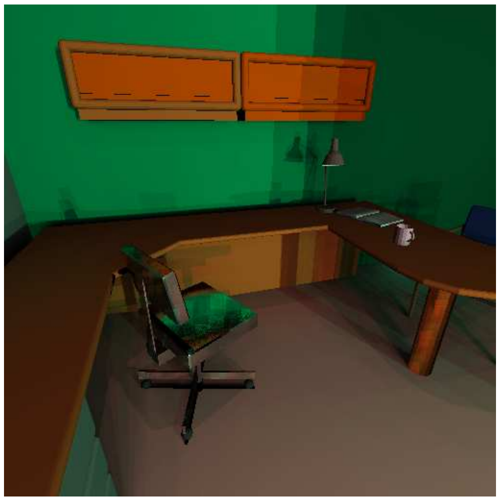
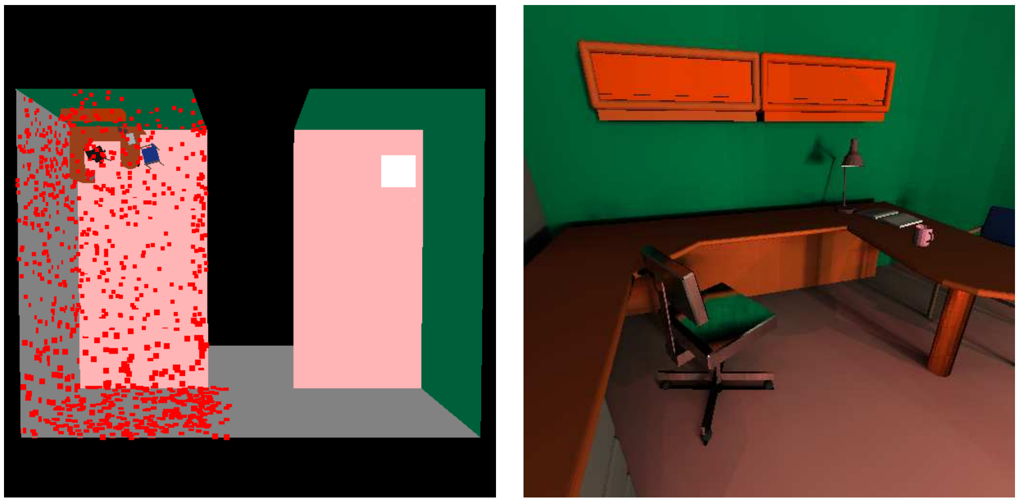
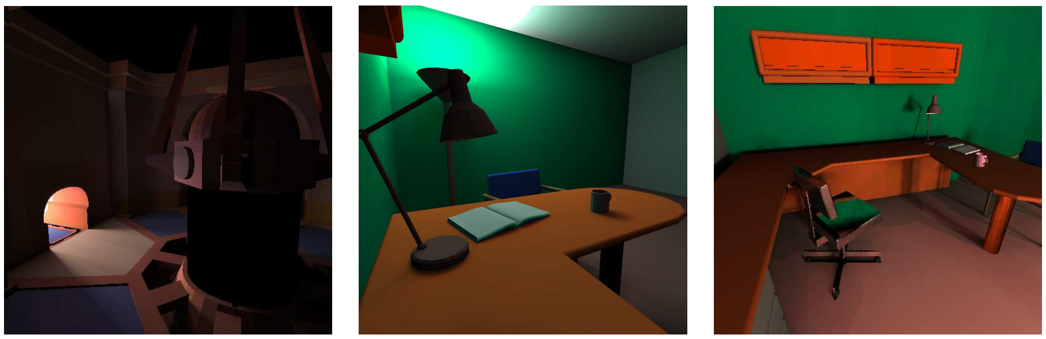

(d)

Figure 7.4: **The "U" Office.** (a) shows the VPLs sampled with Standard IR sampling. (c) is the resulting picture. No VPL illuminates the areas seen by the camera. The picture is black (b) shows the VPLs sampled with Reverse IR sampling. (d) the resulting raytraced picture with only 200 VPLs and *no resampling*.



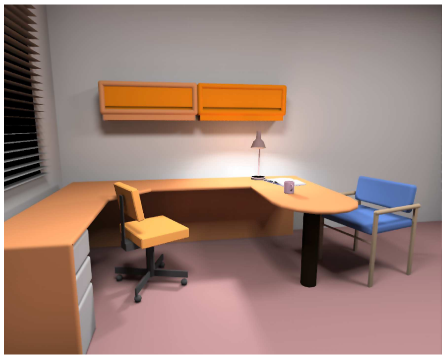
(a) (b)

Figure 7.5: **Bidirectional Instant Radiosity with the "U" Office.** Sampling / resampling creates a distribution proportional to the power brought to the camera. (a) shows the VPL distribution and (b), the raytraced image obtained with only 250 VPLs.

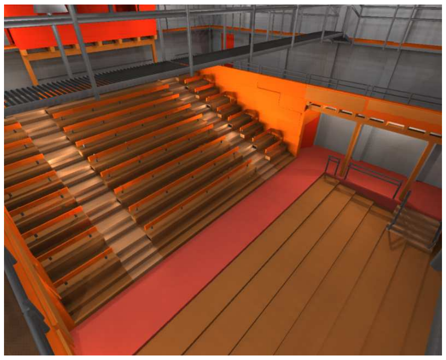


(a) (b) (c)

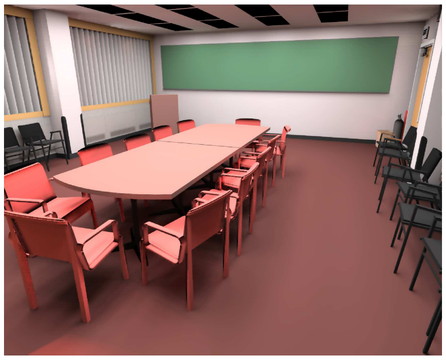
Figure 7.6: **Sampling Performance.** Several pictures computed with the sampling techniques described in the chapter. Bidirectional sampling finds the relevant VPLs no matter the visibility layout. (a) A part of Q3tourney1 (courtesy of ID software) indirectly lit trough a small corridor. Light paths make at least 3 bounces before coming into the room. Most of the VPL are found with reverse IR strategy. (b) A simple office indirectly lit by a halogen lamp easily handled by Standard Instant Radiosity. (c) The U Office (raytraced). The scene is completely indirectly lit through a small corridor. Bidirectional IR quickly finds the relevant VPLs.



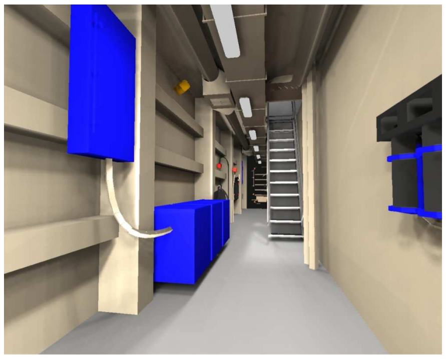
(a)
2.4 s / 15 f/s



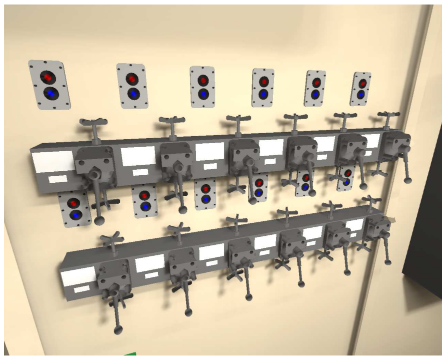
(b)
2.8 s / 6 f/s



(c)
3.8 s / 4.5 f/s



(d.1)
5.9 s / 2.2 f/s



(d.2)
4.6 s / 2.2 f/s

Figure 7.7: **Overall Performance with a GPU Deferred Shading Renderer.** With our method, convergence is obtained in a few seconds for very different scenes. Convergence times are obtained with less than 1% RMS error. Frame rates are obtained with 30 VPLs per second. The screen resolution is equal to 1280×1024. The graphics card is a NVidia GeForce 6800 GT. (a) An office with a very bright lamp. 35,000 triangles are rendered. (b) Candlestick Theater with 100,000 triangles. (c) The conference room. 200,000 triangles are rendered. (d) Two different views of Cruiser. One million triangles are displayed.

CHAPTER 8

Metropolis Instant Radiosity

In this chapter, we present Metropolis Instant Radiosity (MIR), an unbiased algorithm to solve the light transport problem.

MIR is actually an extension of Instant Radiosity (see Chapter 4 for more details about this technique): as Instant Radiosity, it is a hybrid technique which consists in representing the incoming radiance field by a set of Virtual Point Lights (VPLs) and in computing the response of all sensors in the scene by accumulating their contributions. In contrast to other similar approaches and more particularly Bidirectional Instant Radiosity we presented in the previous chapter, MIR proposes to sample the VPLs with an innovative Multiple-try Metropolis-Hastings (MTMH) algorithm: the goal is to build an efficient, aggressive, and unconditionally robust variance reduction method that works well regardless of the scene layout. Indeed, the quality of the sample set is fundamental: as shown, for example, in Figure 8.6.f, placing VPLs close to the light source will provide a bad solution, since the parts of the scene seen by the camera are not illuminated by the regions close to the source. Conversely, Metropolis Instant Radiosity presented here, will compute a VPL set which has an interesting intrinsic property: each VPL will provide the same amount of power to the camera. Figure 8.6 illustrates the efficiency of our approach with very different situations. Finally, we present a fast ray tracing implementation using MIR and show how our complete rendering pipeline can produce high-quality and high-resolution pictures in few seconds.

The remainder of the chapter is finally organized as follows. Section 8.1 presents an overview of our contribution. Section 8.2 reviews the two techniques upon which we base our method. Sections 8.3 and 8.4 expose in details the new sampling strategies we set up to compute global illumination. In Section 8.5, we present the results we obtained with our new sampler and some comparisons with related approaches. The limitations and a conclusion are given in Section 8.6.

8.1 Overview of our Contribution

As presented in Chapter 4, Veach and Guibas developed Metropolis Light Transport (MLT) [88] to overcome most of the variance issues involved by independent path sampling strategies. The decisive advantage of a Metropolis sampler over independent Monte-Carlo estimators is its ability to exploit coherence in path space and therefore to preserve the sampling context. Since 1997, the Metropolis-Hastings algorithm has been widely explored. Pauly et al. [61] extended it by adding extra Monte-Carlo Markov Chain (MCMC) mutations that handle participating media. Kelemen et al. [46] proposed a simplification of the MLT algorithm which increases the acceptance rate and directly works in the space of uniform random numbers used to build up paths. Fan et al. [22] also used a Metropolis-Hastings algorithm to populate photon maps by exploiting coherence among light paths. More recently, Cline et al. [11] developed an efficient algorithm that uses Metropolis mutation strategies in a standard Monte-Carlo integrator. They first generate a set of path samples from the camera to the light sources and then use a sequence of MCMC mutations to redistribute the power of each path over the image plane in an unbiased way. All these techniques therefore build low variance estimators and try to directly solve the problem in its high-dimensional aspect. They are unfortunately slow as they do not generally exploit the computation coherence or the current CPU / GPU architectures.

Conversely, another large class of Monte-Carlo rendering techniques focuses on the algorithmic speed rather than on an aggressive variance reduction like Photon Mapping [40] or Instant Radiosity [47] (see Chapter 4 for more details). One may also refer to Wald's PhD [90] for discussions and effective ray tracing implementations of these approaches.

All the above problems therefore motivated Metropolis Instant Radiosity (presented in Algorithm 6). As we want to propose a rendering technique which remains numerically robust and fast for all kinds of scenes, combining a Metropolis sampler which can provide very relevant samples and Instant Radiosity which can be very efficiently implemented sounds good. In this chapter, we therefore present an innovative VPL sampler using a modified Metropolis-Hastings: the "Multiple-try Metropolis-Hastings Algorithm" (MTMH) [41]. As we will show it, our method provides a faster exploration of the sampled space than any other related technique does, and finally offers very good estimators without severe performance penalties.

Algorithm 6 Metropolis Instant Radiosity

```
1: Set all pixel intensities to 0;
2: Compute the power $P_c$ received by the camera (see Section 8.3.1);
3: With a Metropolis-Hastings sampler (either the standard version presented
   in Section 8.3.2 or the Multiple-try one presented in Section 8.4), compute
   a set of $n$ VPLs with a density *proportional to the power they bring to the
   camera*. We do *not* know the outgoing radiance functions of the VPLs but we
   know the scene transmits the same amount of the VPL power to the camera;
4: for $i = 1$ to $n$ do
5:     • Suppose that VPL $i$ is on a diffuse surface and that it has a constant
         outgoing radiance function equal to 1. Compute the intensity of each
         pixel in the screen and the total power $P'$ transmitted to the camera
         through the scene from VPL $i$;
       • As we know that VPL $i$ transmits a power equal to $P_c/n$ to the camera
         and that there is a linear relation between the outgoing radiance
         function of the VPL and the transmitted power, rescale the intensities
         of the pixels by a $\frac{P_c}{n\ P'}$ factor (see Section 8.3.4);
       • Accumulate VPL $i$ contribution.
6: end for
```

8.2 Metropolis Sampling for Light Transport

We give here a short overview of the Metropolis-Hastings algorithm and its application to the global illumination problem as introduced by Veach and Guibas [88].

8.2.1 Metropolis-Hastings (MH) Algorithm

We first recall that a sequence of random variables $(X^{(t)})_{t \in N}$ is a Markov Chain if $X^{(t)}$ depends only on $X^{(t-1)}$ through a transition function $g(\cdot|x^{(t-1)})$. The goal of the Metropolis-Hastings algorithm is to construct a Markov Chain that has a equilibrium distribution π_∞ by applying successive mutations on its elements. This algorithm does not solve *a priori* an integration problem but may provide a very elegant variance reduction technique in the case where many correlated integrals have to be computed.

The algorithm starts at $t = 0$ with the selection of $X^{(0)} = x^{(0)}$ randomly drawn from a distribution π_0 with the only requirement that $\pi_0(x^{(0)}) > 0$. Given $X^{(t)} = x^{(t)}$, the algorithm computes $X^{(t+1)}$ as follows:

1. Sample a candidate value X^* from a proposal distribution $g(\cdot|x^{(t)})$;

2. Calculate the Metropolis-Hastings ratio $R(x^{(t)}, x^*)$, where:

$$R(u, v) = \frac{\pi_\infty(v) \cdot g(u|v)}{\pi_\infty(u) \cdot g(v|u)};$$

3. Sample a value for $X^{(t+1)}$ according to the following:

$$X^{(t+1)} = \begin{cases} X^* & \text{with probability } \min\{R, 1\} \\ x^{(t)} & \text{otherwise;} \end{cases}$$

It is possible to show that under general conditions, the sequence $(X^{(t)})_{t\in N}$ is a Markov Chain with equilibrium distribution π_∞.

8.2.2 Ergodicity

With the MH sampler, we can therefore sample almost any distribution π_∞. If we ensure the *ergodic* property of the chain (i.e. all states are equally probable according to π_∞ after a long time passed in the chain), we are furthermore able to use *all* the samples of the Markov Chain as if they exactly follow the stationary distribution. To do this, it is sufficient to ensure that $g(x|y) > 0$ when $\pi_\infty(x) > 0$ and $\pi_\infty(y) > 0$ since all states can be reached with a non-null probability through only one mutation step.

8.2.3 Application to Light Transport

Veach and Guibas proposed to use a MH sampler as a powerful variance reduction technique for the global illumination problem. They first evaluate the total power received by the camera and then use a Metropolis sampler to compute correlated random variables with a density directly proportional to the integrand $f^{(c)}$ which is actually the set of *all* camera sensors. During the sampling process, they finally estimate the pixel intensities by counting the number of paths going through each pixel and by proportionally distributing the total power over all of them. For a more detailed introduction to Metropolis sampling and its application to rendering, we refer to [63].

8.2.4 Using the Path Integral Formulation for Instant Radiosity

Instant Radiosity, as presented in Chapter 4 is an elegant method to compute global illumination for diffuse or not-too-shiny materials. It simply consists in generating light paths from the light sources and storing each bounce as a Virtual

Point Light or VPL. As in the remainder of this chapter, we will explicitly combine the path integral formulation and the VPL samplers, we think that it is important to remind that each VPL is actually a complete light path and not only its ending point. Therefore, if $\mathbf{x_v}$ is the last point of the light path and $\overline{x}_s$ is the remainder of the path which is connected to a light source, we will say that $\mathbf{x_v}$ is a *geometric* VPL and that $\{\mathbf{x_v}, \overline{x}_s\}$ is a *path* VPL (see Figure 8.1).

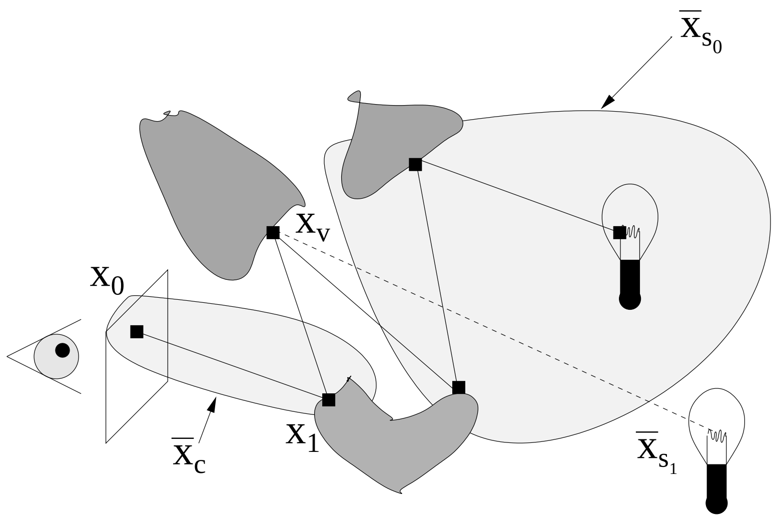


Figure 8.1: **Path and Geometric VPLs:** the camera path, $\overline{x}_c = \{\mathbf{x_0}, \mathbf{x_1}\}$, the location of the VPL, $\mathbf{x_v}$, and $\overline{x}_{s_0}$, the remainder of the path which "brings some power" to $\mathbf{x_v}$. We have here a geometric VPL $\mathbf{x_v}$ which contains two path VPLs, $\{\mathbf{x_v}, \overline{x}_{s_0}\}$ and $\{\mathbf{x_v}, \overline{x}_{s_1}\}$ (see Section 8.3.3).

8.3 Metropolis Instant Radiosity

We introduce here the core of our contribution, Metropolis Instant Radiosity (see Algorithm 6). To be more precise, our goal is to propose an efficient global illumination algorithm by computing a Markov-Chain of VPLs such that each VPL will bring the same amount of power to the camera.

As our algorithm heavily relies on marginals, we first recall their definition. If $X = (X_0, X_1 \cdots X_{n-1}) \in (\Omega_0 \times \Omega_1 \times \cdots \times \Omega_{n-1})$ is a random variable, then X_i is called marginal of X. Furthermore, if f is the density of X, the density f_i of X_i

is defined by:

$$\begin{aligned} f_i(y) &= \int_{\Omega_0} \cdots \int_{\Omega_{i-1}} \int_{\Omega_{i+1}} \cdots \int_{\Omega_{n-1}} f(x_0, \cdots, x_{i-1}, y, x_{i+1}, \cdots, x_{n-1}) \\ & \quad d\mu(x_0) \cdots d\mu(x_{i-1}) d\mu(x_{i+1}) \cdots d\mu(x_{n-1}) \end{aligned} \tag{8.1}$$

We therefore have to integrate f on $\prod_{j \neq i} \Omega_j$ to obtain f_i.

8.3.1 Compute the Power P_c Received by the Camera

We first compute the power P_c received by the camera by generating a family of bidirectional paths going from the camera to the light sources. To eliminate start-up bias, we also resample the generated paths to provide a "good" initial random variable with a law close to the integrand $f^{(c)}$ described in the next section. For more details about this technique, one may refer to [88].

8.3.2 Generating the VPLs with a Metropolis Sampler

The goal of a Monte-Carlo integrator for the light transport problem is to integrate $f^{(j)}$ as shown in Chapter 4 by Equation 4.4. To achieve this goal, the best sampling strategy is to generate samples with a density directly proportional to $f^{(j)}$. What we propose here is thus to sample the entire path space Ω proportionally to the response $f^{(c)}$ of the camera to the power brought by a path, to project these samples on the appropriate sub-space and to finally consider them as VPLs.

By directly using the Metropolis sampling technique designed by Veach and Guibas, we are able to sample a distribution of paths proportional to $f^{(c)}$. Indeed, as indicated in Section 8.3.1, we first simulate a random variable X_0 with a density close to $f^{(c)}$ and then, we use the mutation strategies proposed by Veach and Guibas to compute a Markov-Chain with an invariant law proportional to $f^{(c)}$ (see Section 8.5.1 for more details about the implementation). It is important to notice that these paths are complete since they go from the camera to a physical light source. In the remainder of the chapter, we will finally set the normalization constant a such as: $a = \frac{1}{P_c} = \frac{1}{\int_\Omega f^{(c)}(\overline{x}) d\mu(\overline{x})}$

As the sub-path $\{\mathbf{x_v}, \overline{x}_s\}$ is a marginal of $\overline{x} = \{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_v}, \overline{x}_s\}$ and as we directly sample $\overline{x}$ with density $a \cdot f^{(c)}(\overline{x})$, Equation 8.1 gives us the density $p_{vs}^{(c)}(\{\mathbf{x_v}, \overline{x}_s\})$ of $\{\mathbf{x_v}, \overline{x}_s\}$:

$$a \cdot f_{vs}^{(c)}(\{\mathbf{x_v}, \overline{x}_s\}) = \int_{\mathcal{M}} \int_{\mathcal{M}} a \cdot f^{(c)}(\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_v}, \overline{x}_s\}) \; dA(\mathbf{x_0}) \; dA(\mathbf{x_1})$$

In other words, by sampling complete paths from the camera to a light source with a density proportional to $f^{(c)}$, we also have an interesting class of sub-paths $\{\mathbf{x_v}, \overline{x}_s\}$ with a density proportional to $f_{vs}^{(c)}$. Actually, they all bring *the same amount of power to the camera.* Indeed, if $(\{\mathbf{x_{v_i}}, \overline{x}_{s_i}\})_{i\in[1\ldots n]}$ is a set of n VPLs, then:

$$\begin{array}{rcl} P_c & = & \int_\Omega f^{(c)}(\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_v}, \overline{x}_s\}) d\mu\{\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_v}, \overline{x}_s\} \\ & \simeq & \frac{1}{n}\sum_{i=1}^{n} \frac{\int_\mathcal{M}\int_\mathcal{M} f^{(c)}(\{\mathbf{x_0},\mathbf{x_1},\mathbf{x_{v_i}},\overline{x}_{s_i}\})dA(\mathbf{x_0})dA(\mathbf{x_1})}{p(\{\mathbf{x_{v_i}},\overline{x}_{s_i}\})} \\ & \simeq & \frac{1}{n}\sum_{i=1}^{n} \underbrace{\frac{f_{vs}^{(c)}(\{\mathbf{x_{v_i}}, \overline{x}_{s_i}\})}{p_{vs}^{(c)}(\{\mathbf{x_{v_i}}, \overline{x}_{s_i}\})}}_{\text{VPL } i \text{ contribution}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{a\cdot n} = \frac{1}{n}\sum_{i=1}^{n}\frac{P_c}{n} \end{array}$$

Additionally, we may remark that even if we know that every VPL brings the same amount of power to the camera (equal to P_c/n if there are n VPLs), the outgoing radiance function of each of them is *unknown* (but not needed as we know P_c).

As soon as the sampling step is finished, we finally have a set of sub-paths $(\{\mathbf{x_{v_i}}, \overline{x}_{s_i}\})_i$: each of them is a *path* VPL which "emits" light from $\mathbf{x_{v_i}}$ and transmits a power equal to P_c/n to the camera.

8.3.3 Clustering the Physical VPLs

Each VPL is thus a complete sub-path which starts from a light source and goes to the corresponding geometric VPL $\mathbf{x_v}$. While applying mutations on complete paths, it is furthermore possible that the VPL location $\mathbf{x_v}$ does not change. This case occurs when:

- The candidate is rejected and the path is duplicated;
- Only the sub-path $\overline{x}_c = \{\mathbf{x_0}, \mathbf{x_1}\}$ is mutated;
- Only the sub-path $\overline{x}_s$ is mutated.

With this sampler, each geometric VPL can therefore contain several path VPLs: if k path VPLs go to the given location $\mathbf{x_v}$, the contribution of the *geometric* VPL $\mathbf{x_v}$ will be equal to $k \cdot P_c/n$. Finally, if the user requires m different locations for the VPLs (i.e. m geometric VPLs), the number n of path VPLs that must be generated is not known but is automatically determined during the sampling step by precisely monitoring the mutations.

8.3.4 Accumulating the VPL Contributions

After the sampling step, we have a set of m path VPLs $\mathbf{x_{v_i}}$: each of them brings a fixed amount of power to the camera equal to $P_i = k_i \cdot P_c/n$ where n is the number of paths generated during the sampling step (i.e. the total number of path VPLs) and k_i is the number of path VPLs connected to $\mathbf{x_{v_i}}$. To compute the contribution P_i of VPL $\mathbf{x_{v_i}}$ for each pixel of the screen, we simply dispatch P_i among all pixels. To achieve such a result, we first suppose that the surface at $\mathbf{x_{v_i}}$ is diffuse and that its outgoing radiance function is constant and equal to 1. Then, we perform the lighting computations and evaluate the intensity of every pixel. Once it is done, we evaluate the total power P'_i received by the camera and scale all pixel intensities by a P_i/P'_i factor such that the total power emitted by $\mathbf{x_{v_i}}$ and transmitted to the camera becomes P_i.

8.3.5 MIR with Common Renderers

Metropolis Instant Radiosity is conceptually different from Instant Radiosity since we do not know the outgoing radiance function of each VPL. This may be a practical limitation since most of the implementations of Instant Radiosity assume that this function is known and therefore base their code-design on this assumption. Fortunately, our sampling technique can be easily integrated to any of these renderers by adding an extra pass: the VPL outgoing radiance function estimation. This pass simply consists in randomly casting rays from the camera and then, in scaling their outgoing radiance function in relation to the power they bring to the camera. Thus, we do not have to extend any pre-existing renderer using Instant Radiosity since all the properties of the VPLs (normals, powers, and positions) are determined.

8.3.6 Handling Overmodulations due to the $1/r^2$ Term

As indicated in Section 4.4.2, a major problem with Instant Radiosity (and Metropolis Instant Radiosity) is that overmodulation issues occur when the $VPLs$ are close to the surface currently shaded. To handle this problem, we propose a consistent strategy: as usually done, we first avoid the overmodulation problem by bounding the visibility term $G(\mathbf{x_v} \leftrightarrow \mathbf{x_1})$ during the shading passes *and the sampling passes*. Once the VPL contributions have been accumulated, we therefore have a *biased* result where a part of the total incoming energy is missing. To compute this missing part, we are however able to use the technique proposed by Kollig and Keller in [51] which consists in adding an extra pass which exactly compensates the bias introduced by bounding $G(\mathbf{x_v} \leftrightarrow \mathbf{x_1})$. We may notice that we did *not* implement this last pass and all the pictures presented in the article are therefore biased.

8.4 A VPL Multiple-try Metropolis-Hastings (MTMH) Sampler

In the previous section, we described a complete rendering pipeline using a standard Metropolis-Hastings (MH) sampler. The main problem with such a sampler is the important correlation which may occur between successive samples in the chain: in worst cases, the algorithm may be slow to converge and it may be trapped in a local mode of integrand f (see Figures 8.2 and 8.3). To overcome these difficulties, Liu et al. [41] proposed an alternative strategy known as Multiple-try Metropolis-Hastings sampling. What we propose here is to slightly change Step 3 of MIR by replacing the MH algorithm by the Multiple-try one.

8.4.1 The MTMH Algorithm

The approach is to generate a larger number of candidates thereby improving the exploration of π_∞ near x. One of these proposals is then selected in a manner that ensures that the chain has the correct limiting stationary distribution. To achieve such a result, we still use a proposal distribution g, with optional positive weights $\lambda(u,v)$ where the *symmetric* function λ is presented further below. To ensure the correct limiting stationary distribution, it is necessary to require that $g(x^*|x^{(t)}) > 0$ if and only if $g(x^{(t)}|x^*) > 0$, and that $\lambda(x^{(t)}, x^*) > 0$ whenever $g(x^*|x^{(t)}) > 0$. Let $x^{(0)}$ denote the starting value, and define $w(u,v) = \pi_\infty(v)g(u|v)\lambda(u,v)$. Then, for $t \in N$, the algorithm proceeds as follows:

1. Sample p independent proposals $X_1^* \dots X_p^*$ from $g(\cdot|x^{(t)})$;
2. Randomly select a single proposal X_j^* from the set of proposals, with probability proportional to $w(x^{(t)}, X_j^*)$ for $j = 1, \dots, p$;
3. Given $X_j^* = x_j^*$, sample $p-1$ independent random variables $X_1^{**}, \dots X_{p-1}^{**}$ from the proposal density $g(\cdot|x_j^*)$. Set $X_p^{**} = x^{(t)}$;
4. Compute the generalized Metropolis-Hastings ratio:

$$R_g = \frac{\sum_{k=1}^p w(x^{(t)}, X_k^*)}{\sum_{k=1}^p w(X_j^*, X_k^{**})};$$

5. Set

$$X^{(t+1)} = \begin{cases} X_j^* & \text{with probability } \min\{1,\ R_g\} \\ x^{(t)} & \text{otherwise.} \end{cases}$$

We can give an intuitive explanation to understand the MTMH algorithm. With a standard Metropolis-Hastings sampler, we test two samples, x and x^*, and keep only one of them with the respective probabilities 1-$\min(1, R)$ and $\min(1, R)$. With MTMH, we conversely test two families of samples, $(x_1^* \dots x_p^*)$ and $(x_1^{**} \dots x_p^{**})$, and keep only one element of each family x_j^* or $x_i = x_p^{**}$ with the respective probabilities 1-$\min(1, R_g)$ and $\min(1, R_g)$: instead of only testing two points, we finally also deal with their "Metropolis neighborhood".

8.4.2 Application to VPL Sampling

As the limiting properties do not change with a MTMH sampler, the generation of complete paths $\overline{x}$ will provide the sub-path class $\{x_v, \overline{x}_s\}$ that has the same properties as those obtained with a standard Metropolis-Hastings sampler. We set $\lambda(u, v) = [g(u|v) \cdot g(v|u)]^{-1}$ to encourage certain types of proposals: by using this specific λ, $w(x_i, x^*)$ corresponds to the importance weight $\pi_\infty(x^*)/g(x^*|x_i)$ and the chosen candidate X_j^* becomes the probably most interesting sample among $X_1^* \dots X_p^*$. The decisive advantage of MTMH sampler over a Metropolis-Hastings one is algorithmic. Since the VPL generation step is not the most expensive one in a complete rendering pipeline, we have some computation time to finely tune the VPL distribution: MTMH is therefore quite appropriate to achieve this goal since it tends to decorrelate the successive VPLs and to explore the whole integration space faster. For a classical Metropolis Light Transport (MLT) implementation, this approach may be however much less interesting since generating $2k-1$ extra paths without using them may be inefficient. Nevertheless, we may notice that if we find a way to use all of them, MTMH may also provide an aggressive variance reduction technique in a MLT environment.

8.5 Implementation and Results

We present in this section how we have implemented our complete renderer, the results we obtained with MIR, and finally, several comparisons with existing similar approaches.

8.5.1 The VPL Sampling Pass

The first thing to do is to virtualize and replace the complete incoming radiance field by a set of virtual point lights computed with our Metropolis sampler. As described in the previous sections, we generate these VPLs by mutating and projecting light paths which go from the camera to a light source. Since our method is limited to diffuse and not-too-shiny environments, we do not deal with caustics and therefore only use bidirectional perturbations: for a complete and detailed explanation of the technique, we refer to [88].

8.5.2 Rendering with Coherent Ray Tracing

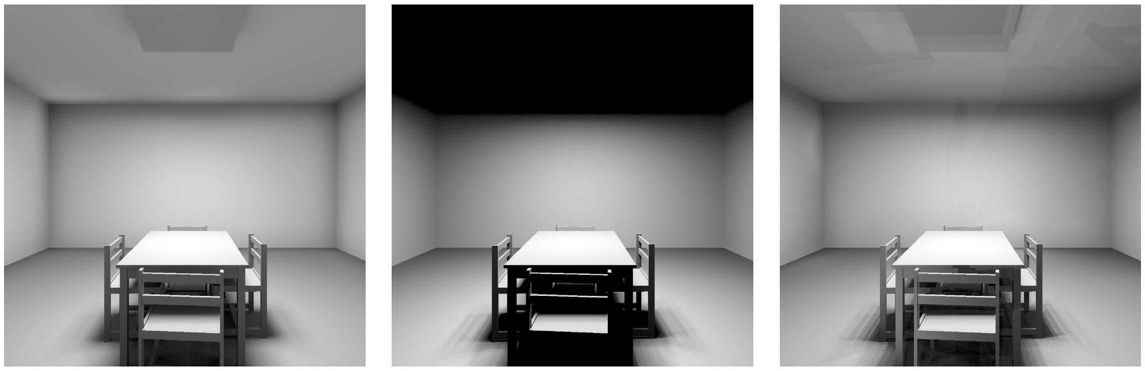
The core of our implementation relies on coherent ray tracing algorithms and implements the OpenRT API [17]. To render a picture, we therefore use the now classical rendering technique: "Instant Global Illumination" presented in [97]. First, we perform the ray casting requests with a finely tuned coherent ray tracer using the SIMD SSE* instruction sets today available on almost every commodity PC. This approach simply consists in packing several rays inside one vectorized structure and performing all the operations by using SSE operands. Then, we use Interleaved Sampling [48] to accumulate distinct VPL contributions for every pixel inside a $n \times m$ tile. Once these contributions have been accumulated, we finally filter the resulting picture inside the continuous zones of the screen with a discontinuity buffer, thereby virtually providing $n \times m$ times more samples per pixel. More details about these techniques and the construction of efficient acceleration structures for ray tracing can be found in the literature [4,34,90,95].

8.5.3 MTMH vs MH

We present here some simple configurations to show why it may be attractive to use Multiple-try Metropolis-Hastings rather than Metropolis-Hastings. In this section and the remainder of the chapter, we will use 10 candidates for MTMH, a commonly used value in computational statistics.

As shown in Figure 8.2.b, the MH sampler gets stuck in the direct local mode. Even if it will finally find the indirect contributions, the large correlation between successive samples will produce a "non-representative" sample set if only a small number of VPLs is used. Conversely, MTMH provides much better results as shown in Figure 8.2.c: with very few VPLs, it proposes a good sample distribution with a direct/indirect repartition close to the reference one.

Figure 8.3 presents another kind of configuration where we explicitly create two important local modes: the parts of the scene seen by the camera can be


(a) - 12800 VPLs (b) - MH (c) - MTMH

	Direct	Indirect	RMS error
Ground Truth	49%	51 %	-
MH (128 VPLs)	100 %	0 %	0.04%
MTMH (128 VPLs)	56 %	44 %	0.005%

Figure 8.2: **Direct / Indirect Modes.** (a) is the reference image. (b) shows the results obtained with a MH sampler: the sampler gets stuck in the direct local mode. (c) shows the results obtained with a MTMH sampler: it is able to explore direct and indirect local modes.

illuminated either by the left room or by the right one. As expected, with only 256 VPLs, the MH sampler does not equally explore the two local modes whereas the MTMH algorithm provides much better results by computing a more representative sample set. In the remainder of the chapter, we will therefore always use MTMH instead of MH.

8.5.4 Results with Easy Configurations

We present here the results we obtained with Metropolis Instant Radiosity in "easy" scenes, i.e. scenes which are mostly directly lit. We compare our approach with two other methods, Bidirectional Instant Radiosity (see Chapter 7) and {Standard Instant Radiosity + Efficient Light Source Cumulative Distribution Function (CDF)} [91]. The first method consists in generating a larger number of VPLs than desired from the camera or the light sources, in computing the power they transfer to the camera and finally, in keeping the most relevant candidates (i.e. those which bring the larger contributions). The second method consists in computing the power brought by each *physical* light source to the camera through direct and indirect contributions and in associating to each of them the corresponding density. As shown in Figure 8.4, a standard sampler without any variance reduction technique gives poor results. On the contrary,

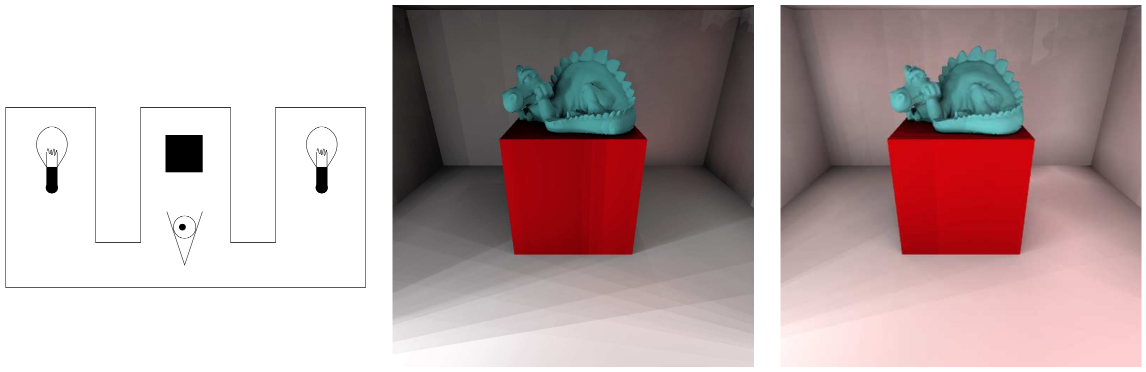

(a) (b) - MH (c) - MTMH

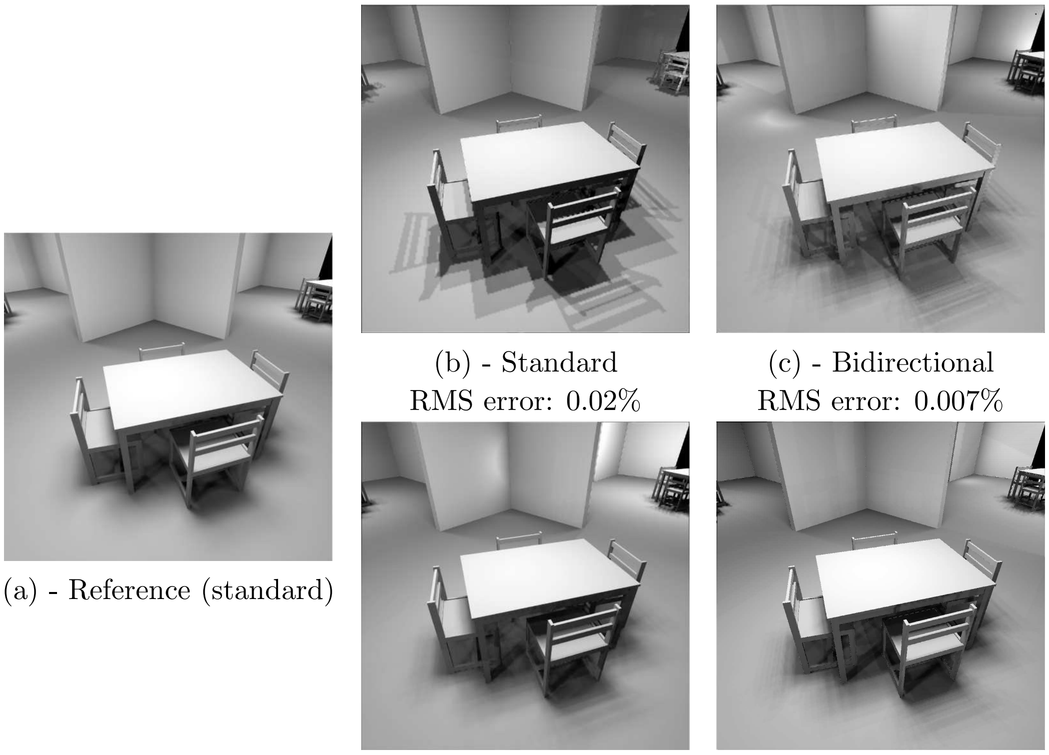
Figure 8.3: **Exploration of Left and Right Contributions.** (a) is the overview of the scene: the two lights emit the same amount of energy. (b) shows an image computed with 256 VPLs and a MH sampler. (c) is the same scene with 256 samples, the same computation time, but computed with a MTMH sampler.

	with CDF	without CDF
Office	50 %	54%
Shirley's Scene 10	54 %	91%

Table 8.1: **Rejection Rate with MTMH**

the three other methods achieve much more satisfactory renderings with very comparable qualities. This can be easily explained by the fact that the three approaches try to generate a VPL distribution with a density close to the power brought to the camera. Since the scene is mostly directly illuminated, Wald's et al. approach approximates it by first computing a suitable CDF for the physical light sources. The resampling method tries to compute it by discarding the less interesting ones. Metropolis Instant Radiosity finally approximates it by directly simulating the desired density through a Markov-Chain.

We may also remark that the three approaches are complementary. Indeed, the MTMH sampler chooses the best candidate with a technique close to a sampling / resampling strategy. Furthermore, associating to each source a density proportional to the power they bring to the camera, remains interesting with a MTMH sampler since it decreases the rejection rate during the sampling process (see Table 8.1) and accelerates the VPLs generation. This combination is more efficient if we want to generate a large number of VPLs or if the environment is highly occluded.



(a) - Reference (standard)

(b) - Standard
RMS error: 0.02%

(c) - Bidirectional
RMS error: 0.007%

(d) - CDF
RMS error: 0.008%

(e) - MTMH
RMS error: 0.009%

Figure 8.4: **Tests with Shirley's Scene 10.** The reference image is computed with 12800 VPLs, the other ones, with 256 VPLs.

8.5.5 Results with Difficult Visibility Configurations

As presented above, MIR is efficient for scenes which are mostly directly lit. However, the decisive advantage of our strategy is its ability to handle very hard visibility issues. To prove it, we compared our approach to Standard Instant Radiosity (SIR) and Bidirectional Instant Radiosity (BIR) by testing the three techniques with two awkward configurations presented in Figures 8.6.f and 8.3.a. To be fair, we finally ensure that the VPL generation time with BIR is close to the VPL generation time with MIR.

As shown in Figures 8.5.a, 8.5.b and 8.5.c, Standard Instant Radiosity fails to find the relevant VPLs since it does not discard the direct contributions. With BIR, the result is much better but an important noise is still noticeable. With MIR, we finally achieve an excellent result. The difference between the two approaches can be intuitively explained: BIR proposes to build a sample distribution with a density proportional to the power brought to the camera in an approximate

	Scene 6	Scene 10	Office	Conf	Theater	Cruiser	3 Dragons
VPL generation time (MTMH)	0.32s	0.31s	0.31s	0.49s	1.0s	0.92s	2.4s
VPL generation time (BIR)	0.82s	0.82s	0.62s	0.92s	1.0s	1.3s	1.0s
Rendering time	4.5s	5.0s	3.2s	7.4s	11.1s	7.7s	7.1s

Table 8.2: **Computation Times on a Core Duo T2600.** The interleaved sampling pattern size is equal to 8×8 and the screen resolution to 1024×1024. The resampling rate for BIR and the number of candidates generated with MTMH are equal to 10. As we use the same renderer for Bidirectional Instant Radiosity and Metropolis Instant Radiosity, the rendering times are identical. The given VPL generation time finally includes the construction time of the CDF for the physical light sources.

way since the sampling / resampling strategy provides the exact density only if we resample an infinity of candidates. Furthermore, to ensure that we do not discard a VPL which brings some power to a small part of the screen, we must set a non-null probability for all VPLs and therefore increase the variance of the estimator. On the contrary, MIR proposes to compute the desired density with a Markovian process. Since we have an appropriate initial random variable and an efficient mutation strategy using multiple candidates, the density is obtained very soon in the chain and the overall quality of the resulting estimators is very good.

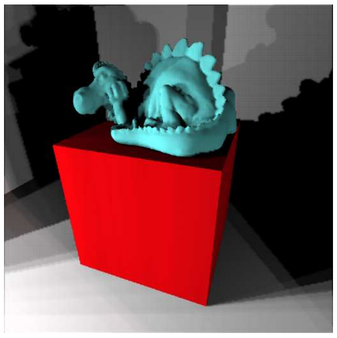
The scene presented in Figures 8.5.d, 8.5.e, 8.5.f finally shows that sampling VPLs with independent random variables can be extremely inefficient. With this layout, the ratio of the measure of relevant paths and the measure of all paths is indeed so small that both bidirectional and standard VPL sampling strategies are inefficient. On the contrary, MIR effectively explores the integration space around the relevant candidates and provides a very good VPL distribution.

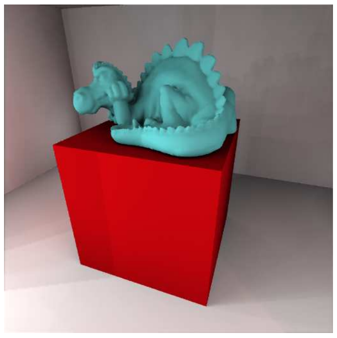
8.5.6 Overall Results

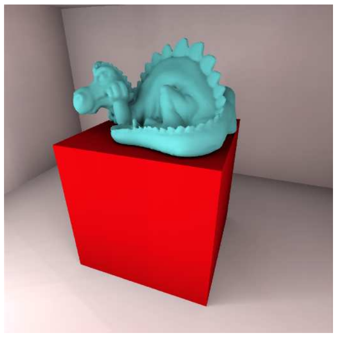
Table 8.2 sums up the computation times obtained with our implementation with the scenes and the camera positions presented in this chapter. For almost all scenes, MIR provides more relevant sample sets with a smaller amount of time. For the Three Dragon Room scene, where it is slower, MIR spends the most time to the evaluation of P_c (see Section 8.3.1) because we do not fix the total number of paths but the total number of paths which bring some power to the camera. However, even if we let the same computation time to BIR, the resulting distribution is much less relevant. On the other hand, if we want to achieve the quality provided by MIR with BIR, the needed resampling rate is superior to 1000 and thus inappropriate for interactive rendering. Finally, the current implementation of the samplers is neither optimized nor multi-threaded and we believe that the sampling process can be easily accelerated.

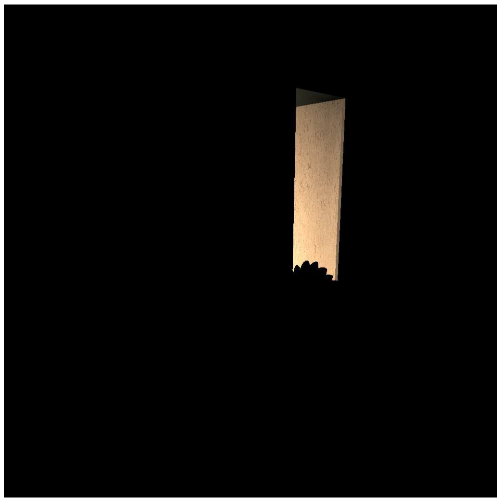
8.6 Limitations, Discussions and Conclusion

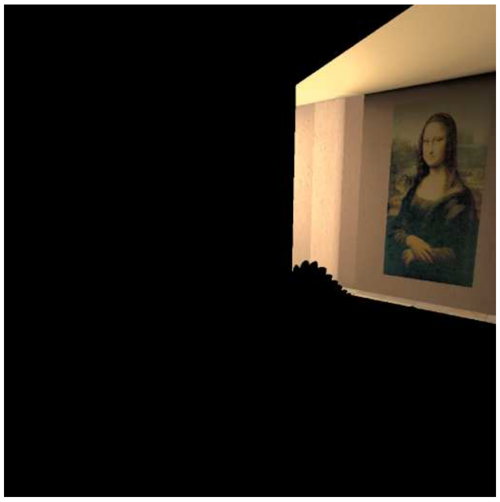
Even if we think that our approach provides significant improvements in difficult cases, we must clearly underline its limitations. First, as our method is view-dependent, flickering issues may occur. To solve this kind of problem, Ghosh et al. [24] recently proposed a sequential Monte-Carlo technique to limit flickering while sampling environment maps. Adapting and applying this strategy to the generation of VPLs may provide satisfactory results. Secondly, Instant Radiosity and MIR only handle diffuse or not-too-glossy surfaces. Directly using VPLs to illuminate very specular surfaces will produce very high variance estimators: it would be interesting to generate a reflected ray and to find a diffuse surface to perform the VPL gathering. Thirdly, even if our approach provides good results with directly-lit scenes in comparison with other importance-driven methods, we think that the better sample distribution offered by a low discrepancy sequence will give higher-quality results. Moreover, Owen and Tribble recently proposed a Quasi-Monte Carlo Metropolis algorithm [59] which could provide a good sample space stratification *and* an effective exploration of the integration space thanks to MCMC mutations. Finally, our technique cannot handle caustics and this can motivate an alternative but attractive research direction: trying to set up an interactive Metropolis Light Transport system. We propose in the next chapter some results in that direction since we design a coherent Metropolis Light Transport system.

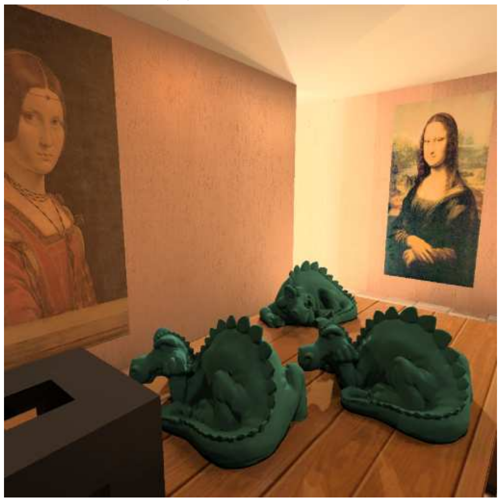

(a) - Standard

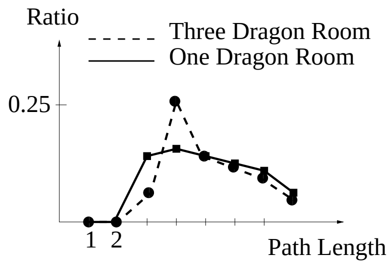

(b) - Bidirectional


(c) - MTMH


(d) - Standard

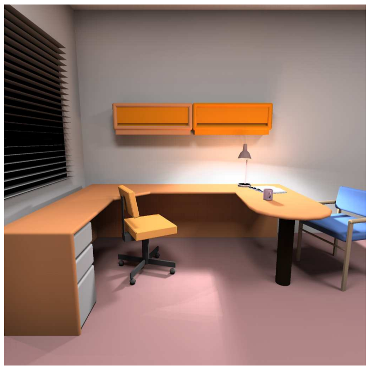

(e) - Bidirectional

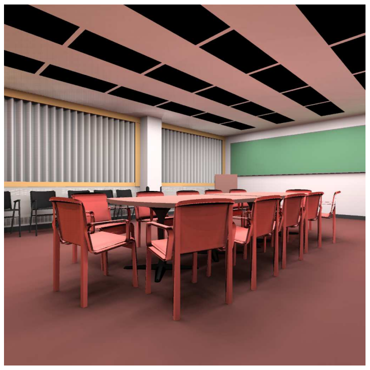

(f) - MTMH

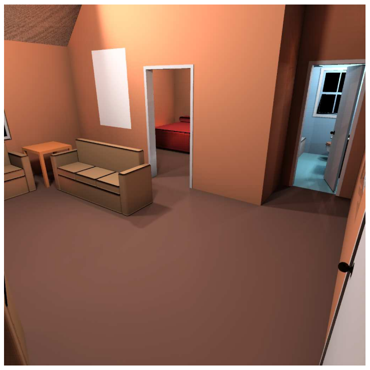


(h) - Path Lengths

Figure 8.5: **Indirect Illumination Stress Tests.** All pictures are computed with 1024 VPLs. We also give the different numbers of paths per length obtained while performing the mutations.

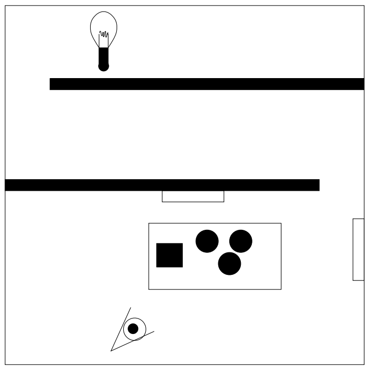

(a) - Office

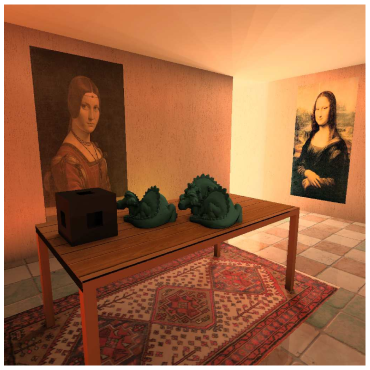

(b) - Conference


(c) - Cabin

(d) - Theater

(e) - Cruiser


(f)


(g) - Three Dragon Room

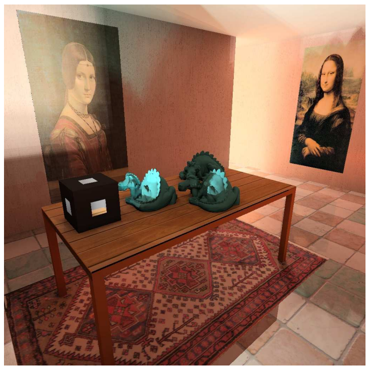

(h) - Jack-o-Lantern in (g)

Figure 8.6: **Some Images Rendered with Metropolis Instant Radiosity.** A coherent ray tracer and 1024 VPLs are used here. Our method which consists in describing a Virtual Point Light (VPL) sampler as a Markovian process provides very satisfactory results in many cases: in mostly directly-lit scenes as shown in (a), (b) and, (c) with many light sources or complex scenes as shown in (d) and (e), and with very difficult visibility issues as shown in Figures (f), (g), and (h).

CHAPTER 9

Coherent Metropolis Light Transport with Multiple-Try Mutations

We present in this chapter an effective way to implement coherent versions of Metropolis Light Transport (MLT) by using a class of Multiple-Try mutation strategies. Indeed, even if MLT is an unconditionally robust rendering technique which can handle any kind of lighting configurations, it does not exploit any computation coherency. For example, it is difficult to cluster similar light rays into beams or cones, to perform SIMD computations on vectorized data or to efficiently use geometry caching with non-tessellated scenes. To make Metropolis Light Transport suitable for most of the currently existing commercial renderers, we therefore propose to divide the algorithm into two parts: the first one explores the entire integration space in a way very similar to the initial implementation of Metropolis Light Transport while the second one ”splits” in an unbiased way each sample into a family of arbitrarily coherent samples. We finally propose to illustrate the efficiency of our approach with an example of implementation of coherent ray tracing using SIMD instructions.

As we show it further in this chapter, Coherent Metropolis Light Transport (CMLT) is in a sense close to Instant Radiosity [47] and its derivatives (see Chapters 7 and 8): instead of sampling a set of virtual point lights and using them to illuminate the parts of the scene seen by the camera, CMLT samples a set of complete paths (going from the light sources to the camera) and proposes to perturb them in a coherent manner to compute the final picture.

The remainder of the chapter is finally organized as follows. Section 9.1 presents an overview of our contribution. Section 9.2 exposes in details the new sampling and mutation strategies we set up to compute global illumination by combining MLT and ray packets. Section 9.3 gives all the necessary details to achieve the implementation of our technique. In Section 9.4, we present the results we obtained and some comparisons with other related approaches. The limitations and possible future work are finally given in Section 9.5 and Section 9.6 concludes.

9.1 Overview of the Algorithm

Metropolis Light Transport (MLT) is an innovative algorithm proposed by Veach and Guibas [88]. In the previous chapter, we described in details the algorithm and introduced a new algorithm, Metropolis Instant Radiosity, based on MLT and Instant Radiosity. On the contrary, we propose in this chapter an orthogonal approach: instead of making Instant Radiosity more predictive and more robust, we aim at accelerating the Metropolis Light Transport algorithm by making it more suitable to today CPU architectures. Indeed, the limitations of Instant Radiosity to not-too-glossy scenes and the difficulty to handle caustics make us think that MLT would provide better results if its rendering times were sufficiently improved.

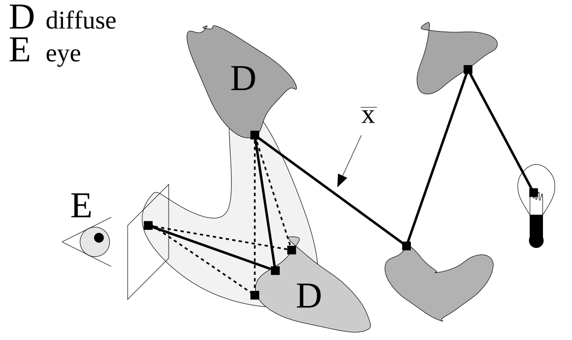


Figure 9.1: **Multiple-Try Mutations with Metropolis Light Transport.** Here, we have mutated a part of path $\overline{x}$ given by a MLT algorithm. By using a class of Multiple-Try mutations, we generate at once, several sub-path candidates from the camera (represented by dashed lines): these mutations allow us to cluster rays into ray packets, to factorize cache accesses and so on.

In fact, Metropolis Light Transport has several decisive advantages over almost all other sampling and integration techniques:

- its theoretical elegance since it proposes a unified solution to the Light Transport Problem;
- its numerical robustness and its insensitive behavior in relation to the scene configuration;
- its unbiasedness.

Unfortunately, two major drawbacks make it unsuitable for production renderers.

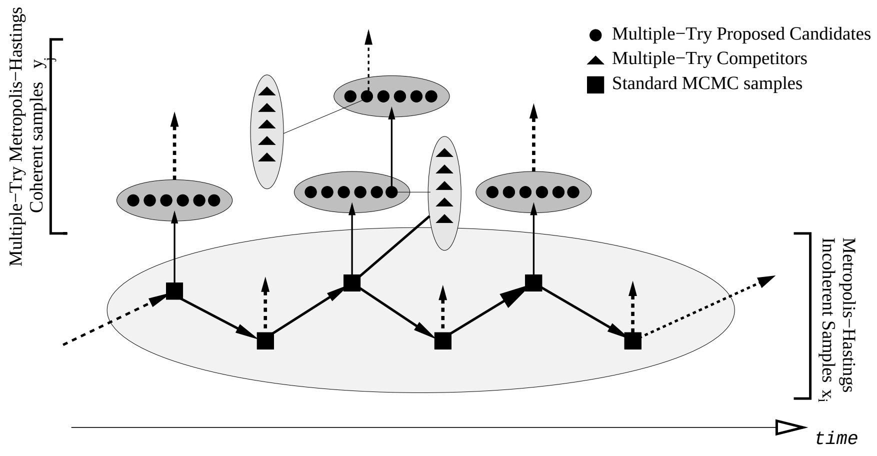
Algorithm 7 Coherent Metropolis Light Transport with main chain length n and sub-chain length m

```
1: Set all pixel intensities to 0;
2: Compute the power $P_c$ received by the camera and choose a first path sample $\overline{x}_1$;
3: for $i = 1$ to $n$ do
4:     $\overline{y}_1 \leftarrow \overline{x}_i$
5:     for $j = 1$ to $m$ do
6:         With a Multiple-Try mutation, "split" current sample $\overline{y}_j$ into several candidates. Use ray packets or ray packet frustums to compute ray intersections or BRDF evaluations. Accumulate all the multiple-try candidates proportionnaly to the power they bring to the camera and the generalized Metropolis ratio used to generate them {see Section 9.2 for more details};
7:         Choose the next sample $\overline{y}_{j+1}$;
8:     end for
9:     Generate the next candidate $\overline{x}_{i+1}$ with a Metropolis-Hastings mutation;
10: end for
11: Scale the pixel intensities such that the power received by the camera becomes $P_c$.
```

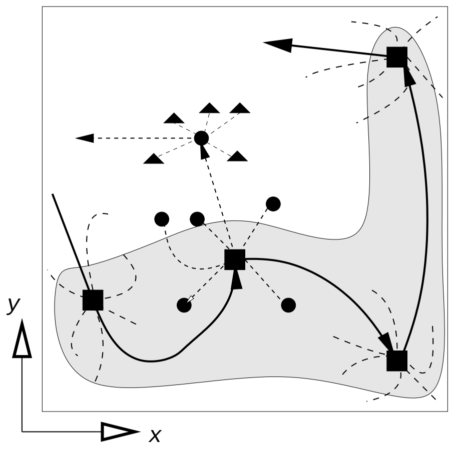
- due to the very large number of random paths generated for a given picture, flickering problems are difficult to handle when rendering an animation;

- it has very poor *algorithmic* properties. Indeed, since the samples are sequentially generated, the result of sample n is needed to compute sample $n+1$ and it therefore becomes very difficult to cluster rays. Furthermore, to make the algorithm unbiased, large changes may happen from one sample to the next one making thereby caching strategies inefficient.

Fortunately, the ray tracing literature has recently known large and decisive advances so that making Metropolis Light Transport efficient seems possible. Wald et al. developed in 1999 a vectorized ray tracing implementation using SSE SIMD instructions [92]. Since his first implementation, Wald applied the coherent ray tracing algorithm to a large number of applications like interactive global illumination or interactive photon mapping [90]: these applications were the first source of inspiration when we were implementing Metropolis Instant Radiosity presented in the previous chapter. More recently, Reshetov et al. proposed a conservative extension of ray tracing, the Multi-Level Ray Tracing Algorithm (MLRTA) [68]: by clustering rays into pyramids, they managed to achieve real time rendering with complex scenes on commodity computers. In 2006, other improvements were done and interactive or real time frame rates were also achieved with dynamic scenes [31, 89, 93]. All these strategies nevertheless require that rays are coherently grouped into beams or pyramids or more generally that they are sufficiently coherent. This motivates our approach. We want to bring new numerical tools to enhance and speed up MLT and make it suitable to the recent advances made in ray tracing. Our second goal is to propose a new organization of the method such that MLT may be implemented in commercial renderers. Indeed, Christensen showed that ray tracing can be used to render very complex scenes if ray coherency (which can be tracked by ray differentials) is ensured [8,9].



(a)



(b)

Figure 9.2: **Coherent Metropolis Light Transport.** (a) presents the temporal representation of the algorithm while (b) shows it in screen space where each light path is projected.

All the above problems therefore motivated our approach: by making MLT coherent, we want to produce a first step towards its implementation in production renderers. Algorithm 7 sums up our technique: it consists in extending the original implementation of Metropolis Light Transport [88] by adding new sequences of Multiple-Try mutations which can be easily handled by many efficient ray tracing methods. As shown in Figure 9.1, we divide the method into two parts to ensure the computation coherency. First, we explore the entire integration domain by generating a set of standard light path samples with Metropolis Light Transport algorithm. Then, for each MLT sample, we generate a family of Multiple-Try candidates. This second part therefore creates many candidates at once around the original MLT sampled path and computes the intersections and the BRDF evaluations with ray packets or ray packet frustums. With this approach, the variance of the estimators may be slightly increased but their efficiency are considerably improved.

To sum up our method, we propose to amortize the incoherent Metropolis Light Transport sampling process by using sequences of coherent, fast and parallel Multiple-Try mutations for each sample obtained with MLT.

9.2 Coherent Metropolis Light Transport

In this section, we present the core of our contribution, i.e. the new mutation family we add to enhance the speed of MLT.

9.2.1 MTMH Algorithm

The main reason for the poor coherency of computations while using MLT is the fact that all {sampling/BRDF evaluation/accumulation} processes are sequential: the results for path n are needed to evaluate the results for path $n + 1$. To break the sequential aspect of the algorithm, we propose to generate many samples at once using a Multiple-Try Metropolis Hastings algorithm (MTMH). One may refer to Chapter 8 for a complete presentation of the algorithm.

Figure 9.2 presents the two algorithms, MH and MTMH, as they are used in our new implementation of MLT. As we can see, to determine if the proposed candidate is accepted or rejected, the MTMH competitors represented by triangles are tested against the MTMH candidates represented by disks.

9.2.2 Application to Metropolis Light Transport

MTMH can be applied in very different ways. The first obvious implementation would be to replace all MH mutations by MTMH mutations. Unfortunately, this technique would provide very poor results: if you generate for example, 10 samples for each MTMH mutation, you will accumulate only one candidate among the 10 ones thereby resulting in a very poor efficiency. Furthermore, if the candidates are independently generated, the computation coherency can not be ensured since bidirectional mutations (see [88] for all necessary details about this mutation strategy) may explore very different parts of the sample space. What we propose here is therefore a bit different.

9.2.2.1 Unbiased Exploration of the Sample Space

We first generate a path sample family with only bidirectional mutations by using standard MH proposals (represented by squares in Figure 9.2). This sampling process provides a sample family with a density equal to $f^{(c)}/||f^{(c)}||$ and the resulting estimator is therefore unbiased. Conversely to the original MLT, we do not use the caustic perturbation strategy or the lens one since our goal is to explore the entire integration domain in an unbiased way and not to provide efficient estimators. This first part finally provides a set of n path samples $(\overline{x}_i)_{i\in[1\dots n]}$.

9.2.2.2 The "Lens Sub-path" Space

For a given path $\overline{x}_i$, the lens sub-path $(\mathbf{x_{i,1}}\dots\mathbf{x_{i,k}})$ is the sub-path of the form $ES^*DS^*(L|D)$ (where we use Heckbert's regular expression notation [35]; S, D, E, and L stands for non-diffuse, diffuse, lens and light vertices respectively); the light sub-path is on the contrary, the remainder of the path (which may be void if the lens sub-path is already connected to a light source). We can give an intuitive representation of the lens and light sub-paths: if we would want to discretize the incoming radiance field by a set of virtual point lights (as done in Instant Radiosity [47]), appropriate positions for these point lights would be the second diffuse surface, i.e. the ending point of the lens sub-path. This point can therefore be considered as a temporary virtual point light which illuminates the first diffuse surface seen by the camera through specular reflections.

For each $\overline{x}_i$, we can therefore define lens sub-path spaces Ω_i^{ls} such as $\overline{x}$ belongs to Ω_i^{ls} if and only if the light sub-path of $\overline{x}$ is identical to the light sub-path of $\overline{x}_i$. This sub-space is quite interesting since it is much smaller than the entire path space and can be entirely explored by using only lens or caustics *perturbations*: once the initial path $\overline{x}$ is given, we just have to generate new sub-paths around the lens sub-path of $\overline{x}$ to sample it.

Figure 9.1 gives for example the strategy that can be used to explore a *diffuse* lens sub-path, i.e. a sub-path of the form EDD: this approach here consists in sampling rays from the camera and in connecting the computed intersection to the second diffuse surface of the lens sub-path of $\overline{x}$. If we are able to cluster many camera rays *before* performing any intersection or BRDF evaluation, we will therefore be able to factorize many operations. As these strategies are implementation dependent, we give in Section 9.3 all the necessary details to implement effective perturbations to explore the lens sub-path space.

What we propose is therefore to use a MTMH sampler to explore each subspace Ω_i^{ls} and thus, to compute new unbiased estimators from the initial unbiased estimator given by the sample family $(\overline{x}_i)$. Before extensively presenting the MTMH version of the lens sub-path space, we first detail a MH version and explain why it does not provide both computation coherency and low-variance estimators. We will finally show that the exploration of the lens sub-path space with MTMH will give us the theoretical roots of a coherent implementation of the Metropolis Light Transport algorithm.

9.2.2.3 Exploring the Lens Sub-path Space with MH

As the samples $(\overline{x}_i)_i$ are generated proportionally to $f^{(c)}$, each of them brings an equal quantity of energy to the camera (equal to $||f^{(c)}||/n$). By counting how many path samples go through each pixel, we have, as shown by Veach and Guibas, an unbiased estimator of the power received by each pixel. From a given $\overline{x}_i$, we can then perform a MH move to generate a new path $\overline{y}_i$ with an acceptance probability equal to R_i and a transition function $q(\overline{x}_i, \overline{y}_i)$. As we use a MH move, the detailed balance is satisfied and the energy transfered from $\overline{x}_i$ to $\overline{y}_i$ is equal to the energy transfered from $\overline{y}_i$ to $\overline{x}_i$. Therefore, the estimator obtained by weighting the contributions of the two families $(\overline{x}_i)_i$ and $(\overline{y}_i)_i$ by $1 - R_i$ and R_i is still unbiased.

This assumption can lead to several strategies to explore the lens sub-path space. Cline et al. propose, for example, to build a finite length Markov-Chain from each $\overline{x}_i$ and to recursively deposit the energy according to the Metropolis ratio evaluated at each step [11]. This strategy is interesting to reduce the variance of the estimators, but as the paths are sequentially generated, it seems very difficult to cluster the generated rays. Another strategy, also suggested in [11], would be to split each sample into m candidates and to proportionally deposit the energy of each generated sample. This method seems to ensure the computation coherency, but it leads to high variance estimators since the energy received by the mutated samples will greatly vary. Iterating in this manner will result in an exponential growth in the number of samples. As our goal is to maintain low variance estimators *and* to exploit the computation coherency, MTMH moves provide the best of the two previous approaches.

9.2.2.4 Exploring the Lens Sub-path Space with MTMH

With MTMH mutations, we are able to generate p different candidates $\overline{x}_1^* \dots \overline{x}_p^*$ (represented by disks in Figure 9.2) for a given initial sample $\overline{x}^{(t)}$ (see Section 9.2.2 for the notations). Once the single proposal $\overline{x}_j^*$ has been chosen, we generate p competitor samples ($\overline{x}_1^{**} \dots \overline{x}_{p-1}^{**}$, $\overline{x}_p^{**} = \overline{x}$) from $\overline{x}_j^*$. Then, the process is repeated by setting as current sample, either $\overline{x}$ or $\overline{x}_j$. This approach therefore allows to generate many samples at once and to perform a Markov Chain which can efficiently explore the lens sub-path space. The last problem which has to be solved is to use and accumulate the contributions of *all* the generated samples, i.e. all the candidates $\overline{x}^*$ and all the competitors $\overline{x}^{**}$. To achieve such a result, we can make a simple remark: compared to the MH algorithm, MTMH only replaces the single candidates by families of candidates and, as indicated in [41], the detailed balance is therefore still maintained. This leads to the following strategy; for each step of the MTMH mutations, we build $2p$ estimators:

- the first p ones are $\overline{x}_1^* \dots \overline{x}_p^*$. Their contributions are respectively weighted by $R_g \cdot w_i^*$ where $w_i^* = f^{(c)}(\overline{x}_i^*)/\sum_{k=1}^{p} f^{(c)}(\overline{x}_k^*)$;
- the last p ones are $\overline{x}_1^{**} \dots \overline{x}_p^{**}$. Their contributions are respectively weighted by $(1 - R_g) \cdot w_i^{**}$ where $w_i^{**} = f^{(c)}(\overline{x}_i^{**})/\sum_{k=1}^{p} f^{(c)}(\overline{x}_k^{**})$.

This approach can be intuitively explained: as we want to maintain the detailed balance, we first accumulate each family proportionally to "its Metropolis ratio", then, we accumulate each element $\overline{x}$ inside a given family proportionally to its energy function, i.e. $f^{(c)}(\overline{x})$.

9.2.2.5 User Parameter Values

We set $\lambda(u, v) = [g(u|v) \cdot g(v|u)]^{-1}$ to encourage certain types of proposals: by using this specific λ, $w(x_i, x^*)$ corresponds to the importance weight $\pi_\infty(x^*)/g(x^*|x_i)$. However, we noticed that the algorithm was not particularly sensitive to the value of λ.

For the Gaussian standard deviations, we finally found that appropriate values were close to the parameters proposed by Veach and Guibas in [88]. For example, for the radius R of the MTMH perturbations in screen space, setting σ between 5% and 10% of the image size (width or height) provides good results.

However, they strongly depend on the number p of accumulated candidates and competitors. For more details about the numerical behavior of the strategies, please refer to Section 9.4.2.

9.2.3 Summary of the Approach

The roots of Coherent Metropolis Light Transport are therefore the MTMH mutations. On the one hand, we explore the entire sample space with MH bidirectional mutations. On the other hand, for each MH path, we sample the corresponding lens sub-path space in a fast manner by performing a finite length sequence of MTMH *perturbations*. Compared to the standard Metropolis Light Transport, we may therefore notice two deep changes:

- the perturbations and the bidirectional mutations are reorganized. Instead of sequentially alternating {lens/caustics} perturbations and bidirectional mutations, we deterministically apply a sequence of perturbations for each bidirectional mutation;
- all MH perturbations are replaced by MTMH perturbations.

9.2.4 Conceptual Differences with Metropolis Instant Radiosity

One may wonder what the differences with Metropolis Instant Radiosity are (see Chapter 8).

First, the way we propose to use the Multiple-Try mutations is orthogonal. With Metropolis Instant Radiosity, Multiple-Try mutations aim at decorrelating the successive virtual point lights: indeed, as many candidates are proposed at each step of the algorithm, the acceptation ratio is increased and large moves are likelier. With Coherent Metropolis Light Transport, the Multiple-Try mutations are used to coherently deposit many contributions at once. We may notice that it is also possible to use the Multiple-Try mutations during the incoherent path sampling process of Coherent Metropolis Light Transport. We just have to replace the standard MCMC mutations by Multiple-Try ones.

Secondly, Coherent Metropolis Light Transport is much more generic. Indeed, it replaces, in a sense, the virtual point lights by complete light paths. The gathering step which consists in illuminating with the set of VPLs, the parts of the scene seen by the camera is thus replaced by the coherent Multiple-Try mutations of the current path.

Finally, Metropolis Instant Radiosity is actually included in Coherent Metropolis Light Transport. Indeed, if you choose as Multiple-Try candidates and competitors, *one path for each pixel in the screen*, Coherent Metropolis Light Transport can be reexpressed as a Metropolis Instant Radiosity algorithm. It is however not the best strategy when, for example, there are many different local modes in the screen (i.e. there are many parts in the screen which are illuminated by different surfaces in the scene). Local explorations of the sampled space is, in this kind of configuration, more interesting.

We now have to analyze a last point: how can we mutate the lens sub-paths in a fast and coherent way and how can Coherent Metropolis Light Transport be implemented?

9.3 Implementing Coherent Metropolis Light Transport

The previous sections did not detail the implementation of our technique. We therefore present here how we perform the MTMH mutations to explore the lens sub-path space and how we can cluster rays to speed up the computations.

9.3.1 Designing an Effective Lens Sub-space Exploration

The first step is to build an effective mutation strategy to generate the MTMH candidates and competitors. Before detailing our method, we can make two remarks which motivated our choice:

- paths $\overline{x}$ generated by standard MLT (represented by squares in Figure 9.2) are already an effective sample set with density $f^{(c)}/||f^{(c)}||$: that motivates to explore the close neighborhood of each sample;
- the mutation strategy to generate MTMH mutations must not introduce extra sampling artifacts or patterns which may be slow to disappear.

These two points lead us to use Gaussian random variables with density $q_{\mu,\sigma}$ defined by $q_{\mu,\sigma} = \frac{1}{2\sigma\pi} exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$. Indeed, depending on the nature of the given sub-path, we have to use two different path generation strategies:

- let assume that the lens sub-path $\mathbf{x_0} \ldots \mathbf{x_k}$ has the form $ES^*DS^+(D|L))$ (it is a caustic lens sub-path): we generate a path starting from x_k, the second diffuse surface (or the light source). The direction of the segment $\mathbf{x_k}, \mathbf{x_{k-1}}$ is perturbed by a random (θ, ϕ) where ϕ is a uniform random variable and θ is a Gaussian random variable with $\mu = 0$ and σ is user-defined;
- otherwise, the lens sub-path $\mathbf{x_0} \ldots \mathbf{x_k}$ has the form $ES^*D(D|L)$ (it is a non-caustic lens sub-path): we perturb the old image location by moving it a

Gaussian random distance R and a uniform angle ϕ.

For these two kinds of mutations, we noticed that the Gaussian strategies were the most effective. Since we want to use a large number of MTMH candidates per MH sample, a uniform density for R or θ gives very poor results. After implementing it, we noticed that bright circles (which were slow to disappear) may appear around duplicated MH samples. Other strategies were tested like linear densities, but none of them gives satisfactory results. Conversely, since all derivatives of Gaussian distributions are null at μ, the Gaussian MTMH perturbations offer very smooth transitions and the results we got are very similar to the ones obtained with a standard MLT (see Section 9.4).

We finally have to make an important remark concerning Gaussian densities: in our case, we do not use exact Gaussian densities, but clamped ones which are defined on a given domain Ω by:

$$q_{clamped_{\mu,\sigma}} = \frac{q_{\mu,\sigma}}{\int_\Omega q_{\mu,\sigma}(\omega)d\omega}$$

In the particular case where Ω is the set of screen coordinates, $q_{clamped_{\mu,\sigma}}$ is different for each pixel. We first precompute and store on the hard disk a Gaussian map which gives for every pixel the value of $1\ /\ \int_\Omega q_{\mu,\sigma}(\omega)d\omega$. This allows us to sample clamped Gaussian random variables with a rejection technique and no extra computation at run-time.

9.3.2 Making the Computations Coherent

As described before, the MH part of the algorithm (represented by the Markov Chain of squares in Figure 9.2) is incoherent and we do not intend to speed this part of the algorithm. On the contrary, we want to amortize the expensive cost of each MH by computing sub-sequences of MTMH lens sub-path samples. Several implementations are possible.

9.3.2.1 Coherent MLT with SIMD Ray Packets

In this section, we will suppose that the lens sub-path is non-caustic (the approach is quite similar for a caustic lens sub-path). Let us consider the given current path $\overline{x}$ we have to mutate and its corresponding lens sub-path $\overline{x}^{ls}$. To perform the perturbation of $\overline{x}^{ls}$, we first perform a $n \times m$ jittered and uniform sampling as presented in Figure 9.3. Then, by inverting the Gaussian density (a Box-Muller transform can, for example, be used), we generate a set of $n \times m$ ray segments that we use to trace the corresponding MTMH candidates. These complete sub-paths are obtained by perturbing the remainder of the segment in a way very similar to the multi-chain perturbation technique presented by Veach and Guibas in [88]. To achieve effective SIMD computations as presented by Wald in [92], we finally cluster 4 neighborhood rays (see Figure 9.3) and we perform the camera path tracing (or the light path tracing if we have to mutate a caustic lens sub-path) using a coherent ray tracer. We may notice that the path tracing part remains coherent since we cluster a set of close camera rays and we perturb the remainder of the path around the *same* given camera path $\overline{x}^{ls}$. As described by Benthin in

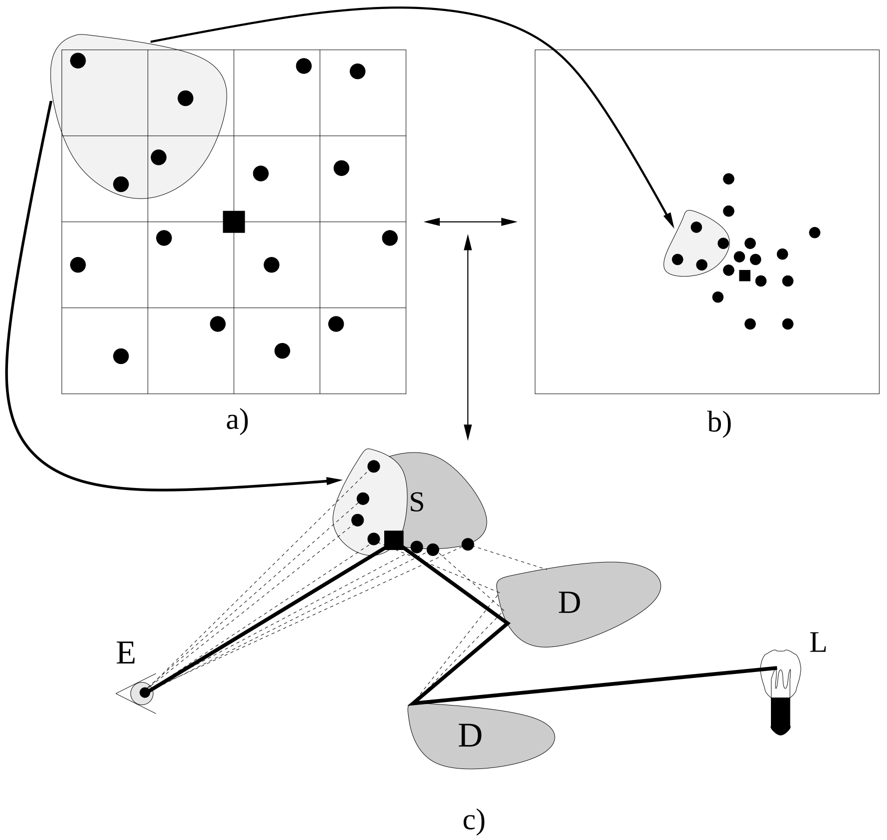


Figure 9.3: **Our Technique to Ensure Ray Coherency.** (a) is the uniform jittered 2D sampling. (b) shows the corresponding Gaussian sampling in screen space. We can see the 2×2 ray packet we made with 4 neighbor rays. (c) shows some resulting camera sub-paths: we have mutated here a $ESDD$ lens sub-path.

his Ph.D. [4], larger ray packets can also be used to more effectively speed up the intersection operations: this can also be implemented with the jittered approach and no extra theoretical difficulties.

9.3.2.2 Coherent MLT with Ray Packet Frustums

Very efficient algorithms were recently introduced to exploit in a better manner ray coherency by using ray packet frustums [68, 93, 96]: as the interval arithmetic supporting all these algorithms requires a common origin for all rays in the packet, we can not compute all kinds of lens sub-paths with ray packet frustums only. For EDD lens sub-paths, there is no specific problem, since the two ray packets have common origins (the first one is the camera while the second one is the second diffuse surface). Nevertheless, for a $ESDSL$ sub-path or other sub-path types, we can use ray packet frustums only for the rays starting from the camera and the rays starting from the light source: the remaining rays have to be processed by casual SIMD packets.

9.3.3 Summary and Remarks

MTMH perturbations thus allow us to generate many coherent rays at once, to perform coherent ray tracing and to use ray packet frustums when it is possible. The next section presents the results we obtained with our implementation and the tests we did to demonstrate the efficiency of our approach.

Furthermore, as indicated by Craiu and Lemieux in [14], the MTMH candidates and competitors do not even need to be independent so that the MTMH proposals can be directly generated with low-discrepancy number sequences and quasi Monte-Carlo techniques. This can lead to better estimators and may be really helpful to handle flickering-free rendering of animations (we give some ideas about this in Section 9.5).

9.4 Results

We implemented a complete rendering system to perform SIMD computations when using MTMH perturbations. We actually recoded the OpenRT API [17] and divided our CMLT renderer into two parts: the first one sequentially generates paths with a standard Metropolis Light Transport algorithm while the second one performs the sequence of MTMH mutations and executes the SIMD part of the shaders: the ray generation and the BRDF evaluations are completely vectorized and encapsulated in a user-friendly interface.

We finally have to notice that we did not implement a ray packet *frustum* technique so that much better performance can be expected if it was used (particularly for EDD lens sub-paths).

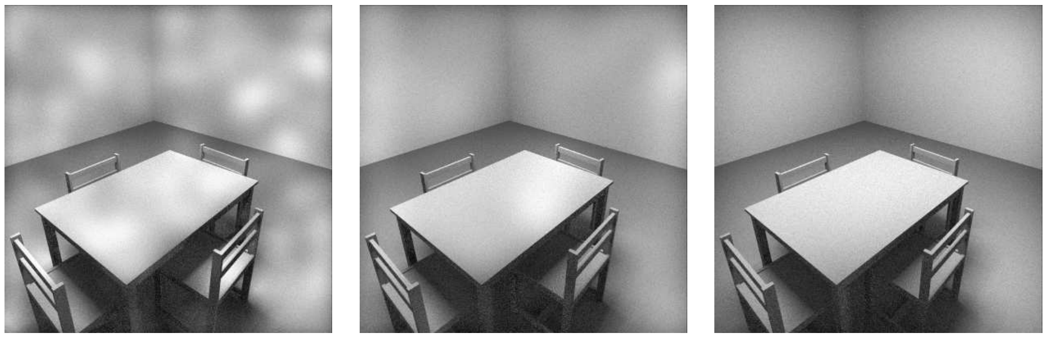

(a) $\sigma = 16$ pixels (b) $\sigma = 32$ pixels (c) $\sigma = 64$ pixels

Figure 9.4: **Different Values of σ.** For the 3 tests, we use 256 MTMH candidates and 256 MTMH competitors. The screen size is equal to 1024×1024 and about 20 mutations per pixel have been evaluated.

9.4.1 Comparison with Metropolis Light Transport

The first remark we can made is that, in a sense, our technique strictly contains MLT. Indeed, if you implement *MLT* and use the same number of perturbations and bidirectional mutations (it is what Veach proposed in [88]), implementing CMLT with length 1 MTMH sub-sequences and only 1 MTMH candidate will roughly provide the same results. However, CMLT adds three parameters which have to be analyzed:

- the standard deviation σ of the Gaussian random variables;
- the number of MTMH candidates (and competitors);
- the length of the MTMH sub-sequences.

9.4.1.1 Length of MTMH Sub-sequences

We first noticed that our method was almost insensitive to the sub-sequence length since our samples already follow density $f^{(c)}$. However, since the initial samples provided by a path tracer do *not* follow $f^{(c)}$, this behavior must be quite different if you combine MTMH sub-sequences and an Energy Redistribution Path Tracer: it would be certainly more interesting to use long MTMH sub-sequences. Finally, in a more complex rendering system (with for example, geometry caches), this parameter may need to be more carefully tuned.

9.4.1.2 σ and the Number of MTMH Candidates

Even if the algorithm remains unbiased for all values of σ, we have to carefully balance the number p of MTMH candidates and the size of the MTMH perturbations. Figure 9.4 shows a simple scene tested with different values of σ and $p = 256$. As we can see, generating too many MTMH candidates in a too small area causes very poor results. Indeed, Metropolis Hasting does not stratify well so that we directly inherit this poor stratification for small values of σ. On the contrary, the jittered sampling used for the MTMH perturbations greatly enhances the stratification of the sample set. In a sense, CMLT also offers a trade-off between variance and stratification over the image plane.

For a given implementation, it is therefore fundamental to carefully tune the respective values of σ and p. For all the pictures computed in this chapter, we set $p = 256$ and $\sigma = 5\%$ of the screen size.

9.4.2 Overall Performance

As indicated above, we perform the MTMH perturbations with a multi-threaded coherent ray tracer using the SSE SIMD instruction set. Compared to a non-vectorized implementation, we achieved a speed-up varying from 2.3 to 1.5: the acceleration actually depends on the lens sub-path length, the number of MTMH candidates and the size of the Gaussian perturbations. Once again, it is important to carefully tune the sampling parameters to maintain computation speed, good stratification properties and low variance. With our current implementation, we achieved between 1 or 2 million mutations per second on most of the scenes we tested. Our implementation seems to be very competitive compared to other existing MLT renderers. We actually think that aggressively exploring the lens sub-path space offers a good sampling strategy since perturbations are much less computationally expensive than bidirectional mutations. We finally believe that using ray packet frustums can still bring major improvements as speed-ups superior to 10 have been recently reported [68, 93].

9.4.3 Cache Simulation

Christensen et al. recently published two papers about distribution ray tracing in complex scenes [8, 9]. We believe that our technique, combined with multi-resolution geometry caching, can fit in production renderers such as the ray tracing system used within the RenderMan interface. Indeed, MTMH mutations are, in essence, very close to a standard distribution ray tracing framework: instead of generating reflections, refractions or shadow rays for a *uniform* set of pixels,

	Diffuse Office			Glossy Office		
	4 Tri	16 Tri	128 Tri	4 Tri	16 Tri	128 Tri
MLT	55%	61%	62%	38%	40%	45%
CMLT	87%	92%	99%	82%	91%	98%
IR	87%	92%	99%	81%	90%	98%

	Diffuse Conference			Glossy Conference		
	4 Tri	16 Tri	128 Tri	4 Tri	16 Tri	128 Tri
MLT	65%	77%	79%	70%	74%	77%
CMLT	80%	95%	98%	81%	95%	99%
IR	81%	96%	98%	82%	95%	99%

Table 9.1: **Some Cache Hit Statistics.** We use the four test scenes presented in Figure 9.5 and 3 cache sizes: 4, 16, and 128 triangles. Coherent Metropolis Light Transport (CMLT) easily outperforms standard Metropolis Light Transport (MLT). It roughly behaves as Instant Radiosity (IR) does.

we actually use the same strategies for a *Gaussian* pixel distribution. To analyze the performance of our technique, we simulated and implemented a simple cache system.

We actually associate a fixed size cache to the triangles of our test scenes (see Figure 9.5). When a ray / triangle intersection is requested, we check if the triangle is in the cache: if we find it, it is a cache hit, otherwise, it is a cache miss and the triangle is inserted in the cache. To evaluate the performance, we finally use four test scenes: a purely diffuse office, the same scene but with a glossy wall, a purely diffuse conference room and the same scene with a glossy floor. The goal is to exhibit the coherence properties of the tested algorithms with various layouts. We compare our method (CMLT) with Metropolis Light Transport (MLT) and Instant Radiosity (IR) [47] which is one of the most coherent and successful methods when combined with coherent ray tracing techniques. We noticed that CMLT provided results which are very similar to the results obtained with IR. Actually, the two methods have close algorithmic properties. For IR, we implemented a multi-threaded rendering tile system: each thread renders small tiles which are associated to a set of VPLs. At the same time, our CMLT rendering system finally replaces the tiles by a jittered Gaussian path distribution. Once the rays have been generated, the operations are quasi-identical.

Compared to IR, we however have to remind that a CMLT system requires to write *anywhere* in the frame buffer which may be limiting in a multi-threaded environment.

9.5 Limitations and Future Work

As an extension of Metropolis Light Transport, our method inherits some of its drawbacks.

9.5.1 Flickering Problems

The first one is the difficulty to handle animations without flickering problems. This issue has actually two main reasons:

- as the samples are sequentially dependent, if one of them considerably changes from one frame to the next one, all the following samples in the Markov Chain will be consequently modified and major flickering issues will suddenly occur;
- as Metropolis Light Transport is a pure Monte-Carlo technique, we have to use a lot of pseudo-random numbers which are difficult to reuse in a flickering-free way.

We however think that handling flickering issues may be much easier with Coherent Metropolis Light Transport. First, it seems to be possible to store all the MH paths (represented by squares in Figure 9.2) on hard-disk. Indeed, compared to a MLT algorithm, most of the computed paths are generated with MTMH perturbations and the MH paths obtained with bidirectional mutations only are much less numerous. When we compute a new frame, we can therefore re-read the stored paths, mutate them with a sequential sampler, and make them follow the density of the current frame as done by Ghosh et al. to compute illumination with environment maps in [24]. In other words, if we have a set of paths which follow the density $f_n^{(c)}$ of frame n, a sequential sampler can mutate these paths to make them follow $f_{n+1}^{(c)}$. As the MTMH samples do not need to be independent, we can moreover use deterministic quasi random sequences of samples which can easily be regenerated from one frame to the next one. These ideas may thus provide satisfactory results to handle flickering problems since most of the paths of the previous frame could be "reprojected" in an unbiased way into the next frame.

9.5.2 Other Sampling Strategies

The MTMH mutations strategies can finally certainly be applied to other samplers: there must be no specific problems to handle participating medias, motion blurs or spectral representations of materials. Furthermore, MTMH mutations can certainly be combined to an Energy Redistribution Path Tracing algorithm [11]. Instead of performing a sequence of MH perturbations, it would be not too difficult to replace them by a sequence of coherent MTMH candidates as we did with MLT.

9.5.3 Implementing a Distributed Framework

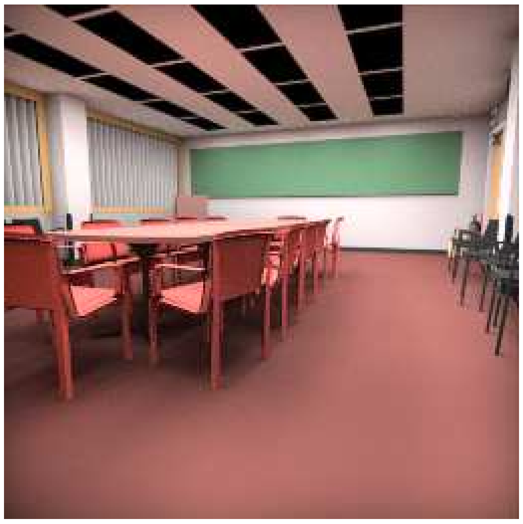
One difficult challenge is to efficiently implement a Metropolis Light Transport algorithm with a computer cluster. With our current implementation, it seems impossible to achieve interactive or real time frame rate, since running separate MLT on separate threads / computer is not appropriate to a distributed frame work (it would require a too large bandwidth to gather all pictures). A good approach would perhaps be to use an Interleaved Sampling technique [48] and therefore to compute the contributions of different pixel sub-sets on different machines. This would require only few changes into the different MLT samplers. Combining MLT and Interleaved Sampling could also provide good results with inhomogeneous multi-core architectures like Cell processors.

9.6 Conclusion

We presented in this chapter, a coherent extension of Metropolis Light Transport which can easily be combined with most of the recent advances achieved in the field of ray tracing. By adding Multiple-Try perturbations, we can coherently compute most of the intersection and BRDF operations by efficiently using ray packets or ray packet frustums. We now believe that an interactive and even real time MLT system can be implemented: if the remaining problems previously presented are solved, we could finally have an unconditionally robust rendering system which could handle any scene of any complexity and as MLT initially did, any lighting layout without the necessity to use aggressive filters or to bias the method.

(a) - Diffuse Office

(b) - Glossy Office


(c) - Diffuse Conference Room

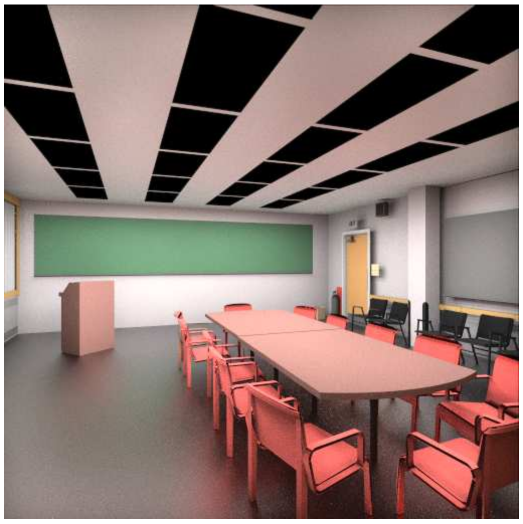
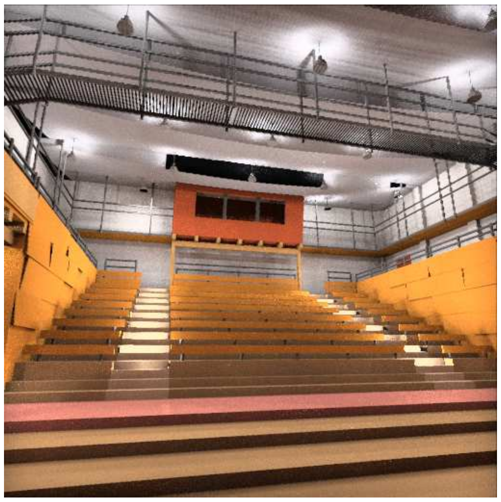
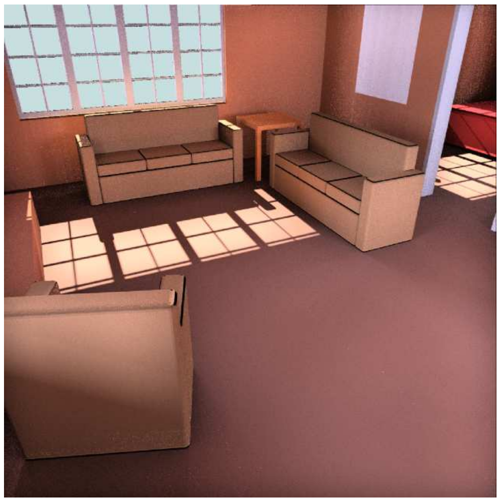

(d) - Glossy Conference Room

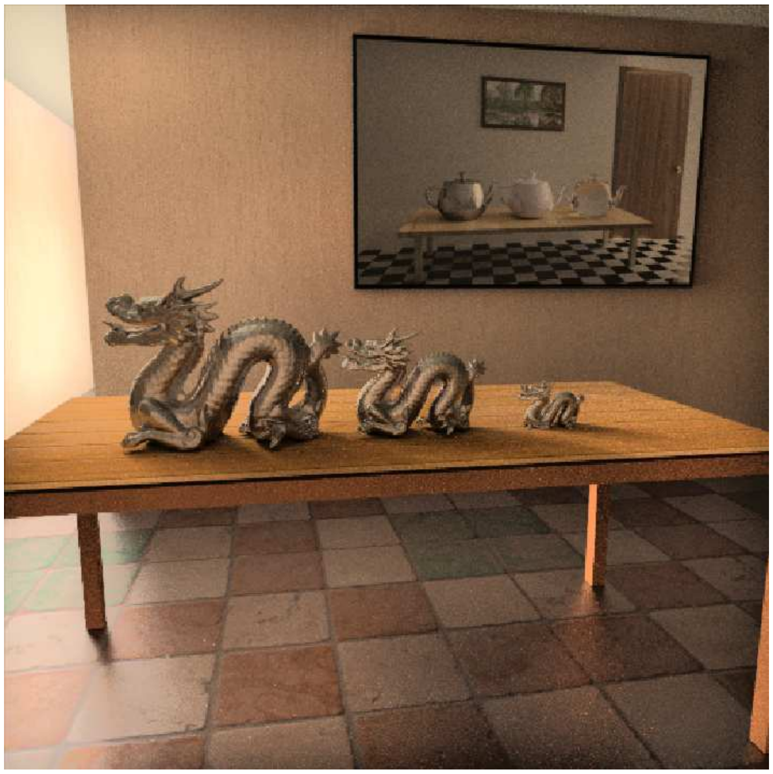
Figure 9.5: **The Four Scenes to Test the Coherency of our Algorithm with a Simulated Cache System.** The 1024×1024 pictures presented here, are computed in about 5 minutes on a Core Duo and a 2×2 SIMD ray packet system. Office contains $35,000$ triangles and Conference Room $200,000$ triangles



(a) Candelstick Theater



(b) Cabin Lit by a Simple Sky



(c) The MLT Room

Figure 9.6: **Three** 1024×1024 **pictures rendered in** 10 **minutes with Coherent Metropolis Light Transport.** About 1000 mutations per pixel are used. We accelerate the Metropolis Light Transport algorithm by adding a new class of Multiple-Try coherent mutations. With these mutations, we are now able to use ray packets or ray packet frustums to speed up the intersections and the BRDF evaluations.

CHAPTER 10

Final Summary, Conclusions and Future Work

In this dissertation, we focus on the domain of realistic image synthesis in computer graphics while targeting physically-based as well as real-time or interactive rendering. We actually explore two different but complementary research areas. First, we aim at efficiently using modern hardware architectures (GPUs or CPUs) and at implementing either effective rasterization renderers or ray tracing systems. Secondly, we try to propose new probabilistic numerical schemes using Virtual Point Lights (VPLs). As already indicated in the thesis, effective implementations may require some restrictions in the abstraction and the possibilities provided by the numerical algorithm: using virtual point lights allows us to simulate many light transport phenomena with few restrictions while maintaining interactive or real-time frame rates. We can more precisely sum up our contributions in the following manner:

10.1 Summary of Contributions

10.1.1 Another Presentation of Monte-Carlo Rendering

In the first chapters of the thesis, we try to expose the Monte-Carlo Rendering techniques, and more particularly those which aim at solving the light transport problem in its global form. Even if most of the ideas are not new and previously exposed in several Ph.D. thesis or books also dealing with Monte-Carlo rendering, we think that it is fundamental to recap the roots of any Monte-Carlo renderer:

- Probability theory and the Central Limit Theorem: they provide the converging properties of all sampling strategies: almost sure convergence and convergence in distribution of the random variable partial sums towards a Gaussian distribution;
- The path sampler families currently existing in computer graphics: they either generate paths of any length or intensively use Russian Roulettes to explicitly compute an infinite sum of integrals.

Even if most of engineers and researchers in computer graphics are familiar with Monte-Carlo methods, we think that a suitable formalization is fundamental. Once it is done, Monte-Carlo techniques are certainly ones of the simplest and most powerful numerical schemes since they only consist in:

Sampling, evaluating, and accumulating.

10.1.2 Interleaved Sampling on GPUs

We provide two new techniques to perform uncorrelated computations over neighbor screen pixels. The first approach, "Non-Interleaved Deferred Shading of Interleaved Sample Pattern" is very generic and can be used to extend any pre-existing renderer. With some extra lines of code to integrate it in an existing renderer, accumulating many light source contributions therefore becomes much easier. The second approach, "Interleaved Deferred Shading" is simpler but more restrictive and remains more adapted to real-time applications such as video games.

10.1.3 Variance Reduction Methods for VPL Rendering Techniques

We provide two variance reduction strategies to improve the location of virtual point lights. Indeed, as it is useless to sample virtual point lights which are not able to illuminate the parts of the scene seen by the camera and since the lighting computations are quite expensive, it is fundamental to improve the VPL distribution. The first technique, Bidirectional Instant Radiosity, proposes to sample VPL both from the camera and the light sources and to use a sampling / resampling step to improve the distribution. The second method, Metropolis Instant Radiosity, aims at directly sampling an optimal VPL distribution by using a Markov-Chain Monte-Carlo sampler. As these two techniques are entirely based upon Instant Radiosity, implementing them in a renderer already using virtual point lights seems not-too-difficult. In fact, we propose several examples of implementations using them (with CPUs or GPUs, with ray tracing or rasterization).

10.1.4 Coherent Metropolis Light Transport

Thanks to the experiences acquired with the study of virtual point light sampling strategies and the difficult implementations of interactive ray tracing techniques, we propose to extend the successful Metropolis Light Transport algorithm in a coherent manner. Our wish is actually simple: instead of illuminating parts of the scene seen by the camera with a set of virtual point lights, we similarly propose to coherently perturb complete light paths around some path candidates. The technique provides interesting results but as indicated in Chapter 9, we do not obtain interactive results yet. However, the excellent robustness of Metropolis Light Transport samplers makes us think that implementing an interactive MLT system remains quite appealing.

10.2 Conclusion

This Ph.D. thesis, even if it only focuses on a very small part of computer graphics, finally tries to make one more step towards the ultimate renderer, i.e. the renderer which would be able to simulate in real-time all the visible lighting phenomena regardless to the input properties. Of course, we do not succeed in such a quest but we really hope that the community will appreciate the results, the ideas and the efforts provided in this thesis. It is quite difficult to know the directions which will be chosen in the next years. First, the "good" technical choices remain today difficult to find. On the one hand, rasterization techniques are extremely fast, robust and can display any scene without expensive precomputations. On the other hand, ray tracing techniques recently make outstanding advances and remain the most natural approaches to deal with the light transport problem. The situation concerning the numerical schemes is worst and much more uncertain. Monte-Carlo renderers are the more flexible but one major step has to be made to implement them in real-time renderers such as those used in video games. However, even if techniques using precomputation steps are today the most popular ones, we think that the quest toward total interactivity strictly requires fast, robust and predictive rendering strategies. Contrary to off-line rendering where designers can improve and change the lighting design to make the scene most realistic, real-time rendering does not offer such a chance: it is therefore possible that real-time renderers will require more "physically-based simulation" than off-line ones will in the future.

10.3 Future Work

There is of course, a very large room for improvement. First, we think that it is quite important to handle flickering problems while using importance driven techniques. The three sampling methods we propose, Bidirectional and Metropolis Instant Radiosity, and Coherent Metropolis Light Transport, do not correctly deal with these issues. Secondly, Instant Radiosity does not handle very glossy or shiny surfaces and specific lighting phenomena such as caustics. It may therefore be very interesting to propose sampling techniques to tackle these problems. Finally, the incoming new processor architectures, using inhomogeneous configurations and massively parallel features will certainly bring strong and deep changes in the numerical scheme requirements: some algorithms will certainly have to be discarded while other ones will become competitive.

References

[1] James Arvo. *Analytic Methods for Simulated Light Transport.* PhD thesis, Yale University, 1995.

[2] Lionel Atty, Nicolas Holzschuch, Marc Lapierre, Jean-Marc Hasenfratz, Chuck Hansen, and François Sillion. Soft Shadow Maps: Efficient Sampling of Light Source Visibility. *Computer Graphics Forum*, 2006.

[3] Jeff Beck and Willian Chen. *Irregularities of Distribution.* Cambridge University Press, New York, 1987.

[4] Carsten Benthin. *Realtime Ray Tracing on Current CPU Architectures.* PhD thesis, Computer Graphics Group, Saarland University, 2006.

[5] Carsten Benthin, Ingo Wald, and Philipp Slusallek. A Scalable Approach to Interactive Global Illumination. In *Computer Graphics Forum (Proceedings of Eurographics 2003)*, pages 621–630, 2003.

[6] David Burke, Abhijeet Ghosh, and Wolfgang Heidrich. Bidirectional Importance Sampling for Direct Illumination. In *Proceedings of the 16th Eurographics Symposium on Rendering*, pages 147–156, 2005.

[7] Per Christensen. Adjoints and Importance in Rendering: An Overview. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, pages 329–340, 2003.

[8] Per Christensen. Ray Tracing for the Movie "Cars". In *IEEE Symposium on Interactive Ray Tracing*, pages 1–6, 2006.

[9] Per Christensen, David M. Laur, Julian Fong, Wayne L. Wooten, , and Dana Batali. Ray Differentials and Multiresolution Geometry Caching for Distribution Ray Tracing in Complex Scenes. *Computer Graphics Forum (Proceedings of Eurographics 2003)*, pages 543–552, 2003.

[10] 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.

[11] David Cline, Justin Talbot, and Parris Egbert. Energy Redistribution Path Tracing. In *SIGGRAPH '05 (Proceedings of the 32th annual conference on Computer graphics and interactive techniques)*, pages 1186–1195, 2005.

[12] Robert Cook. Stochastic Sampling in Computer Graphics. *ACM Transaction on Graphics*, pages 51–72, 1986.

[13] Robert Cook, Thomas Porter, and Loren Carpenter. Distributed Ray Tracing. In *SIGGRAPH '84 (Proceedings of the 11th annual conference on Computer graphics and interactive techniques)*, pages 137–145, 1984.

[14] Radu Craiu and Christiane Lemieux. Acceleration of the Multiple-try Metropolis Algorithm using Antithetic and Stratified Sampling. *Statistics and Computing*, 2007.

[15] Carsten Dachsbacher and Marc Stamminger. Splatting Indirect Illumination. In *SI3D '06 (Proceedings of the 2006 symposium on Interactive 3D graphics and games)*, pages 93–100, 2006.

[16] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The Triangle Processor and Normal Vector Shader: a VLSI System for High Performance Graphics. In *SIGGRAPH '88 (Proceedings of the 15th annual conference on Computer graphics and interactive techniques)*, pages 21–30, 1988.

[17] Andreas Dietrich, Ingo Wald, Carsten Benthin, and Philipp Slusallek. The OpenRT Application Programming Interface – Towards A Common API for Interactive Ray Tracing. In *Proceedings of the 2003 OpenSG Symposium*, pages 23–31, 2003.

[18] Kirill Dmitriev, Stefan Brabec, Karol Myszkowski, and Hans-Peter Seidel. Interactive Global Illumination using Selective Photon Tracing. In *Proceedings of the 12th Eurographics Symposium on Rendering*, pages 21–33, 2002.

[19] Craig Donner and Henrik Wann Jensen. Light Diffusion in Multi-layered Translucent Materials. In *SIGGRAPH '05 (Proceedings of the 32th annual conference on Computer graphics and interactive techniques)*, pages 1032–1039, 2005.

[20] Philip Dutré. *Mathematical Framework and Monte-Carlo Algorithms for Global Illumination in Computer Graphics.* PhD thesis, University of Lueven, 1996.

[21] Philip Dutré, Eric Lafortune, and Yves Willems. Monte Carlo Light Tracing with Direct Computation of Pixel Intensities. In *Compugraphics '93 (Proceedings of Third International Conference on Computational Graphics and Visualization Techniques)*, 1993.

[22] Shaohua Fan, Stephen Chenney, and Yu chi Lai. Metropolis Photon Sampling with Optional User Guidance. In *Proceedings of the 16th Eurographics Symposium on Rendering*, pages 127–138, 2005.

[23] Randima Fernando. Percentage Closer Soft Shadows. *Sketch in GDC 2005*, 2005.

[24] Abhijeet Ghosh, Arnaud Doucet, and Wolfgang Heidrich. Sequential Sampling for Dynamic Environment Map Illumination. In *Proceedings of the 17th Eurographics Symposium on Rendering*, pages 116–126, 2006.

[25] Abhijeet Ghosh and Wolfgang Heidrich. Correlated Visibility Sampling for Direct Illumination. *The Visual Computer*, pages 693–701, 2006.

[26] Andrew Glassner. A Model for Fluorescence and Phosphorescence. In *Proceedings of the 5th Eurographics Symposium on Rendering*, pages 60–70, 1994.

[27] Ned Greene. Environment Mapping and other Applications of World Projections. *IEEE Computer Graphics Application*, pages 21–29, 1986.

[28] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical Z-buffer Visibility. In *SIGGRAPH '93 (Proceedings of the 20th annual conference on Computer graphics and interactive techniques)*, pages 231–238, 1993.

[29] Gael Guennebaud, Loic Barthe, and Mathias Paulin. Real-Time Soft Shadow Mapping by Backprojection. In *Proceedings of the 17th Eurographics Symposium on Rendering*, pages 227–234, 2006.

[30] Gael Guennebaud, Loic Barthe, and Mathias Paulin. High Quality Adaptive Soft Shadow Mapping. *Computer Graphics Forum (Proceedings of Eurographics 2007)*, 2007.

[31] Johannes Günther, Heiko Friedrich, Ingo Wald, Hans-Peter Seidel, and Philipp Slusallek. Ray Tracing Animated Scenes using Motion Decomposition. *Computer Graphics Forum (Proceedings of Eurographics 2006)*, pages 517–525, 2006.

[32] Johannes Günther, Ingo Wald, and Philipp Slusallek. Realtime Caustics using Distributed Photon Mapping. In *Proceedings of the 14th Eurographics Symposium on Rendering*, pages 111–121, 2004.

[33] John Hammersley and David Handscomb. *Monte-Carlo Methods.* London: Chapman and Hall, 1964.

[34] Vlastimil Havran. *Heuristic Ray Shooting Algorithms.* Phd thesis, Czech Technical University, 2000.

[35] Paul 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.

[36] Wolfgang Heidrich and Hans-Peter Seidel. Realistic, Hardware-Accelerated Shading and Lighting. In *SIGGRAPH '99 (Proceedings of the 26th annual conference on Computer graphics and interactive techniques)*, pages 171–178, 1999.

[37] Stefan Heinrich. Monte Carlo Approximation of Weakly Singular Integral Operators. *Journal of Complexity*, pages 192–219, 2006.

[38] Henrik Wann Jensen. Global Illumination Using Photon Maps. In *Proceedings of the 7th Eurographics Symposium on Rendering*, pages 21–30, 1996.

[39] Henrik Wann Jensen. Rendering Caustics on Non-Lambertian Surfaces. In *Computer Graphics Forum (Proceedings of Eurographics 1997)*, pages 57–64, 1997.

[40] Henrik Wann Jensen. *Realistic image synthesis using photon mapping.* A. K. Peters, Ltd., 2001.

[41] Wing Hung Wong Jun S. Liu, Faming Liang. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, pages 121–134, 2000.

[42] James Kajiya. The Rendering Equation. In *SIGGRAPH '86 (Proceedings of the 13th annual conference on Computer graphics and interactive techniques)*, pages 143–150, 1986.

[43] Malvin Kalos and Paula Whitlock. *Monte-Carlo Methods.* New York: Wiley & Sons, 1986.

[44] Jan Kautz and Michael McCool. Approximation of Glossy Reflection with Prefiltered Environment Maps. In *Graphics Interface*, pages 119–126, 2000.

[45] Jan Kautz, Pere-Pau Vázquez, Wolfgang Heidrich, and Hans-Peter Seidel. Unified Approach to Prefiltered Environment Maps. In *Proceedings of the 10th Eurographics Symposium on Rendering*, pages 185–196, 2000.

[46] Csaba Kelemen, Lásló Szirmay-Kalos, György Antal, and Ferenc Csonka. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. In *Computer Graphics Forum (Proceedings of Eurographics 2002)*, pages 531–540, 2002.

[47] Alexander Keller. Instant Radiosity. In *SIGGRAPH '97 (Proceedings of the 24th annual conference on Computer graphics and interactive techniques)*, pages 49–56, 1997.

[48] Alexander Keller and Wolfgang Heidrich. Interleaved Sampling. In *Proceedings of the 12th Eurographics Symposium on Rendering*, pages 269–276, 2001.

[49] Alexander Keller and Ingo Wald. Efficient Importance Sampling Techniques for the Photon Map. In *Proceedings of Vision Modelling and Visualization*, pages 271–279, 2000.

[50] Gary King and William Newhall. Efficient Omnidirectional Shadow Maps. *ShaderX3*, pages 435–448, 2004.

[51] Tomas Kollig and Alexander Keller. Illumination in the Presence of Weak Singularities. *Monte Carlo and Quasi-Monte Carlo Methods*, pages 245–257, 2004.

[52] Lauwerens Kuipers and Harald Niederreiter. *Uniform Distribution of Sequences.* John Wiley and Sons, New York, 1974.

[53] Jaroslav Křivánek. *Radiance Caching for Global Illumination Computation on Glossy Surfaces.* PhD thesis, Université de Rennes 1 and Czech Technical University in Prague, 2005.

[54] Eric Lafortune. *Mathematical Models and Monte-Carlo Algorithms for Physically Based Rendering.* PhD thesis, University of Lueven, 1996.

[55] Eric Lafortune and Yves Willems. Bi-directional Path Tracing. In *Compugraphics '93 (Proceedings of Third International Conference on Computational Graphics and Visualization Techniques)*, pages 145–153, 1993.

[56] Samuli Laine, Hannu Saransaari, Janne Kontkanen, Jaako Lehtinen, and Timo Aila. Incremental Instant Radiosity for Real Time Indirect Illumination. In *Proceedings of the 18th Eurographics Symposium on Rendering*, pages 277–286, 2007.

[57] Ren Ng, Ravi Ramamoorthi, and Pat Hanrahan. All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation. In *SIGGRAPH '03 (Proceedings of the 30th annual conference on Computer graphics and interactive techniques)*, pages 376–381, 2003.

[58] Harald Niederreiter. *Random number generation and quasi-Monte Carlo methods.* Society for Industrial and Applied Mathematics, 1992.

[59] Art Owen and Seth Tribble. A Quasi-Monte Carlo Metropolis Algorithm. *Proc Natl Acad Sci U S A*, pages 8844–8849, 2005.

[60] Art Owen and Yi Zhou. Safe and Effective Importance Sampling. *Journal of the American Statistical Association*, pages 135–157, 2000.

[61] Mark Pauly, Thomas Kollig, and Alexander Keller. Metropolis Light Transport for Participating Media. In *Proceedings of the 11th Eurographics Symposium on Rendering*, pages 11–22, 2000.

[62] Ingmar Peter and Georg Pietrek. Importance Driven Construction of Photon Maps. In *Proceedings of the 9th Eurographics Symposium on Rendering*, pages 269–280, 1998.

[63] Matt Pharr. Chapter 9: Metropolis Sampling. In *Monte-Carlo Ray Tracing, SIGGRAPH'03 Course 44, course notes*, 2003.

[64] Matt Pharr and Greg Humphreys. *Physically Based Rendering : From Theory to Implementation.* Morgan Kaufmann, 2004.

[65] Ravi Ramamoorthi and Pat Hanrahan. An Efficient Representation for Irradiance Environment Maps. In *SIGGRAPH '01 (Proceedings of the 28th annual conference on Computer graphics and interactive techniques)*, pages 497–500, 2001.

[66] Ravi Ramamoorthi and Pat Hanrahan. Frequency Space Environment Map Rendering. In *SIGGRAPH '02 (Proceedings of the 29th annual conference on Computer graphics and interactive techniques)*, pages 517–526, 2002.

[67] William Reeves, David Salesin, and Robert Cook. Rendering Antialiased Shadows with Depth Maps. In *SIGGRAPH '87 (Proceedings of the 14th annual conference on Computer graphics and interactive techniques)*, pages 283–291, 1987.

[68] Alexander Reshetov, Alexei Soupikov, and Jim Hurley. Multi-Level Ray Tracing Algorithm. In *SIGGRAPH '05 (Proceedings of the 32th annual conference on Computer graphics and interactive techniques)*, pages 1176–1185, 2005.

[69] Reuven Rubinstein. *Simulation and the Monte Carlo Method*. New York: Wiley & Sons, 1981.

[70] Holly Rushmeier and Kenneth Torrance. The Zonal Method for Calculating Light Intensities in the Presence of a Participating Medium. In *SIGGRAPH '87 (Proceedings of the 14th annual conference on Computer graphics and interactive techniques)*, pages 293–302, 1987.

[71] Takafumi Saito and Tokiichiro Takahashi. Comprehensible Rendering of 3-D Shapes. In *SIGGRAPH '90 (Proceedings of the 17th annual conference on Computer graphics and interactive techniques)*, pages 197–206, 1990.

[72] Jörg Schmittler, Ingo Wald, and Philipp Slusallek. SaarCOR – A Hardware Architecture for Ray Tracing. In *Proceedings of the conference on Graphics Hardware 2002*, pages 27–36, 2002.

[73] Yuli Shreider. *The Monte-Carlo Method*. Oxford: Pergamon Press, 1966.

[74] Maryann Simmons and Carlo H. Séquin. Tapestry: A Dynamic Mesh-based Display Representation for Interactive Rendering. In *Proceedings of the 11th Eurographics Symposium on Rendering*, pages 329–340, 2000.

[75] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *SIGGRAPH '02 (Proceedings of the 29th annual conference on Computer graphics and interactive techniques)*, pages 527 – 536, 2002.

[76] Jérôme Spanier and Ely Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley, Reading, Massachussets, 1969.

[77] Jos Stam. Diffraction Shaders. In *SIGGRAPH '99 (Proceedings of the 26th annual conference on Computer graphics and interactive techniques)*, pages 101–110, 1999.

[78] Marc Stamminger and Georges Drettakis. Perspective Shadow Maps. In *SIGGRAPH '02 (Proceedings of the 29th annual conference on Computer graphics and interactive techniques)*, pages 557–562, 2002.

[79] Frank Suykens. *On Robust Monte-Carlo Algorithms for Multi-pass Global Illumination*. PhD thesis, University of Lueven, 2002.

[80] Frank Suykens and Yves Willems. Density Control for Photon Maps. In *Proceedings of the 11th Eurographics Symposium on Rendering*, pages 23–34, 2000.

[81] Jörg Schmittler Sven Woop and Philipp Slusallek. RPU: A Programmable Ray Processing Unit for Realtime Ray Tracing. In *SIGGRAPH '05 (Proceedings of the 22nd annual conference on Computer graphics and interactive techniques)*, pages 434–444, 2005.

[82] Justin Talbot. Importance Resampling for Global Illumination. Master's thesis, Brigham Young University, 2005.

[83] Justin Talbot, David Cline, and Parris K. Egbert. Importance Resampling for Global Illumination. In *Proceedings of the 16th Eurographics Symposium on Rendering*, pages 139–146, 2005.

[84] David Tannenbaum, Peter Tannenbaum, and Michael Wozny. Polarization and Birefringency Considerations in Rendering. In *SIGGRAPH '94 (Proceedings of the 21st annual conference on Computer graphics and interactive techniques)*, pages 221–222, 1994.

[85] Eric Veach. *Robust Monte-Carlo Methods for Light Transport Simulation.* PhD thesis, Standford University, 1997.

[86] Eric Veach and Leonidas Guibas. Bidirectional Estimators for Light Transport. In *Proceedings of the 5th Eurographics Symposium on Rendering*, pages 147–162, 1994.

[87] Eric Veach and Leonidas 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.

[88] Eric Veach and Leonidas Guibas. Metropolis Light Transport. In *SIGGRAPH '97 (Proceedings of the 24th annual conference on Computer graphics and interactive techniques)*, pages 65–76, 1997.

[89] Carsten Waechter and Alexander Keller. Instant Raytracing: The Bounding Interval Hierarchy. In *Proceedings of the 17th Eurographics Symposium on Rendering*, pages 139–149, 2006.

[90] Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination.* PhD thesis, Computer Graphics Group, Saarland University, 2004.

[91] Ingo Wald, Carsten Benthin, and Philipp Slusallek. Interactive Global Illumination in Complex and Highly Occluded Environments. In *Proceedings of the 14th Eurographics Symposium on Rendering*, pages 74–81, 2003.

[92] Ingo Wald, Carsten Benthin, Markus Wagner, and Philipp Slusallek. Interactive Rendering with Coherent Ray Tracing. In *Computer Graphics Forum (Proceedings of Eurographics 2001)*, pages 153–164, 2001.

[93] Ingo Wald, Solomon Boulos, and Peter Shirley. Ray Tracing Deformable Scenes using Dynamic Bounding Volume Hierarchies. *ACM Transactions on Graphics*, 2007.

[94] Ingo Wald, Johannes Günther, and Philipp Slusallek. Balancing Considered Harmful – Faster Photon Mapping using the Voxel Volume Heuristic. In *Computer Graphics Forum (Proceedings of Eurographics 2004)*, pages 595 –603, 2004.

[95] Ingo Wald and Vlastimil Havran. On Building Fast kd-trees for Ray Tracing, and on Doing that in O(N log N). In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, pages 61–69, 2006.

[96] Ingo Wald, Thiago Ize, Andrew Kensler, Aaron Knoll, and Steven G Parker. Ray Tracing Animated Scenes using Coherent Grid Traversal. In *SIGGRAPH '06 (Proceedings of the 33rd annual conference on Computer graphics and interactive techniques)*, pages 485–493, 2006.

[97] Ingo Wald, Tomas Kollig, Carsten Benthin, Alexander Keller, and Phillip Slusallek. Interactive Global Illumination using Fast Ray Tracing. In *Proceedings of the 13th Eurographics Symposium on Rendering*, pages 74–81, 2002.

[98] Ingo Wald, Philipp Slusallek, Carsten Benthin, and Markus Wagner. Interactive Distributed Ray Tracing of Highly Complex Models. In *Proceedings of the 12th Eurographics Symposium on Rendering Techniques*, pages 277–288, 2001.

[99] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P. Greenberg. Multidimensional Lightcuts. In *SIGGRAPH '06 (Proceedings of the 33rd annual conference on Computer graphics and interactive techniques)*, pages 1081–1088, 2006.

[100] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P. Greenberg. Lightcuts: a Scalable Approach to Illumination. In *SIGGRAPH '05 (Proceedings of the 32nd annual conference on Computer graphics and interactive techniques)*, pages 1098–1107, 2005.

[101] Gregory 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.

[102] Gregory Ward, Francis Rubinstein, and Robert 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.

[103] Lance Williams. Casting Curved Shadows on Curved Surfaces. In *SIGGRAPH '78 (Proceedings of the 5th annual conference on Computer graphics and interactive techniques)*, pages 270–274, 1978.

[104] Michael Wimmer, Daniel Scherzer, and Werner Purgathofer. Light Space Perspective Shadow Maps. In *Proceedings of the Eurographics 15th Symposium on Rendering*, pages 143–152, 2004.