Toward more realism and robustness in global illumination

Adrien Gruson

ANNÉE 2015

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Européenne de Bretagne

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Informatique

Ecole doctorale MATISSE

présentée par

Adrien Gruson

préparée à l'unité de recherche UMR 6074 IRISA
et au centre IRISA - Rennes Bretagne Atlantique
ISTIC

Toward more Realism and Robustness in Global Illumination

Thèse soutenue à Rennes le 6 Juillet 2015

devant le jury composé de :

Tamy BOUBEKEUR
Professeur à Télécom ParisTech / rapporteur

Elmar EISEMANN
Professeur à TU Delft / rapporteur

Luce MORIN
Professeur à INSA Rennes / examinateur

Jaroslav KŘIVÁNEK
Maître de Conférences à Charles Univ. / examinateur

Kadi BOUATOUCH
Professeur à Univ. de Rennes 1 / directeur de thèse

Rémi COZOT
Maître de Conférences à Univ. de Rennes 1 / co-directeur de thèse

Abstract

Using computer generated images has grown up for several years. For example, such images are used in the entertainment industry to produce contents (films, video games) or to preview future projects/prototypes. Generated images have several levels of realism. In this PhD, we focus on the generation of photo-realistic images by using physically based rendering processes.

Such images are often generated to achieve an artistic aim (atmosphere, aesthetics, information, etc.). To do so, the artist has several aspects to manage: the 3D scene itself and all the parameters used at the different steps necessary for generating images (rendering process, post production steps, etc.). The creation of computer graphics generated images can be a difficult and time consuming process. The aim of this thesis is to help the artist achieve his goals.

In the first part of this PhD, we focus on improving the rendering step. This step is responsible for the image generation by taking as input the 3D scene and several other parameters (number of samples, type of rendering technique, etc.). The aim of our work is to accelerate or make more robust this step so that it is easier for the artist to use.

In that perspective, we propose a new rendering algorithm that renders participating media on GPU. This algorithm is fast enough to make visualisation almost interactive. However, it only supports participating media, which limits its usage.

This is why we design a more general rendering algorithm that can handle a vast variety of 3D scenes. This rendering algorithm is robust and progressive (making preview of the final image possible). To achieve this, our technique improves (stochastic) progressive photon mapping (SPPM) by adding participating media support. Moreover, we use Metropolis sampling procedure to be able to get high efficiency on complex 3D scenes.

With this new algorithm, we have a robust general rendering algorithm. However, it still shares disadvantages with Metropolis sampling procedure: bad repartition of the relative error on the image plane. To address this issue, we propose a new importance function that distributes this error better. We propose two practical versions of this importance function for SPPM: an image-based formulation and a spatial one. Moreover, we use replica exchange and multiple importance sampling (MIS) to make this technique as robust as possible.

The second part of the thesis focuses on assisted tools for the artist. For example, we propose a new way to estimate the reference illuminant in 3D scenes. Then, this illuminant can be used to make white balancing or style-transfer during the post processing step. Another example of our work lies in the possibility of automatizing some lighting configurations in 3D scenes. Indeed, the lighting is important for the final image appearance. However, it is difficult for the artist to find adequate

parameters to configure the lighting. We propose a new algorithm that takes as an input the artist's intention (aesthetics). Then, the algorithm optimizes and finds the light sources configuration (size and flux) that matches the artist's wishes.

Acknowledgements

First of all, I would like to thank my two supervisors: Kadi Bouatouch and Rémi Cozot. Each of them helped me on their different areas of expertise. I would especially like to thank Kadi for his time and attention. I know that I am not patient and easy-going (at times) but you were always positive and helped me a lot.

I would also like to thank my co-authors/research partners: Jaroslav, Mickaël, Vincent, Charly, Ajit and Sumant. Special thanks to Jaroslav and Mickaël. Jaroslav, thanks for your time and all you taught me about research (I'm sorry I was helpless respecting deadlines). Moreover, I have enjoyed your perfectionism that pushed me beyond my limits. Mickaël, thanks to you too: you are more than a colleague, you are a friend. Thanks for your collaboration (almost on all my research projects) and your patience (contrary to me).

Many thanks as well to the FRVSense group (Ricardo, Billal, Matis, Hristina, Ronan, Mahmoud, Christian and Maryse). More globally, I also want to thank the graphics community and the people from the research lab.

Finally, I would like to thank all my relatives and close friends: François for the beers and the SC2 team game sessions (except that you still have a very weak level). These helped me relax! Thanks to my parents for their love, support, and for showing me that working is important. A special thank to Mathilde for her unconditional support and love throughout these four years.

Contents

List of Figures

List of Tables

Introduction 1

Computer generated images can achieve different levels of realism. In this thesis, we focus on photorealistic images obtained with the help of physics laws related to light propagation. However, these images are generated to meet an artist's aesthetic objective. One way to achieve this goal is to let the artist play with the different input parameters at his/her disposal (green boxes, fig. 1.1): the 3D scene itself and all the parameters needed during the different rendering steps (green boxes, fig. 1.1). Specifically, a 3D scene consists of: a virtual camera, a set of light sources, different 3D objects as well as the associated materials defining their appearance. A material describes how light is changed when it interacts with an object. All these variables (3D scene configuration and all other parameters) make it difficult for the artist to produce images that match his/her intent. A common solution is to use a time consuming trial-error process (fig. 1.1).

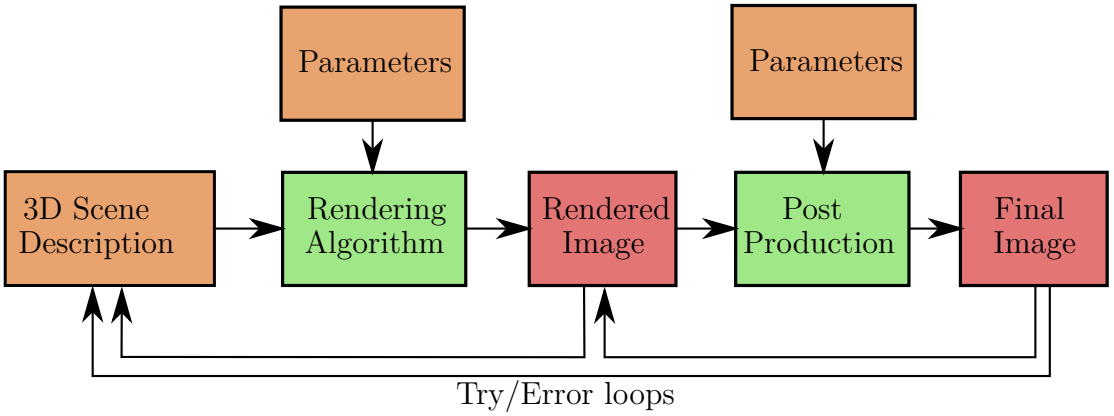


Figure 1.1 – The different steps to produce computer generated images. All these steps have several inputs (orange boxes) and generate images as output. However, the variety of parameters makes it difficult for the artist to generate an image that matches his/her intent. So, several iterations are required to generate an acceptable image.

The motivations of this PhD work is to simplify the artist's workflow. For that, we have focused on 2 main topics:

1. development of new rendering techniques that are more robust and faster. Robustness is needed so that the used rendering technique achieves good performances in all possible 3D scenes. As a consequence, the artist can use the same rendering technique for all his/her projects (which reduces potential errors of configuration). Moreover, a fast feedback to the user (*i.e.* progressive rendering) is required so that the user does not wait for the result for a long

time. However, when using progressive rendering a preview must be consistent with the final image quality.

2. development of new tools dedicated to the artist, for example a technique that automates a redundant task allowing saving time for the artist. Another example is a technique that extracts useful information that can be used during the post production step.

Organization of the dissertation

This manuscript is decomposed into three main parts:

1. **Part 1 – Background on GI:** physically-based rendering using physics laws to simulate light/matter interactions. In this part, we will introduce the mathematical model used for physically-based rendering (Chapter 2). Then, we define Monte Carlo estimator used to evaluate the rendering equation (Chapter 3). Finally, we will introduce Monte Carlo Markov Chain (MCMC) that uses a Metropolis algorithm to render complex 3D scenes in terms of geometry, visibility and light/matter interactions (Chapter 4).

2. **Part 2 – Efficient and robust rendering techniques:** First, we will present our new GPU algorithm to render participating media (Chapter 5). This algorithm allows almost interactive visualization for single and multiple scattering. Second, we will show how we have extended (stochastic) progressive photon mapping (SPPM) to handle participating media, placed in a 3D scene, together with their interaction with the scene's objects (Chapter 6). This technique is slower than the first one but is more general and robust. Indeed, it is able to handle any type of 3D scenes: different kinds of material, scene complexity, etc. Third, we will propose a new importance function for Metropolis-based rendering techniques that better distributes the relative error (Chapter 7). While the SPPM algorithm is robust as it uses Metropolis sampling procedure, it still has a major drawback: the error of the estimator is not evenly distributed over the image plane. Our new importance function addresses this issue by proposing two practical implementations: an image-based importance function and a spatial one.

3. **Part 3 – Computer-aided GI techniques for the artist:** This part contains the description of two methods aiming at helping the artist in his creation process. First, we will present our new method to estimate the main illuminant color for a 3D scene (Chapter 8). To do that, we estimate the lighting (average irradiance) arriving at the observer. This illuminant can be used during a post-production step to apply color transformations (white balancing, color transfer, style transfer, etc.). Second, we will present a new method to determine the lighting setup for a 3D scene (light source size and flux) (Chapter 9). Our technique takes as input the user intent and uses it to find a lighting configuration (setup) that matches the artist's desire.

For the reader For a reader familiar with global illumination, the chapters 1 and 2 can be skipped. Moreover, if the reader has a good background about metropolis rendering technique, the chapter 3 can be also skipped. The first three chapters do not bring any new contribution, they only help understanding the next chapters corresponding to our contributions.

1 Summary of the contributions

The work presented in this thesis brings the following contributions to the computer graphics field:

- a new rendering algorithm for participating media implemented on GPU;
- a visibility guided metropolis algorithm for photon mapping inside participating media;
- a new importance function for Metropolis rendering to evenly distribute the relative error of the estimator,
- a robust estimation of the reference illuminant in a 3D scene,
- a flexible automatic technique for determining the lighting setup to target a specific aesthetics desired by an artist.

2 Publications

Most of the work presented in this thesis is published in the following papers:

- **A. Gruson**, A. Hakke Patil, R. Cozot, K. Bouatouch and S. Pattanaik, "Light Propagation Maps on Parallel Graphics Architectures", Eurographics Symposium on Parallel Graphics and Visualization, 2012
- C. Collin, M. Ribardiere, **A. Gruson**, R. Cozot, S. Pattanaik and K. Bouatouch, "Visibility-driven progressive volume photon tracing", CGI 2013 and The Visual Computer: International Journal of Computer Graphics - Volume 29, Issue 9
- **A. Gruson**, R. Ribardiere, R. Cozot and K. Bouatouch, "Rendu Progressif basé Metropolis-Hasting dans des scènes à topologies multiples", AFIG 2014 and REFIG Vol. 8
- **A. Gruson**, R. Ribardiere and R. Cozot, "Eye-Centred Color Adaptation in Global Illumination", Pacific Graphics 2013 and Computer Graphics Forum, Volume 32 (2013), Number 7
- V. Leon, **A. Gruson**, R. Cozot and K. Bouatouch, "Automatic Aesthetics-based Lighting Design with Global Illumination", Pacific Graphics (Short paper), 2014

Part I

Background on Global Illumination

Introduction

In this part, we will summarize all the technical and mathematical details needed to produce realistic computer generated images. In particular, we will focus on physically based rendering. In physically based rendering, a physical model describes the interaction between light and the different elements of a 3D environment, often composed of:

1. light sources;
2. surfaces or/and volumes that interact with light stemming from light sources (reflection, refraction, scattering, absorption, etc.);
3. a virtual camera which represents the viewer.

Actually, light can be expressed by a flux emitted by light sources that progressively reaches an energy equilibrium. However, the speed of light is so fast that the energy equilibrium is reached instantly. The aim of physically based rendering techniques is to evaluate this equilibrium numerically so as to compute a final image. This equilibrium can be expressed as an integral equation. In practice, generating an image consists in finding light paths that start from light sources, interact several times with the scene, and finally reach the camera. However, determining these light paths is difficult because of the complexity of the different light interactions. In chapter 2, we will present the physical models used to describe the different light interactions.

The set of possible light paths is infinite, an approximation is then needed. One solution is to use Monte Carlo methods that randomly create light paths in the scene (chapter 3). Then, the solution is to average the contributions of the randomly sampled paths to produce an image. This solution is elegant and can provide high quality images. However, the efficient creation of valid light paths is crucial when it comes to produce noiseless images. This generation can be difficult in a complex 3D environment with complex materials or difficult visibility.

Intensive research has been done to build efficient strategies to construct valid paths. Some of them are targeted to handle special scenes or certain light phenomena, others combine several techniques to handle different kinds of phenomena. At the end, the rendering technique can be difficult to implement and not easy for use (e.g. a lot of parameters). One simple solution is to extract some knowledge about the scene by using some previous sample paths. This solution is often built upon a Markov Chain model and uses the Metropolis-Hasting algorithm [MRR+53, Has70] to efficiency produce light paths (chapter 4).

Mathematical and Physical models 2

Light propagates in a 3D space Light can reach (from an incoming direction) or leave (in an outgoing direction) a surface. It can also be emitted, reflected, scattered or absorbed by taking into account incident/outgoing directions. So, for this reason, before going into the definitions of physically light quantities (radiometric quantities), we first review some mathematical concepts in the 3D space. Indeed, light propagation can be expressed in the direction domain (direction of scattering) or in the surface domain. In this section, we will review the surface and direction domain formulations and their associated measures to use them in the rest of the manuscript. Then, we will show how to move on from one domain to another.

Surface domain In a 3D environment, we assume that the scene geometry consists of a finite set of surfaces in $\mathbb{R}^3$. The union of all the surfaces is denoted $\mathcal{M}$. Its measure is denoted $\mathrm{d}A(\vec{x})$ at the surface point $\vec{x}$. Moreover, each surface point has an associated normal $\vec{n}(\vec{x})$ (written $\vec{n}$ for simplicity) that describes the surface orientation in a 3D space.

Directional domain This domain is important because, in physically based rendering, a lot of sampling decisions are taken in it. Each direction is represented by a normalized vector $\vec{\omega} \in \mathbb{R}^3$. The domain of direction originating from a surface point $\vec{x}$ can be divided in two parts. First, we can define the space of the upper hemispheres:

$$\Omega_+ = \{\vec{\omega} : |\vec{\omega}| = 1, \vec{\omega} \cdot \vec{n} \geq 0\} \tag{2.1}$$

Second, the other part of the sphere (the lower hemisphere) is defined as:

$$\Omega_- = \{\vec{\omega} : |\vec{\omega}| = 1, \vec{\omega} \cdot \vec{n} \leq 0\} \tag{2.2}$$

In general, in computer graphics, we are often interested only in the upper hemisphere that receives incoming light (except in volume rendering where the notion of normal is absent, to this end the domain of direction is the total sphere). For simplicity, in the rest of the manuscript, we will use Ω for the space of direction. Moreover, we have the equality:

$$\cos\theta = \vec{\omega} \cdot \vec{n} \tag{2.3}$$

where the angle θ is the polar angle between the direction $\vec{\omega}$ and a surface normal $\vec{n}$. For simplicity, we use an absolute operator in order not to take into account the orientation of these two vectors.

In the directional domain, the measure is a solid angle and is denoted $\mathrm{d}\sigma(\vec{\omega})$ for a given direction $\vec{\omega}$. A solid angle is the equivalent in 3D of an angle defined in a 2D space. Solid angles are expressed in *steradians*.

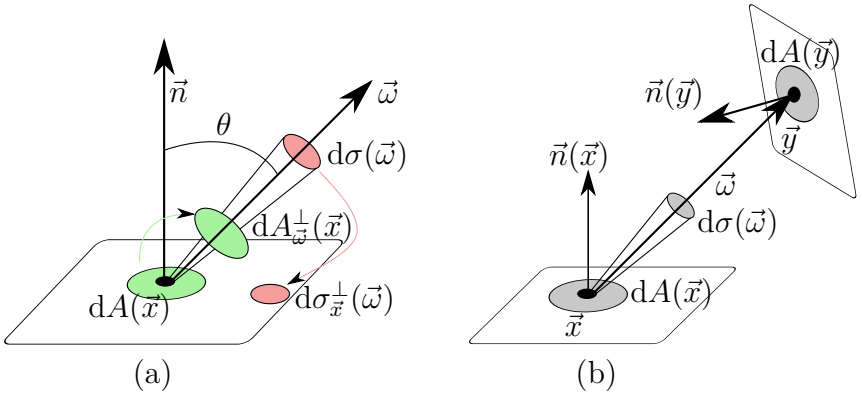


Figure 2.1 – (a) The different projections possible for different measures defined in the surface (in green) or directional (in red) spaces. (b) Surface $\mathrm{d}A(y)$ subtended by solid angle $\mathrm{d}\sigma(\vec{\omega})$. With measure transformation, it is possible to express the solid angle $\mathrm{d}\sigma(\vec{\omega})$ by using $\mathrm{d}A(y)$ (see eq. (2.6)).

Projection and domain transformation Each measure (in the surface ($\mathrm{d}A$) or in the directional ($\mathrm{d}\sigma$) domain) is equivalent to a measure after projection. For example, to project a solid angle onto a surface, we use the polar angle θ between the direction $\vec{\omega}$ and the normal $\vec{n}$:

$$\mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}) = \cos\theta \mathrm{d}\sigma(\vec{\omega}) \tag{2.4}$$

And the inverse process is possible, to project a surface onto a solid angle:

$$\mathrm{d}A^{\perp}_{\vec{\omega}}(\vec{x}) = \cos\theta \mathrm{d}A(\vec{x}) \tag{2.5}$$

These two projection operations are shown in fig. 2.1 (a). In some part of the manuscript, we will express the integral problem in the projected solid angle domain. To express this domain, we will use the symbol $\Omega^{\perp}$.

Moreover, if we want to combine different models expressed in different spaces, we need to express them in the same space. This situation occurs often in rendering where some decisions are made in the directional domain and others in the surface domain. However, it is possible to change the measure from the directional to the surface domain, by transformation from the solid angle $\vec{\omega}$ to the projected point $\vec{y}$:

$$\mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}) = \frac{\mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega})}{\mathrm{d}A(\vec{y})}\mathrm{d}A(\vec{y}) = G(\vec{x}\leftrightarrow\vec{y})\mathrm{d}A(\vec{y}) \tag{2.6}$$

where $G(\vec{x}\leftrightarrow\vec{y})$ is the geometry factor that corresponds to the Jacobian relating solid angle to area. This factor is expressed as:

$$G(\vec{x}\leftrightarrow\vec{y}) = \frac{|\vec{n}(\vec{x})\cdot\vec{\omega}| \times |\vec{n}(\vec{y})\cdot\vec{\omega}|}{||\vec{x}-\vec{y}||^2} V(\vec{x}\leftrightarrow\vec{y}) \tag{2.7}$$

Note the apparition of the term $V(\vec{x}\leftrightarrow\vec{y})$ which expresses the visibility between $\vec{x}$ and $\vec{y}$. In physically based rendering, different decisions are expressed in different domains (direction or surface).

1 Radiometric quantities

Light is an electromagnetic radiation and it could be measured with physical quantities called radiometric quantities. Moreover, by using these quantities correctly, we come up naturally to the rendering equation that is used in rendering. This equation expresses the light transport problem as an integral evaluation problem.

Flux (or radiant power), notated Φ, is the fundamental quantity which expresses the total light energy Q received or emitted per unit of time:

$$\Phi = \frac{\mathrm{d}Q}{\mathrm{d}t}.$$

The unit is watt (W). This quantity will be fundamental as all the next quantities are derived from it.

Irradiance notated E, is the flux received per unit surface area:

$$E(\vec{x}) = \frac{\mathrm{d}\Phi}{\mathrm{d}A(\vec{x})}. \tag{2.8}$$

The unit is Watt per square meter ($W \cdot m^{-}2$). A relation between this quantity and the flux is given by:

$$\Phi = \int_S E(\vec{x})\mathrm{d}A(\vec{x}). \tag{2.9}$$

where S is a surface. Note that the same quantity exists for emission and is named radiosity or emittance.

Radiance notated L, is the flux emitted by a surface, per unit project area, per unit solid angle. The radiance emitted at the point $\vec{x}$ into the direction $\vec{\omega}$ will be notated $L(\vec{x}\rightarrow\vec{\omega})$.

For emitted radiance, the differential expression (see fig. 2.1) is:

$$L(\vec{x}\rightarrow\vec{\omega}) = \frac{\mathrm{d}^2\Phi}{\mathrm{d}\sigma(\vec{\omega})\mathrm{d}A(\vec{x})\cos\theta}. \tag{2.10}$$

where $\mathrm{d}^2\Phi$ is the differential flux of the light emitted at point $\vec{x}$, θ is the angle between the normal $\vec{n}$ and the direction $\vec{\omega}$. This quantity is important because it expresses the light as we perceive it (from a surface into a direction). The radiance is independent of the distance. Moreover, there exists some relationship between radiosity and emitted radiance (eqs. (2.4) and (2.5)):

$$L(\vec{x}\rightarrow\vec{\omega}) = \frac{\mathrm{d}E(\vec{x}\rightarrow\vec{\omega})}{\mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega})} \tag{2.11}$$

There also exists a notion of incident radiance. It is the radiance incident from a direction $\vec{\omega}$ at a point $\vec{x}$. It will be denoted $L(\vec{x}\leftarrow\vec{\omega})$. In this case, the differential flux comes from another surface.

Luminance is a photometric value that is equivalent to radiance (radiometric value). Luminance is equal to radiance up to a factor that is the human eye response. This is why, in the rest of the manuscript, we will use both of them to refer to the same quantity.

2 Surface interaction

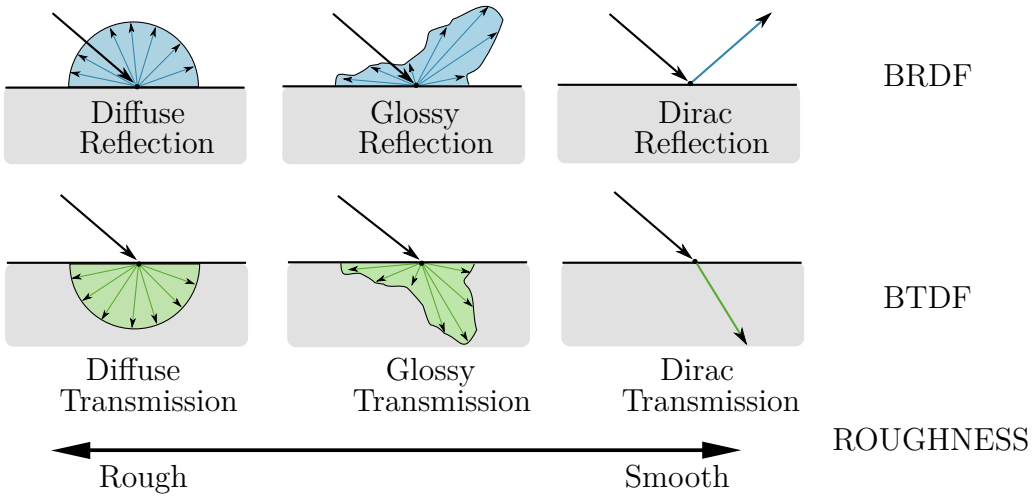


Figure 2.2 – In general there are two main types of material for the surfaces: Reflective material (BRDF) or transmissive material (BTDF). Transmission obeys Snell's law. Moreover, for these two types, different interactions are possible: Diffuse, Glossy and Dirac. These interactions depend on the roughness of the surface. Diffuse interaction concerns an extremely rough surface. On the contrary, a completely smooth surface creates one single light reflection.

Now that we have defined the physical quantities of light, let us see how light interacts with surfaces. To do that, we describe how radiometric quantities change with this interaction. Different surface interactions are possible and summarized in fig. 2.2. All these interactions are handled by a general model named BSDF (*bidirectional scattering distribution function*). This model includes two light phenomena: reflection and transmission. BSDF is approximated by several mathematical models [TS67, ON94, WMLT07]. For more information, the reader can refer to the PBRT book [PH10]. Moreover, note that BSDF models make an approximation: the incoming position and the outgoing position of light are the same. A more general model is expressed by a BSSRDF model [JMLH01]. However, for simplicity, we will focus our presentation only on BSDF model, termed f_{r}. Moreover, to be physically correct, a BSDF model needs to meet two important constraints:

1. *Helmholtz reciprocity principle:* for every pair of direction $\vec{\omega}_i$ and $\vec{\omega}_o$, we have $f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) = f_{\mathrm{r}}(\vec{x}, \vec{\omega}_o \to \vec{\omega}_i)$

2. *Energy conservation:* for all the directions $\vec{\omega}_o$, the total energy of light reflected must meet the following constraint:

$$\int_{\Omega^\perp} f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) \mathrm{d}\sigma^\perp_{\vec{x}}(\vec{\omega}_i) \leq 1.$$

A BRDF is expressed as (we use eq. (2.11) for quantity change) a luminance change of the luminance coming from $\vec{\omega}_i$ toward $\vec{\omega}_o$:

$$f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) = \frac{\mathrm{d}L(\vec{x} \to \vec{\omega}_o)}{\mathrm{d}E(\vec{x} \leftarrow \vec{\omega}_i)} = \frac{\mathrm{d}L(\vec{x} \to \omega_o)}{L(\vec{x} \leftarrow \vec{\omega}_i)\mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}_i)}. \tag{2.12}$$

where $\vec{\omega}_i$ is the incident direction, $\vec{\omega}_o$ is the outgoing direction and $L(\vec{x} \to \vec{\omega}_o)$ the incident radiance at point $\vec{x}$. We can integrate this equation and express the outgoing radiance in the direction $\vec{\omega}_o$:

$$\begin{aligned} \mathrm{d}L(\vec{x} \to \vec{\omega}_o) &= f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) L(\vec{x} \leftarrow \vec{\omega}_i) \mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}_i) \\ L(\vec{x} \to \vec{\omega}_o) &= \int_{\Omega^{\perp}} f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) L(\vec{x} \leftarrow \vec{\omega}_i) \mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}_i). \end{aligned} \tag{2.13}$$

In physically based rendering, we need to compute the outgoing radiance of a surface viewed through the camera. It is possible to express it using eq. 2.13. To do that, we need to include the light source emission L_e, which gives *the rendering equation* [Kaj86]:

$$L(\vec{x} \to \vec{\omega}_o) = L_e(\vec{x} \to \vec{\omega}_o) + \int_{\Omega^{\perp}} f_{\mathrm{r}}(\vec{x}, \vec{\omega}_i \to \vec{\omega}_o) L(\vec{x} \leftarrow \vec{\omega}_i) \mathrm{d}\sigma^{\perp}_{\vec{x}}(\vec{\omega}_i) \tag{2.14}$$

where $L(\vec{x} \to \vec{\omega}_o)$ is the luminance viewed by the camera and $L(\vec{x} \leftarrow \vec{\omega}_i)$ the incident radiance. Note that in the integral, the incident radiance $L(\vec{x} \leftarrow \vec{\omega}_i)$ is unknown. To estimate it, we need to cast a ray from the position $\vec{x}$ in the direction $\vec{\omega}_i$. At the intersection point, we need to evaluate again the same integral. This amounts to a recursive evaluation of this integral. By applying measure changes using eq. (2.6), the rendering equation can be expressed in the surface domain $\mathcal{M}$ as:

$$L(\vec{x} \to \vec{\omega}_o) = L_e(\vec{x} \to \vec{\omega}_o) + \int_{\mathcal{M}} f_{\mathrm{r}}(\vec{x}, (\vec{x} - \vec{y}) \to \vec{\omega}_o) L(\vec{x} \leftarrow \vec{y}) G(\vec{x} \leftrightarrow \vec{y}) \mathrm{d}A(\vec{y}) \tag{2.15}$$

where the incident direction is replaced $\vec{\omega}_i = (\vec{x} - \vec{y})$ and $L(\vec{x} \leftarrow \vec{y})$ the incident radiance emitted by the point $\vec{y}$. Note that a visibility term is included inside the geometry factor.

Previous formula only mention about radiance computation. However, a computer generated image is a 2D array of pixel. Each pixel is oversampled into points $\vec{x}'$ (*i.e.* to solve the aliasing problem or to reduce noise), which corresponds to a set of view rays, each of them passing through a point $\vec{x}'$. These rays may intersect the scene at a point $\vec{x}$. At each point $\vec{x}$, the reflected radiance $L(\vec{x} \to \vec{x}')$ needs to be evaluated. To compute the radiance viewed from a pixel j, an integration over this pixel is needed:

$$I_j = \int_{\mathcal{M} \times \mathcal{M}} W_e^{(j)}(\vec{x}' \to \vec{x}) L(\vec{x}' \to \vec{x}) G(\vec{x}' \leftrightarrow \vec{x}) \mathrm{d}A(\vec{x}') \mathrm{d}A(\vec{x}) \tag{2.16}$$

where $\vec{x}$ lies on the pixels in the image space and $W_e^{(j)}(\vec{x} \to \vec{x}')$ represents the emitted camera importance for the pixel j. For example, this term includes the filtering operation done on the pixel side.

To evaluate the equation 2.15, it is difficult to generate paths of different lengths efficiently. Indeed, light can bounce multiple times before it reaches the camera.

In the model, this is shown in eq. (2.15) where L is on the two expression side. A general formulation was introduced by Veach [Vea97]. It uses the surface domain to define the rendering equation problem for a given pixel j:

$$I_j = \int_{\mathbb{P}} f_j(\overline{\mathbf{x}}) \mathrm{d}\mu(\overline{\mathbf{x}}), \tag{2.17}$$

where $\mathbb{P}$ is the path space, $\overline{\mathbf{x}} = \vec{x}_0\vec{x}_1 \cdots \vec{x}_k$ a path defined by k vertices and $\mathrm{d}\mu(\overline{\mathbf{x}})$ the measure. The measure is the product of measures in the surface domain for each $\vec{x}_i$:

$$\mathrm{d}\mu(\overline{\mathbf{x}}) = \prod_{l}^{k} \mathrm{d}A(\vec{x}_0). \tag{2.18}$$

The contribution function $f(\overline{\mathbf{x}})$ can be expressed as the product of the BSDF, the emission of the light sources and the sensor response:

$$f_j(\overline{\mathbf{x}}) = L_e(\vec{x}_0 \to \vec{x}_1) T(\overline{\mathbf{x}}) W_e^j(\vec{x}_{k-1} \to \vec{x}_k) \tag{2.19}$$

$$T(\overline{\mathbf{x}}) = G(\vec{x}_0 \leftrightarrow \vec{x}_1) \prod_{i=1}^{k-1} f_r(\vec{x}_{i-1} \to \vec{x}_i \to \vec{x}_{i+1}) G(\vec{x}_i \leftrightarrow \vec{x}_{i+1}) \tag{2.20}$$

where $W_e^j(\vec{x}_{k-1} \to \vec{x}_k)$ is the sensor response and $T(\overline{\mathbf{x}})$ the throughput of the given path that includes the geometry factor and the BSDF values. An overview of this formulation is given in fig. 2.3.

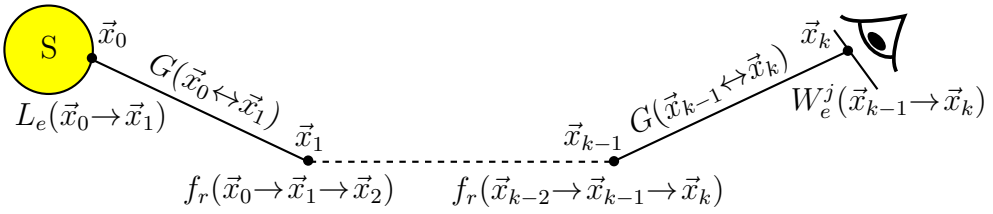


Figure 2.3 – Throughput figure of one path using Veach formulation.

However, the path integral includes all the possible length paths. One possibility is to split different path lengths to get different integrals:

$$I_j = \sum_{k=1}^{\infty} \int_{M}^{k+1} f_j(x_0 \dots x_k) \mathrm{d}A(x_0) \dots \mathrm{d}A(x_k) \tag{2.21}$$

These different integrals over surfaces (eq. (2.21)) need to be evaluated. They are restricted to surface interaction only. However, light can interact with more complex objects such as smoke, liquid, called participating media. These phenomena introduce light volume interactions, which we need to take into account and add to our mathematical formulation.

3 Volume interaction

Volume interactions are due to participating media (such as smoke, fire, clouds, ice, dust, etc.) that interact with light. Participating media are important for realism

in the film industry. However, integrating such media is computationally expensive. Indeed, their integration domain has one more dimension than surfaces. The intersection between a ray and media is a line and not a point (fig. 2.5). Interaction can occur at each point associated with this intersection line, which makes the computation expensive.

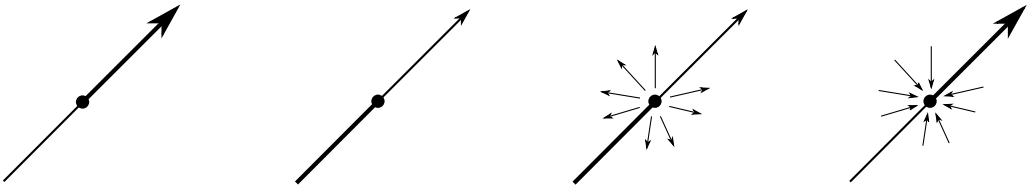


Figure 2.4 – Different interaction between the light and the participating media.

In such media, different light interactions can occur (fig. 2.4):

- *Absorption and out-scattering:* called also *extinction*, these two phenomena are responsible for the energy lost inside the media.
- *In scattering:* is the light scattered in the view direction. Indeed, at each point in the media, light can come from every direction and be partially scattered in the view direction.
- *Emission:* self emission (*i.e.* light source) in the view direction in a media such as fire.

All theses interactions are expressed by the *radiative transfer equation* (RTE) [Sub60]. This equation expresses the change of light at a position $\vec{y}$ inside the volume and in the direction $\vec{\omega}$:

$$\frac{\mathrm{d}L(\vec{y}\rightarrow\vec{\omega})}{\mathrm{d}\vec{y}} = \sigma_a(\vec{y})L_e(\vec{y}\rightarrow\vec{\omega}) + \sigma_s(\vec{y})L_i(\vec{y}\rightarrow\vec{\omega}) - \sigma_t(\vec{y})L(\vec{y}\rightarrow\vec{\omega}) \tag{2.22}$$

where L_e is the self-emitted radiance and L_i the luminance due to the incident luminance that scatters into $\vec{\omega}_i$. Moreover, $\sigma_a(\vec{y})$ is the absorption coefficient at the position $\vec{y}$, $\sigma_s(\vec{y})$ the scattering coefficient and $\sigma_t(\vec{y}) = \sigma_s(\vec{y}) + \sigma_a(\vec{y})$ the extinction coefficient. For homogeneous media, these coefficients remain constant at each point. Otherwise the media are called heterogeneous. Incident light L_i at the position $\vec{y}$ includes all the possible incoming directions:

$$L_i(\vec{y}\rightarrow\vec{\omega}) = \int_\Omega \rho(\vec{y}, \vec{\omega}_i\rightarrow\vec{\omega})L(\vec{y}\leftarrow\vec{\omega}_i)\mathrm{d}\sigma(\vec{\omega}_i) \tag{2.23}$$

where, Ω is the sphere domain and $\rho(\vec{y}, \vec{\omega}_i\rightarrow\vec{\omega})$ is the phase function that is similar to the BSDF describing the proportion of radiance coming from the direction $\vec{\omega}_i$ and reflected in the direction $\vec{\omega}$. Moreover, similarly to the coefficients, the phase function can vary or not inside the media. There exists different models of phase function. For more information, the reader can refer to the book of Engel et al. [HKRs+06].

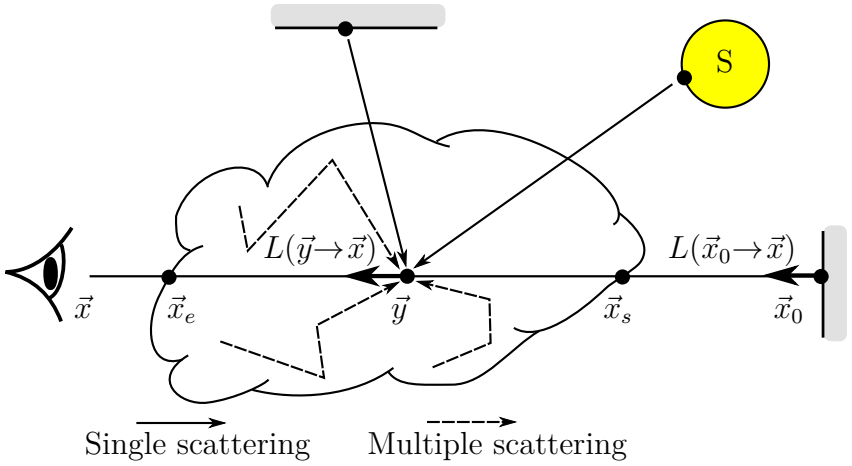


Figure 2.5 – The incident luminance inside the participating media can bounce only once (single scattering) or multiple times (multiple scattering).

However, we need to express the RTE (eq. (2.22)) in an integral form to be able to evaluate it (section 3). For example, we want to evaluate the total energy received by a viewer $L(\vec{x}\leftarrow\vec{\omega})$ from the position $\vec{x}$ in the direction $\vec{\omega}$. To do that, we need to integrate the light interactions along the line between the intersection of the view ray and the medium. For this purpose, we define an entry point $\vec{x}_e$ and an exit point $\vec{x}_s$. Then, we integrate the light interactions over the line using the RTE (eq. (2.22)). Moreover, we have, if it is the case, to take into account the back surface at the position $\vec{x}_0$. All these considerations are shown in fig. 2.5 and expressed as:

$$\begin{aligned} L(\vec{x}\leftarrow\vec{\omega}) &= \int_{\vec{x}_e}^{\vec{x}_s} \tau(\vec{x}_e\leftrightarrow\vec{y})\sigma_a(\vec{y})L_e(\vec{y}\rightarrow\vec{\omega})\mathrm{d}\vec{y} \\ &+ \int_{\vec{x}_e}^{\vec{x}_s} \tau(\vec{x}_e\leftrightarrow\vec{y})\sigma_s(\vec{y})\left[\int_\Omega \rho(\vec{y},\vec{\omega}_i\rightarrow\vec{\omega})L(\vec{y}\leftarrow\vec{\omega}_i)\mathrm{d}\sigma(\vec{\omega}_i)\right]\mathrm{d}\vec{y} \\ &+ \tau(\vec{x}_e\leftrightarrow\vec{x}_s)L(\vec{x}_0\rightarrow\vec{\omega}) \end{aligned} \tag{2.24}$$

where $\tau(\vec{x}_e\leftrightarrow\vec{y}) = e^{-\beta(\vec{x}_e\rightarrow\vec{y})}$ is the transmittance that represents physically the amount of distance in the medium before the light scatters and β is the optical thickness. The higher the value, the more the volume is opaque.

$$\beta(\vec{x}_e\rightarrow\vec{y}) = \int_{\vec{x}_e}^{\vec{y}} \sigma_t(\vec{y})\mathrm{d}\vec{y}$$

Now that we have defined the volume interaction as an integral problem, we want to express it in the path domain. It is totally possible to generalize the path framework, introduced by Veach [Vea97], to include participating media by changing

the throughput formulation:

$$T(\overline{\mathbf{x}}) = \prod_{i=j}^{k-1} f(\vec{x}_i) G(\vec{x}_i \leftrightarrow \vec{x}_{i+1}) V_{\text{att}}(\vec{x}_i \leftrightarrow \vec{x}_{i+1}) \tag{2.25}$$

$$f(\vec{x}_i) = \begin{cases} f_r(\vec{x}_{i-1} \rightarrow \vec{x}_i \rightarrow \vec{x}_{i+1}) & \text{when } \vec{x}_i \text{ is on a surface} \\ \rho(\vec{x}_{i-1} \rightarrow \vec{x}_i \rightarrow \vec{x}_{i+1}) & \text{when } \vec{x}_i \text{ is in a medium} \end{cases} \tag{2.26}$$

$$G(\vec{x} \leftrightarrow \vec{y}) = \frac{D_x(\vec{x}) D_y(\vec{y})}{||\vec{x} - \vec{y}||^2} \tag{2.27}$$

$$V_{\text{att}}(\vec{x} \leftrightarrow \vec{y}) = \tau(\vec{x} \leftrightarrow \vec{y}) V(\vec{x} \leftrightarrow \vec{y}) \tag{2.28}$$

where $D_x(\vec{x})$ is the projection operation that is a cosine term if $\vec{x}$ is on a surface and 1 if $\vec{x}$ is in the medium. V_{att} is the attenuated visibility term that includes the regular visibility term $V(\vec{x} \leftrightarrow \vec{y})$ extracted from the geometric term and multiplied by the transmittance of the medium. Note that if there is no medium between $\vec{x}$ and $\vec{y}$ the transmittance value is equal to 1.

Monte Carlo solutions 3

In the previous chapter, we have defined physically based rendering as a problem of integral evaluation (eq. (2.21)). In this section, we will detail how to evaluate this integral efficiently. Several approaches are possible. Monte Carlo based approaches are good candidates to solve this problem because of their simplicity and flexibility.

For these reasons, we will detail this approach in the following sections. First, we will review the general formulation of the Monte Carlo estimator and its properties. Second, we will present some improvements such as importance sampling or multiple importance sampling. Finally, we will show how to use it in physically based rendering.

1 General formulation

Before giving the Monte Carlo estimator, we need to define some mathematical basis. The *expected value* $E_p[\frac{f(x)}{p(x)}]$, where $p(x)$ is the probability density function (pdf), is equal to the integral value of a function $f(x)$ over a domain Ω:

$$\int_\Omega f(x)\mathrm{d}x = \int_\Omega \frac{f(x)}{p(x)}p(x)\mathrm{d}x = E_p[\frac{f(x)}{p(x)}] \tag{3.1}$$

The Monte Carlo estimator is an estimate of the expected value. However, to compute the integral of $f(x)$, the Monte Carlo estimates the expected value of ${}^{f(x)}/_{p(x)}$. Given a set of N uniform random variables $x_i \in \Omega$, the Monte Carlo estimator F_N gives:

$$E_p[\frac{f(x)}{p(x)}] \approx F_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(x_i)}{p(x_i)} \tag{3.2}$$

where $p(x_i)$ is the probability to generate x_i. The simplest density probability is the uniform density. When using the Monte Carlo approach, the pdf $p(x)$ must be different from zero for all x where $|f(x) > 0|$. Note that, for uniform pdf, this condition is always fulfilled.

For Monte Carlo, variance corresponds to the estimation error. The aim is to reduce this variance as much as possible. The variance of the estimator F_N is defined as:

$$V[F_N] = E\left[(F_N)^2\right] - E[F_N]^2 \tag{3.3}$$

Estimator properties A Monte Carlo estimator F_N is consistent if it converges to the correct solution with an infinite number of samples. This can be expressed by the following formula:

$$lim_{N\to\infty}\mathrm{Prob}[(F_N - \int f(x)\mathrm{d}x) = 0] = 1 \tag{3.4}$$

A Monte Carlo estimator F_N is unbiased if the expected value of the estimate is equal to the correct solution:

$$\int f(x)\mathrm{d}x - E[F_N] = 0 \tag{3.5}$$

Note that an unbiased estimator is not automatically consistent. Indeed, some rendering algorithms use a pre-computation step to evaluate some values. For example, these values can be the overall brightness of the image plane, etc. These values are then used by an unbiased rendering algorithm. However, as these values are not refined, the resulting algorithm can get unbiased but non-consistent.

Infinite dimension of integration In the eq. (2.21), a path can bounce a large number of times. However, fixing a maximum number of bounces (dimension of the integral) makes the estimator biased. One solution is to use a Russian roulette approach that will put a probability q to stop the path. However, the throughput of the path needs to be scaled by the probability to continue a divided by $1 - q$. This technique makes the estimator unbiased:

$$E[I] = (1 - q)\left(\frac{E[I]}{1 - q}\right) + q = E[I] \tag{3.6}$$

This technique can increase variance of the estimator but keeps it unbiased.

Samples placement To increase the efficiency of a Monte Carlo estimator, we need to reduce its variance using sampling strategies such as:

- *Stratified sampling:* it covers the domain of integration better. The idea of this technique is to partition the domain and run a different Monte Carlo estimator for each stratum. The final estimator is a weighted sum of the estimates over all the domains.
- *Importance sampling:* use a pdf p proportional to the integrand. Indeed, the function $f(x)/p(x)$ will be more flattened than $f(x)$. This technique reduces variance and is often used in global illumination.

Note that this list is not exhaustive. Different variance reduction techniques can be combined to achieve lower variance. In physically based rendering, importance sampling is a common technique to reduce the variance. This approach is presented in detail in the next section.

2 Importance sampling

2.1 General framework

The key idea is to choose a pdf p proportional to the integrand f. The more the shape of the pdf is similar to the function, the more the variance is reduced. The

extreme case corresponds to a pdf proportional to the function, *i.e.* $c \cdot p(x) = f(x)$, where c is a constant. In this special case, the variance of the estimator is zero:

$$\frac{1}{N} V[\frac{f(x)}{p(x)}] = \frac{1}{N} V[\frac{1}{c}] = 0$$

However, this is not feasible in practice. The main reason is because the constant factor (which is unknown) is equal to:

$$c = \frac{1}{\int_\Omega f(x) \mathrm{d}x}$$

that is exactly the integral we are trying to evaluate. Moreover, we need to generate samples with the pdf proportional to $f(X)$, which could be an issue. For instance, in physically based rendering, the integrand (eq. (2.21)) is a product of several terms. Some terms are unknown (*i.e.* the incident radiance), which makes harder the determination of the CDF (cumulative distribution function used to sample proportional to a pdf (fig. 3.1)).

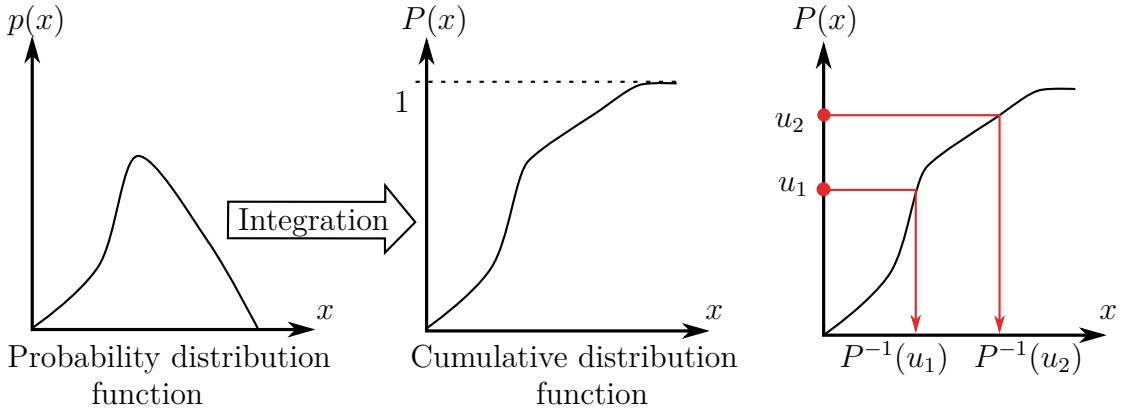


Figure 3.1 – CDF is computed by integrating a PDF. Then, we project uniform samples ($u1$ and $u2$) using the inverse CDF. The projected samples are distributed proportionally to the PDF.

Bad importance sampling (shapes of $p(x)$ and $f(x)$ are very different) could lead to variance higher than that obtained with uniform sampling. Determining a pdf $p(x)$ (for a good importance sampling) for a high dimension integration domain is a difficult task. One solution is to create this pdf $p(x)$ using different pdf $p_i(x)$. Each one defined on a sub-domain.

2.2 Multiple distributions

Constructing $p(x)$ using several pdf to sample the function $f(x)$ may be a good idea at first glance. But the problem is how to use the different sample distributions to get a lower variance. Indeed, averaging over the different sampling strategies can lead to an extra variance (proof in eq. (3.8)). One elegant solution is to combine the

different estimators with a Multiple Importance Sampling estimator. This estimator is defined as follows:

$$F = \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} w_i(x_{i,j}) \frac{f(x_{i,j})}{p_i(x_{i,j})}, \tag{3.7}$$

where n is the number of sampling strategies and n_i is the number of samples allocated to each strategy. $x_{i,j}$ is the j^{th} sample from the distribution p_i. Each sample is assigned a weight w_i. In other words, this formula is the weighted sum of the different estimators $f(x_{i,j})/p_i(x_{i,j})$. To obtain an unbiased estimator, the weighting function w_i must meet two conditions:

1. $\sum_{i=1}^{n} w_i(x) = 1$ for $f(x) \neq 0$,
2. $w_i(x) = 0$ whenever $p_i(x) = 0$

These conditions imply that the set of the sampling techniques need to sample where $f(x) \neq 0$. However, one sampling technique p_i does not need to sample the whole domain, but only one sub-domain.

Now, we need to discuss the choice of the weighting function w_i. Indeed, suppose that we have three pdf p_1, p_2 and p_3, and only one sample is taken for each one. This leads to the following estimator:

$$F = w_1 \frac{f(x_{1,1})}{p_1(x_{1,1})} + w_2 \frac{f(x_{2,1})}{p_2(x_{2,1})} + w_3 \frac{f(x_{3,1})}{p_3(x_{3,1})}. \tag{3.8}$$

If the weighting function is constant and one sampling strategy is bad, then F will have variance as well, since:

$$V[F] = w_1 V[F_1] + w_2 V[F_2] + w_3 V[F_3]$$

Different weighting strategies are discussed by Veach [Vea97]. One possible weighting strategy is the *power heuristic*:

$$w_i(x) = \frac{(n_i p_i(x))^\beta}{\sum_k (n_k p_k(x))^\beta} \tag{3.9}$$

In case of $\beta = 1$, we have another weighting strategy called *balance heuristic*. The idea behind these heuristics is to assign a bigger weight to the sampling strategy with higher probability. Using the formulation of the balance heuristic, we can rewrite the global estimator as:

$$\begin{aligned} F &= \sum_{i=1}^{n} \frac{1}{n_i} \sum_{j=1}^{n_i} \left(\frac{n_i p_i(x_{i,j})}{\sum_k n_k p_k(x_{i,j})} \right) \frac{f(x_{i,j})}{p_i(x_{i,j})} \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n_i} \frac{f(x_{i,j})}{\sum_k n_k p_k(x_{i,j})} \\ &= \frac{1}{N} \sum_{i=1}^{n} \sum_{j=1}^{n_i} \frac{f(x_{i,j})}{\sum_k c_k p_k(x_{i,j})} \end{aligned}$$

where $c_k = n_k/N$. We can see that this formulation clearly expresses a Monte Carlo estimator accounting for several pdf.

Conclusion We have seen in this section how to use Monte Carlo estimator to evaluate the integral. Multiple importance sampling proved to be an efficient way for reducing the variance of the Monte Carlo estimator. This technique is commonly used in physically based rendering. However, we still need to define the set of sampling strategies. In the next subsection, we will discuss the different uses of the Monte Carlo estimator in rendering.

3 Practical aspects

In this section, we present the different ways of using Monte Carlo to solve the light transport equation (expressed in section 2). As we have seen in the previous section, the pdf $p(x)$ needs to mimic the integrand to reduce the variance of the estimator. In rendering, the variance is perceptible as noise in the rendered image. To produce high quality images, this noise has to be below a certain threshold in order not to be perceptible by the user.

To build the pdf $p(x)$, different strategies are possible in physically based rendering. The aim of rendering is to find contributive light paths that connect to the camera. To introduce different sampling strategies, first we introduce a sub-problem in light transport equation that is direct rendering. This problem considers one light bounce only. Due to low dimensionality, this problem is easy to solve with Monte Carlo. It is easy to understand the different sampling strategies and their respective performances. So, at the beginning of this section, we describe a practical implementation for direct rendering. This implementation uses all the techniques described in the Monte Carlo section (importance sampling, multiple importance sampling).

Then, we focus on physically based rendering with several light bounces. We present classical unbiased estimators to evaluate the light transport equation (eq. (2.21)): path tracing, light tracing and bidirectional path tracing. Moreover, we discuss their respective advantage/drawback compared the other rendering techniques.

All unbiased techniques get inefficient for certain sampling scenarios. An example of these scenarios corresponds to the situation where the light integrand domain reduces to a small path domain. For example, this is the case when a light caustic is viewed through a smooth mirror. To address this issue, we introduce biased rendering techniques that are more robust than unbiased ones.

The convergence rate (variance decreases with the number of samples) is different for these two classes of techniques. Unbiased techniques have often better convergence rate than biased ones. On the other hand, biased techniques are more robust to evaluate the path space. Recent research focuses on combining these two classes of techniques. We review them at the end of this section.

Note that in this section, we will consider mainly the surface integration (without participating media). However, in some parts of this section, some methods will be evoked when it comes to render participating media.

3.1 Direct rendering

Direct rendering problem can be reduced to a visibility problem. It amounts to determine the visibility between the light sources and the surfaces viewed through the camera. Only one light bounce is considered. We can express the problem using the directional domain (similar to eq. (2.14)) but we restrict the incoming radiance to the radiance $L_e()$ emitted from the light source and arriving at point $\vec{x}$:

$$L(\vec{x}\rightarrow\vec{\omega}_o) = \int_{\Omega^\perp} f_\mathrm{r}(\vec{x},\vec{\omega}_i\rightarrow\vec{\omega}_o)L_e(\vec{x}\leftarrow\vec{\omega}_i)\mathrm{d}\sigma^\perp_{\vec{x}}(\omega_i), \tag{3.10}$$

where $\vec{x}$ is a point on a surface viewed through the camera and $\vec{\omega}_o$ is the view direction. Moreover, we have restricted the incident radiance $L_e(\vec{x}\leftarrow\vec{\omega}_i)$ to the radiance emitted from the light sources.

The integrand is equal to the product of two terms: the BSDF f_r and the incident direct lighting $L_{e,i}$. As we have seen before, constructing a sampling strategy that includes the two terms is challenging. However, it is possible to construct a sampling strategy by considering only one term: sampling according to the BSDF or sampling according to the radiance of the light source.

Sampling according to the BSDF To sample the BSDF, the incident direction $\vec{\omega}_i$ is randomly sampled according to a PDF $p(\vec{\omega}_i)$. This PDF is proportional to the BSDF value:

$$p(\vec{\omega}_i) \propto f_\mathrm{r}(\vec{x},\vec{\omega}_i,\vec{\omega}_o),$$

where p is expressed using projected solid angles. In practice, to render an image, we first trace a ray from the camera through a given pixel. This ray may intersect the 3D scene at a point $\vec{x}$. Then we choose an incident potential light direction $\vec{\omega}_o$ according to the BSDF and we trace a ray from the point $\vec{x}$ in this direction. For an intersection point $\vec{x}'$, we evaluate the emitted radiance $L_e(\vec{x}\leftarrow\vec{x}')$. For this sampling strategy, the Monte Carlo estimator is:

$$L(\vec{x}\rightarrow\vec{\omega}_o) \approx \frac{1}{N}\sum_{j=1}^{N}\frac{f_\mathrm{r}(\vec{x},\vec{\omega}_j\rightarrow\vec{\omega}_o)L_{e,i}(\vec{x}\leftarrow\vec{\omega}_j)}{p(\vec{\omega}_j)}, \tag{3.11}$$

where N is the number of samples. Note that this sampling strategy does not take into account the incident radiance from the light sources, so only one term of eq. (3.10) is considered.

Sampling according to the emitted radiance To be able to sample the light sources, we need to express the eq. (3.10) in the surface domain using the domain transformation expressed in eq. (2.6):

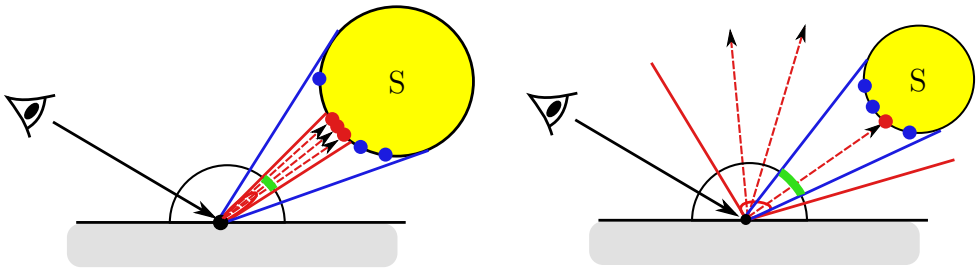
$$L(\vec{x}\rightarrow\vec{x}') = \int_{\mathcal{M}_l} f_\mathrm{r}(\vec{y}\rightarrow\vec{x}\rightarrow\vec{x}')L_e(\vec{x}\leftarrow\vec{y})G(\vec{x}\leftrightarrow\vec{y})\mathrm{d}A(\vec{y}). \tag{3.12}$$

where $\mathcal{M}_l$ is the surface domain of the light sources and $\vec{y}$ a point on a light source. In practice, this strategy starts similarly to the previous one. First, we trace a ray

from the camera through a given pixel (at position $\vec{x}'$). This ray may intersect the 3D scene at a point $\vec{x}$. Then we randomly sample a position on the light source $\vec{y}$ and evaluate the visibility to $\vec{x}$. If $\vec{x}$ is visible from $\vec{y}$, then we evaluate the BSDF value $f_r(\vec{y} \to \vec{x} \to \vec{x}')$. For this sampling strategy, the Monte Carlo estimator is:

$$L(\vec{x} \to \vec{\omega}_o) \approx \frac{1}{N} \sum_{j=1}^{N} \frac{L_e(\vec{x} \leftarrow \vec{y}_j) G(\vec{x} \leftrightarrow \vec{y}_j)}{p(\vec{y}_j)} f_r(\vec{y}_j \to \vec{x} \to \vec{x}') \tag{3.13}$$

where $p(\vec{y}_j)$ is the pdf used to sample a point $\vec{y}_j$ on the light source surfaces and N the number of samples.



(a)
Smooth BSDF
Large light sources

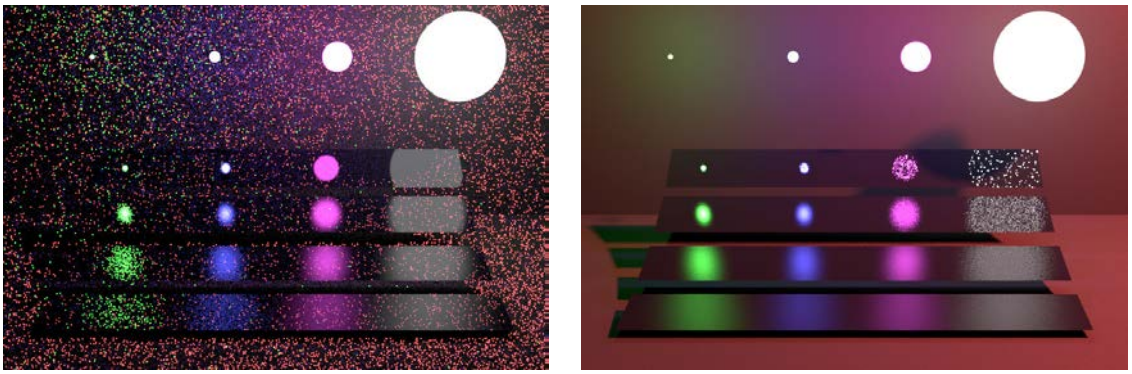
(b)
Rought BSDF
Small light sources

Figure 3.2 – This figure explains the difference between the efficiencies of the sampling procedures graphically. The two sampling technique are shown: red dots represent the BSDF samples while the blue dots the light source samples. Green domain expresses the directional contributive direction domain. In this particular domain, the BSDF and the incident luminance are non zero. (a) BSDF sampling performs better than emitter sampling. Indeed, emitter samples are non contributive due to the BSDF value. (b) Emitter sampling performs better than BSDF sampling.

Discussion Figure 3.3 shows the results of the different sampling strategies in a simple scene [Vea97]. In this scene, we have four light sources with different sizes and colors. Moreover, there are four rectangles with different BSDF roughness. The BSDF is smoother when a rectangle is far away from the camera.

In this scene, the different sampling strategies generate a noise, the level of which varies over the image. BSDF sampling gets better for smooth BSDF. Indeed, in the directional space, the BSDF has a smaller space than the light source (fig. 3.2, case (a)). In this case, light sampling can find a valid path but with a zero BSDF value. On the other hand, with a small light source or a non directional BSDF there is less chance to hit a light source after a bounce according to the BSDF (fig. 3.2, case (b)). In conclusion, the two strategies correspond to different sampling scenarios (light source size, BSDF roughness). By combining them one can get a more robust sampling strategy called: Multiple Importance Sampling (MIS).

Multiple importance sampling In section 2.2, we have presented the multiple importance sampling estimator that turns out to be robust but computationally



(a) BSDF sampling **(b)** Emitter sampling

Figure 3.3 – Images rendered with different sampling strategies. If we exclude the diffuse background, we can observe that the two sampling strategies are complementary.

expensive. Indeed, for each sample, we need to evaluate the $p_i(x)$ for all the sampling strategies. However, this extra computation is negligible compared to the ray tracing operation.

To be able to combine the two strategies with MIS, we need to express them in the same domain. Indeed, the BSDF sampling $p(\vec{\omega}_i)$ is expressed according to projected solid angle and the light source sampling is expressed in the surface domain. One possibility is to express all of them in the surface domain using the following transformation (based on eq. (2.6)):

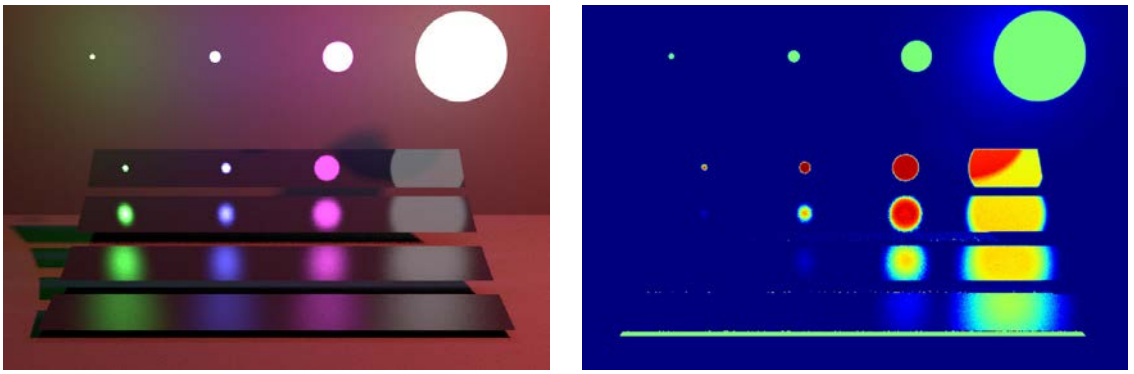
$$p(\vec{y}) = p(\vec{\omega}_i)G(\vec{x}\leftrightarrow\vec{y}) \tag{3.14}$$

where $\vec{\omega}_i$ is the solid angle associated with the direction $\vec{x}\rightarrow\vec{y}$.

Figure 3.4 shows a result obtained with multiple importance sampling. Note that the same number of samples is used for producing the images of fig. 3.3. Moreover, without extra knowledge, we have used the power heuristic (eq. (3.9), $\beta = 2$) with an equal number of samples for each sampling strategy. In other words, each sampling procedure (BSDF or emitter) received twice fewer samples. Assigning the same number of samples to each strategy could generate additional noise (*i.e.* on the diffuse background). The reason is, in regions where only one sampling strategy works (pdf proportional to the integrand), this strategy receives fewer samples than without MIS. Recent research has been done to have a better weighting scheme [PBPP11]. Moreover, in this sub-section, we have only presented two sampling techniques (BSDF and emitter). More sampling strategies are possible. For more information, the reader can refer to Shirley et al. article [SWZ96].

3.2 Indirect rendering with unbiased estimator

Unlike direct rendering, physically based rendering accounts for multiple light bounces. Light paths composed of more than one light bounce are called *indirect paths*. Creating an efficient sampling strategy to sample these paths is challenging. Figure 3.5 shows how indirect lighting is important for scene perception.



(a) Combined sampling **(b)** Relative contribution

Figure 3.4 – On the left: The combined results using multiple importance sampling with balance heuristic. These results have less overall variance than previous results. In areas where only one strategy works, there is extra noise. This is because it uses twice fewer samples than the previous rendered images. On the right: the relative weighted contribution of each sampling technique shown with false colors. Blue color means that the light sampling is more efficient than BSDF sampling. On the contrary, red color means that BSDF sampling is better. Green color means that the two strategies perform equivalently.

(a) Direct rendering **(b)** Indirect rendering **(c)** Combination of the two

Figure 3.5 – In this figure, we show the different rendered images in case of direct or indirect rendering. In the direct rendered image, several shadows are due to the light position. This component has discontinuities and some parts of the scene do not receive any lighting. The indirect component is relatively smooth and quite constant in the scene. This rendering generates more noise than the direct one because of the difficulty of computation. The combination of both produces an image of better quality.

Computing the indirect lighting component can be time consuming. Moreover, a rendering algorithm needs to find a path (of several interactions) that connects the camera to a light source. This kind of path is often called *contributive paths*. There are several reasons why a path is not contributive: zero BSDF value, visibility problem, etc. There exists several techniques to build these contributive paths. All these techniques have their drawbacks and advantages. For this reason, we rapidly present hereafter different techniques that efficiently build such paths: *path*

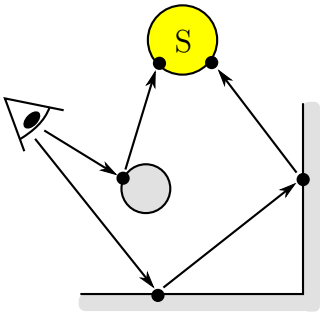
tracing, *light tracing* and *bidirectional path tracing*. We also show how much multiple importance sampling is crucial for robust rendering technique.

3.2.1 Path tracing

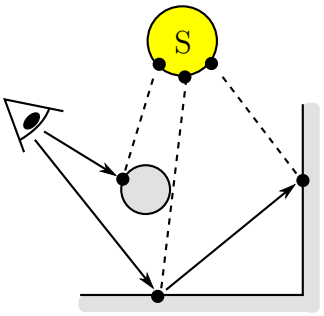
Path tracing is often used in rendering engines (Arnold renderer, Cycles, Octane renderer, PRMan, etc.). It builds a light path from the camera to the light sources incrementally. At the beginning, for each pixel, a number of paths are traced to evaluate the incoming lighting (eq. (2.16)). This algorithm makes possible to control the sampling rate for each pixel. This is a desirable property because controlling the variance allows to control the noise level in the produced image. By equally sampling all the pixels, the algorithm spreads the variance smoothly in the image space. This is important for producing noiseless images until a certain threshold, because the human visual system can perceive the noise.

After throwing the primary ray from the camera and finding the first intersection point, two different path tracing techniques can be used. In the first one, named *primitive path tracing*, a ray continues to bounce until it hits a light source. This is similar to BSDF sampling for direct rendering because it only relies on this strategy. So, this estimator shares the same drawback than direct rendering with only BSDF sampling. In case of a small light source, this estimator will have a high variance.

The second way to implement a path tracing is to use explicit light connection. Similarly to the primitive path tracing, new vertices (path nodes) are generated according to the BSDF. But, at each intersection point, the direct component is evaluated using an emitter sampling technique. Note that when MIS is not used, the fact of hitting randomly a light source (by the bouncing procedure) does not make a contribution. This is because we cannot have two ways of sampling the same component (direct) without combining them.



Primitive path tracing



Path tracing with
explicit light source connection

Figure 3.6 – The two ways of implementing path tracing. On the left: primitive path tracing will bounce until it randomly hits a light source. On the right: the technique will use explicit source light connections. Each new vertices is connected to a randomly sampled point in the light source. A ray is traced to check the visibility between this point and the new vertex.

These two techniques are described in fig. 3.6. In the same spirit as direct rendering, we have two different sampling strategies. In the same way, we can use

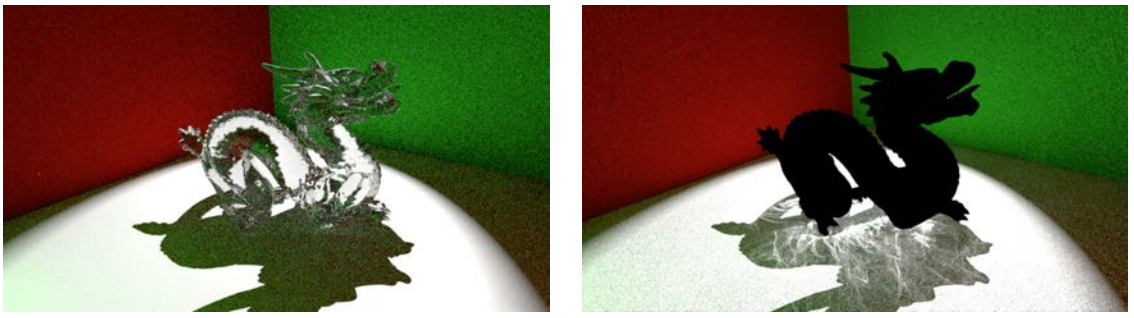
multiple importance sampling to combine them.

3.2.2 Light tracing

Unlike path tracing, *Light tracing* builds a contributive path from the light sources to the camera. To do that, this technique starts by randomly sampling a point on the light sources as well as an outgoing direction. A new ray is traced in this direction and the process is repeated iteratively. Moreover, similarly to path tracing, there are two ways of building such a path.

The first technique, called *primitive light tracing*, keeps on tracing new rays until reaching the camera. This technique relies only on BSDF sampling and can be inefficient. This method fails if a path does not hit the camera. This is because a pinhole camera model is often used.

The second solution, more elegant, is *light tracing with explicit connection to the camera*. Similarly to the primitive technique, light bounces according to the BSDF. Each intersection point is projected into the image plane. For this operation, visibility needs to be evaluated by a ray tracing operation. If the point is visible, then the contribution of the current path is accumulated.



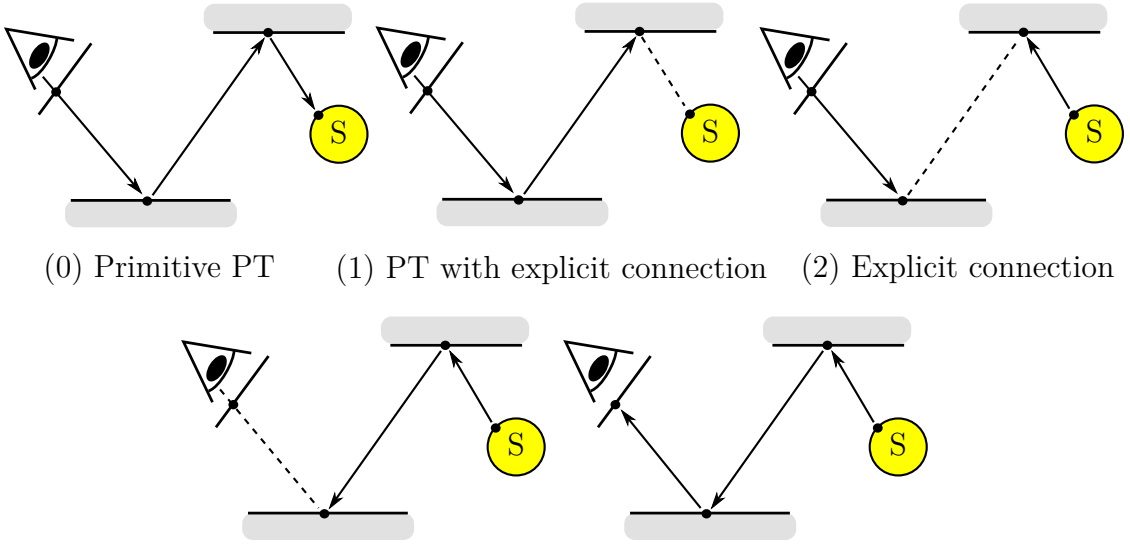
(a) Path tracing **(b)** Light tracing

Figure 3.7 – In this scene, a glass dragon is lighted by a spot light. Because the spot light is not intersected by any ray, path tracing technique is not able to render caustics. Except for this component, path tracing technique achieves good results. For the same number of samples, light tracing can create images with less noise. Moreover, light tracing makes it possible to render caustics. However, light tracing fails to render the dragon material because the dirac produced on its BSDF makes the connection to the camera impossible.

These two strategies can be combined with multiple importance sampling. However, intrinsically this technique suffers from a big issue: no control of the sampling rate in the image space is offered. The paths are created proportionally to the lighting, so, a dark area in the image space receives fewer samples. At the end, the noise is not well distributed, and dark areas remains noisy. However, this rendering method is not useless because of the efficiency of sampling certain sets of paths such as light caustics created by small light sources. Figure 3.7 shows the results obtained with path tracing and light tracing in this case.

3.2.3 Bidirectional Path tracing

We can observe that the two above rendering techniques are complementary. Moreover, for each of these techniques, a multiple importance sampling scheme is used to combine different paths. So, the question is: is it possible to use MIS to combine these two rendering techniques? Note that these two strategies do not cover all the possible paths. For example, there are 5 different ways of building a path of two bounces (fig. 3.8). More generally, a path with k bounces can be built according to $k + 3$ different ways.



(0) Primitive PT (1) PT with explicit connection (2) Explicit connection

(3) LT with explicit connection (4) Primitive LT

Figure 3.8 – This figure enumerates the different ways to build a path with two light bounces. Note that path and light tracing (described previously) is a subset of BDPT. Moreover, the number before the technique name is the number of light path vertices. The middle technique (2 light vertices) is a new one where there is an explicit connection between the two interaction points.

This rendering technique of building all the sets of paths is called *bidirectional path tracing* (BDPT). This approach was proposed by Lafortune et al. [LW93] for the original idea. Veach et al. [VG95] added an optimal weighting strategy based on MIS space later on. In practice, BDPT constructs two different paths, one originating from the camera, the other from the light. These paths are constructed incrementally, similar to path tracing or light tracing. Each time a new vertex (path node) is sampled, an explicit connection between these two paths is performed. In a non-optimized version of BDPT, only one connection is made between the end vertices of the two paths. In an optimized version of BDPT, the new sampled vertex of one path type (camera or light) is connected to all the vertices of the other path. This makes it possible to reuse all the previous sampled vertices efficiently and to build more paths with fewer rays.

Moreover, each time a connection is possible, we need to evaluate the weight used in MIS. This is done by computing the pdf of all possible ways of constructing this path. These pdf can be computed efficiently using a recursive formula. Moreover,

this computation is negligible compared to the ray tracing operation.

At the end, the intensive usage of multiple importance sampling makes BDPT robust. In case of simple interior scenes, BDPT can provide better results than the previous described techniques. This is due to its ability to sample strong indirect lighting paths efficiently. Some results are shown in fig. 3.9. More optimization tools are given in Veach's thesis [Vea97].

(a) Path tracing

(b) Light tracing

(c) Bidirectional path tracing (60 spp)

Figure 3.9 – In this figure, we compare BDPT and the two other techniques for the same rendering time. In this scene, there is strong indirect lighting. With explicit connection, BDPT can render the scene rapidly. Path tracing has more samples per pixel, but doesn't have this sampling technique. This lack results to more noise in the path tracing rendered image.

Discussion All the previous described techniques sample the path domain. These techniques are unbiased because no approximation is made when sampling the path space. This space is a high dimensional space, a small part of which represents the contributive paths. Indeed, a large part of this space is not contributive because of surface occlusion or zero BSDF values.

However, some parts of the contributive path domain can be very small. In extreme cases, this can lead to a point in the path domain. This point represents a certain class of light paths: *non-separable light paths*. Unbiased techniques are not able to sample this type of path.

This issue can be addressed by extending the contributive path domain. This is done by allowing the creation of physically impossible light paths. For example, this can be done by allowing the connection of two neighbour vertices lying on the same surface. This operation would blur the integrand domain and make it possible to sample non-separable light paths. But, with this operation, the estimator will have a bias. Different biased rendering techniques exist and will be described in the next sub-section.

3.3 Indirect rendering with biased estimator

Contrary to an unbiased estimator, a biased estimator entails a systematic error on the estimate value. The bias $B(F_N)$ of an estimator F_N is expressed as:

$$B(F_N) = \int f(x)\mathrm{d}x - E[F_N]$$

If $B(F_N) = 0$, the estimator is unbiased. There are several biased estimators. But here, we will only focus on biased rendering techniques more robust than path-based techniques (path tracing, light tracing, BDPT, etc.). This excludes, for example, virtual point light techniques using clamping.

One famous biased rendering technique is *photon mapping* [Jen01]. It consists of two steps:

1. *Photon shooting step*: Similarly to light tracing, light paths are emitted from the light sources. For each light path, there are several intersection points with the 3D scene (path node). For each intersection, if the surface is rough enough (diffuse), then a photon is created. Several sets of data are assigned to a photon: incident direction, position and flux. Then, this photon is stored in a spatial data structure, named *photon map*, which is used in the next step.

2. *Rendering step*: Rays are traced from the camera. At each interaction point (path node), if the surface is rough enough, a request is sent to the photon map to gather photons close to the interaction point. Using these photons, the incident radiance at the interaction point is estimated. For a surface not sufficiently rough, the camera path continues to bounce similarly to the path tracing technique.

Figure 3.10 shows an example of a difficult scene setting (a lot of smooth materials and small light sources). In this scene, BDPT is inefficient because it cannot connect two points (due to the specular material). However, photon mapping is able to create contributive paths. Note that this gathering process makes it possible to reuse several light paths. This possibility is really important for rendering performance. To underline the robustness of the photon mapping against BDPT, a rendering result is shown in fig. 3.11.

However, photon mapping has several limitations. First, a lot of photons are necessary to produce a noiseless image. Several improvements have been proposed to produce better images with the same amount of photons [Jen95, Chr03]. However, increasing the number of photons to generate noiseless images gets impractical

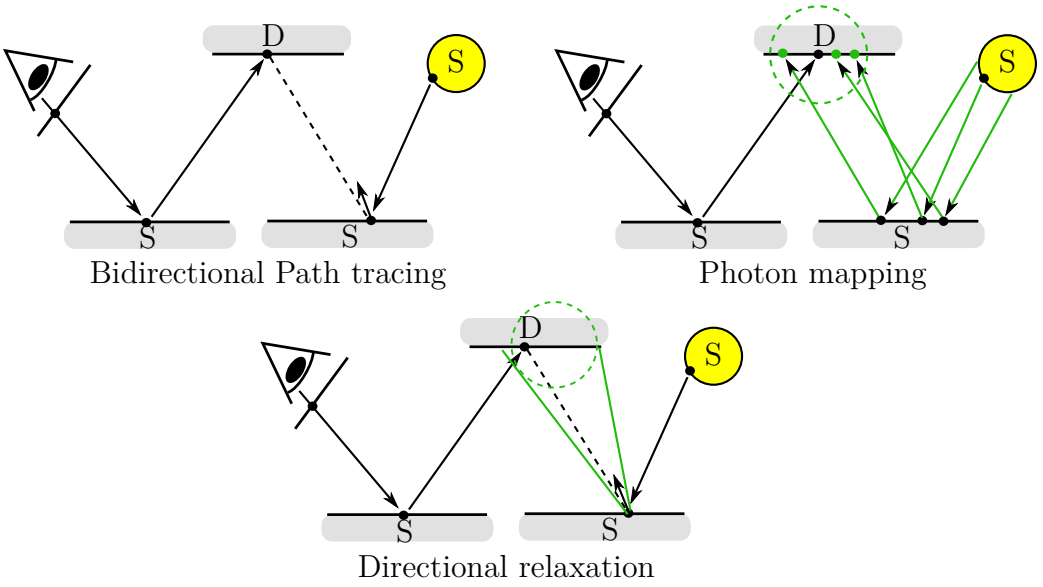
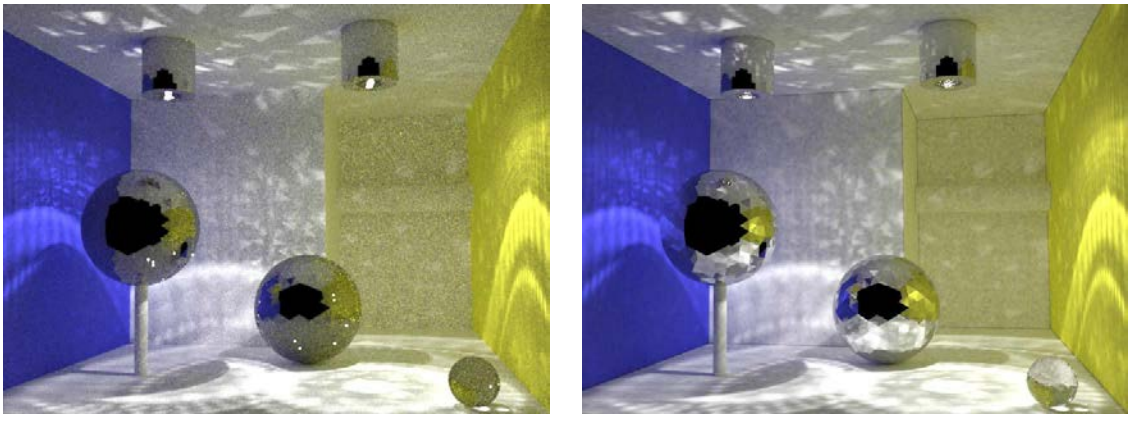


Figure 3.10 – Schematic explanation on the photon mapping robustness. In case of smooth interactions (Surfaces denoted S) and only a diffuse one (Surface denoted D), BDPT cannot connect two vertices due to the BSDF of S. With the gathering procedure, photon mapping is able to build several contributive paths (green dots represent the photons). Directional relaxation makes possible the connection, by modifying the smooth BSDF (green cone). However, in this latter technique, it is not possible to reuse several light paths.



(a) BDPT **(b)** Photon mapping

Figure 3.11 – This scene has several settings that makes BDPT inefficient: light sources are included in a glass probe and there is a lot of specular BSDF. This makes inefficient the explicit connection of two sampled vertices. As a result, photon mapping can sample efficiently certain parts of a domain for which BDPT completely fails.

because of the memory limitation. This last issue can be addressed by a multi-pass approach. All these improvements will be discussed in the following sections.

The photon mapping technique relies on a spatial relaxation. Similar techniques, named *Directional relaxation* [KD13b, BIOP13], use the same principle but expressed in the directional domain. These techniques change the BSDF expression to

expend it to a non zero value domain. This type of technique has several advantages (compared to photon mapping): it does not need to store photons and enables connections between two smooth surfaces. However, their main drawback is the lack of light path reuse because these techniques do not use a gathering procedure.

3.3.1 Photon mapping

The photon mapping approach [Jen01] is based on density estimation expressed in the surface space. To be able to recover the photon mapping formula, let us recall the rendering equation (eq. (2.14) without self emission) :

$$L_r(\vec{x}\rightarrow\vec{\omega}) = \int_\Omega f_r(x,\vec{\omega}'\rightarrow\vec{\omega})L_i(\vec{x}\leftarrow\vec{\omega}')\mathrm{d}\sigma^\perp_{\vec{x}}(\vec{\omega}') \tag{3.15}$$

By rewriting the incident radiance (eq. (2.10)) to make appear the incident flux, we get:

$$\begin{aligned} L_r(\vec{x}\rightarrow\vec{\omega}) &= \int_\Omega f_r(x,\vec{\omega}'\rightarrow\vec{\omega})\frac{\mathrm{d}^2\Phi(\vec{x},\vec{\omega}')}{\mathrm{d}\sigma^\perp_{\vec{x}}(\vec{\omega}')\mathrm{d}A(\vec{x})}\mathrm{d}\sigma^\perp_{\vec{x}}(\vec{\omega}') \\ L_r(\vec{x}\rightarrow\vec{\omega}) &= \int_\Omega f_r(x,\vec{\omega}'\rightarrow\vec{\omega})\frac{\mathrm{d}^2\Phi(\vec{x},\vec{\omega}')}{\mathrm{d}A(\vec{x})} \end{aligned} \tag{3.16}$$

The incoming flux Φ is approximated by the set of photons gathered using the photon map. Each photon represents a proportion $\Delta\phi_p(\vec{\omega}_p)$ of the total emitted flux. By using photons near $\vec{x}$, we obtain the following estimator:

$$L_r(\vec{x}\rightarrow\vec{\omega}) \approx \sum_{p=1}^{n} f_r(\vec{x},\vec{\omega}_p\rightarrow\vec{\omega})\frac{\Delta\phi_p(\vec{\omega}_p)}{\Delta A}, \tag{3.17}$$

where n is the number of photons gathered near $\vec{x}$, $\Delta\phi_p$ the flux carried by a photon and $\vec{\omega}_p$ its incident direction. However, this formula is only expressed in a local domain. To express this estimator with the total set of emitted photons, we use a density kernel with a given support. ΔA expresses the support of the photon density kernel. Moreover, photon mapping assumes that the neighbourhood of $\vec{x}$ is a flat surface (for the kernel normalization factor). By doing that, we can rewrite the estimator as:

$$L_r(\vec{x},\vec{\omega}) \approx \frac{1}{M}\sum_{p=1}^{M} \mathrm{k_r}(\vec{x}-\vec{x}_p)\psi_p, \tag{3.18}$$

where M is the total number of emitted photons, $\mathrm{k_r}$ the kernel used to accumulate the photons and ψ_p the photon contribution. This contribution includes the flux carried by the photon times the BSDF value. In case of constant kernel with a fixed radius r, we have:

$$L_r(\vec{x},\vec{\omega}) \approx \frac{1}{\pi r^2 M}\sum_{p=1}^{M} \psi_p \tag{3.19}$$

The photon mapping estimator is prone to different sources of errors (variance or bias). First, if the flat surface condition is not met (in corner) the estimator will underestimate the right value. Second, if not enough photons are gathered (in

the kernel), the incident flux estimate has a high variance. This variance will be analysed in the next sub-section.

In the rendering step, the photon map is used to estimate the reflected radiance. However, using the photon map directly can result in a very noisy rendering. *Final gathering* uses the photon map only to compute indirect lighting. This is done by integrating along all the incoming directions at the points resulting from the intersection between view rays and the scene. For each incident direction, a ray is traced and may intersect the 3D scene. At the intersection point, a density estimation is performed. Final gathering can produce high quality images. However, this approach is costly because it requires a lot of photon map queries.

This technique is not consistent. Indeed, for consistency, we need to be able to shoot an infinite number of photons, which requires an infinite memory to store the photons, which is not feasible. Progressive schemes have been developed to address this issue. We will present these approaches in the following sub-section.

3.3.2 Progressive photon mapping

To address the memory issue, *progressive photon mapping* (PPM) has been proposed by Hachisuka et al. [HOJ08]. The authors proposed an extended version of this algorithm: *stochastic progressive photon mapping* (SPPM) [HJ09]. In this last technique, several photon shooting steps are achieved to produce different photon maps. Then, for each pixel, a primary ray is traced, it intersects the scene at a certain position. This intersection is denoted *gather point*. For each gather point, the photon map is used to estimate the reflected radiance (eq. (3.18)).

Knaus and Zwicker [KZ11] have developed a cleaner framework than PPM/SPPM. So, for this reason, we will use their mathematical framework to explain how PPM/SPPM works. Then, we will underline the difference between their technique and the techniques proposed by Hachisuka et al.

In their article [KZ11], Knaus and Zwicker propose an expression of the photon mapper error $\epsilon(\vec{x}, r)$ for a given position $\vec{x}$ and a kernel radius r. For simplicity, they use the assumption that the probability density of the photons p_l is constant within the support of the kernel $\mathrm{k_r}$. Their aim is to study the expectation and the variance of the photon mapper error ($\epsilon(\vec{x}, r)$). After several developments, they come up with the expression of the variance of this error:

$$\mathrm{Var}[\epsilon(\vec{x}, r)] \approx \frac{\mathrm{Var}[\psi] + E[\psi]^2) p_l(x)}{M\, r^2} \int_{\mathcal{M}} \mathrm{k_r}(\vec{x} - \vec{y}) \mathrm{d}A(\vec{y}), \tag{3.20}$$

where $\mathrm{Var}[\psi]$ and $E[\psi]$ are the variance and the expectation of photon's contributions. This above formula shows that the variance of the error decreases linearly with the number of photons M. Moreover, they express the expectation of the photon mapper error as:

$$E[\epsilon(\vec{x}, r)] = r^2 E[\psi] \tau, \tag{3.21}$$

where τ is a constant related to the photon distribution. This last formula shows that the expectation of this error decreases proportionally to the square scale of the kernel.

In their technique, the different passes (photon shooting and rendering step) are independent. Each rendering step produces an image I_j obtained by averaging all the previous ones $(I_0, ..., I_{j-1})$. In this case, Knaus and Zwicker propose to study the cumulated error when the different rendering passes are averaged:

$$\bar{\epsilon_K} = \frac{1}{K} \sum_{j=1}^{K} \epsilon_j, \tag{3.22}$$

where ϵ_j is the error for the j^th rendering pass. They prove that the expectation and the variance of this accumulated error have the following expressions:

$$\mathrm{Var}[\bar{\epsilon_K}] = \frac{1}{K^2} \sum_{j=1}^{K} \mathrm{Var}[\epsilon_j] \tag{3.23}$$

$$E[\bar{\epsilon_K}] = \frac{1}{K} \sum_{j=1}^{K} E[\epsilon_j] \tag{3.24}$$

where $\mathrm{Var}[\epsilon_j]$ and $E[\epsilon_j]$ are given in eqs. (3.20) and (3.21). The aim of the authors is to find a mathematical formulation such that when $K \to \infty$, the cumulated error tends towards 0:

$$\mathrm{Var}[\bar{\epsilon_K}] \to 0 \qquad \text{and} \qquad E[\bar{\epsilon_K}] \to 0$$

If these two conditions are met, their estimator is consistent. This is achieved by choosing the following kernel radius reduction formula:

$$\frac{r_{j+1}^2}{r_j^2} = \frac{\mathrm{Var}[\epsilon_j]}{\mathrm{Var}[\epsilon_{j+1}]} = \frac{j+\alpha}{j+1} \tag{3.25}$$

where $\alpha \in]0, 1[$ is a user parameter and r_{j+1} is a radius used during the photon's gathering.

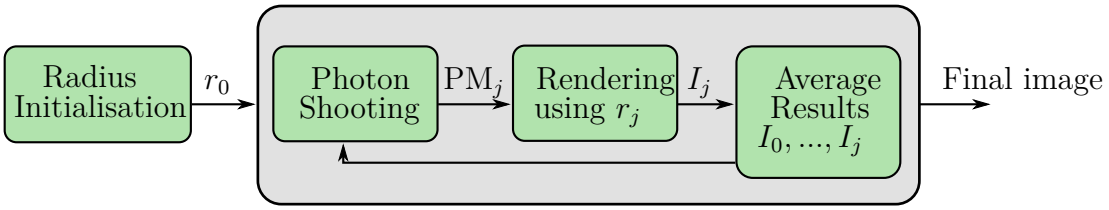


Figure 3.12 – The global framework of Knaus and Zwicker's technique. First, the initial kernel radius r_0 is computed. Then, at each pass of their algorithm, the algorithm produces a photon map PM_j and uses it to produce an image I_j. The images I_j are averaged to produce the final image.

In practice, their algorithm works as follows (fig. 3.12). First, an initial radius r_0 is computed (global radius, ray differential, ... etc.). At each pass, a photon map PM_j is created. This photon map PM_j is used to generate an image I_j using the kernel radius r_j (eq. (3.25)). Then, this image I_j together with all previous rendered images $I_0, ..., I_{j-1}$ are averaged to get the final result. The good algorithmic property of their approach is that each pass is independent from the others. This makes possible the parallelization of their algorithm on a cluster of computers.

The main difference between Knaus and Zwicker's and Hachisuka's works is the problem statement. Knaus and Zwicker make the hypothesis of constant probability to sample contributive paths for a given gather point. Therefore, they use the same rate reduction for all the radii (associated with the pixels). Hachisuka derives the same formula (eq. (3.25)) but makes the hypothesis of constant photon density within the radius. By doing that, he express the radius rate reduction depending on local pixel statistics:

$$\frac{r_{j+1}^2(P)}{r_j^2(P)} = \frac{N_j(P) + \alpha M_j(G_j)}{N_j(P) + M_j(G_j)}, \tag{3.26}$$

where P is a pixel and G_j the associated gather point, $M_j(G_j)$ the number of photons gathered at the gather point during the j^{th} iteration, and $N_j(P) = \sum_{l=1}^{j-1} M_j(G_j)$. It is possible to adapt the radius reduction rate for different pixels. This possibility entails an additional memory cost. Indeed, for each pixel, we need to store several statistics:

- the radius scale $r_j(P)$;
- the number of cumulated numbers of gathered photons $N_j(P)$;
- the unormalized flux $\psi_j(P)$ (in Hachisuka, the final image is not an average over all produced images but a weighting sum).

Other statistics can be assigned to a pixel. Kaplanyan et al. [KD13a] propose to store a derivative information about the photon density. With this additional information, they are able to choose the quasi optimal radius reduction rate adaptively. This makes the algorithm converge faster. Note that the information assigned to the pixels is updated using the information of the previous passes. This makes difficult the parallelization of the algorithm on a cluster of computers.

Finally, with these techniques, two different gathering photon strategies are possible. In the classical one, for each gather point, the photon map is queried to gather the neighbouring photons. In the second one, the gather points are stored in a spatial data structure. Then, during the photon shooting step, for each photon, this spatial data structure is traversed to find neighbouring gather points. For each found gather point, the contribution of the photon is computed and cumulated. These two solutions are valid and their performance relies on the ratio of the number of photons shot per pass to the number of generated gather points. However, the second solution has another advantage. Indeed, its makes it possible to know if a given light path is contributive or not during the shooting process. One of the works done in this thesis was to find an efficient data structure for gather points within participating media. This work will be presented in chapter 6.

3.4 Combining biased and unbiased estimators

In the two previous sections, we have presented two main rendering techniques: BDPT and SPPM. These two techniques are not expressed within the same mathematical framework. Indeed, BDPT is expressed within the path integral framework

while photon mapping within the density estimation framework. However, these two rendering techniques are complementary:

- SPPM is more robust than BDPT in scenes containing a lot of smooth materials. Moreover, the possibility to reuse several light paths during the density estimation makes SPPM more efficient.

- BDPT relies on multiple importance sampling (MIS) to build a good sampling strategy. This method is more efficient than SPPM to compute the direct and the diffuse interactions, because it allows to connect two distant points, which is not possible with SPPM (density estimation needs the two path vertices to be close to each other). Moreover, BDPT has a partial control on the sampling rate in the image space. So, except the lack of robustness for handling smooth materials, BDPT has a better convergence rate than SPPM.

Recent works [GKDS12, HPJ12] studied the possibility to combine BDPT and SPPM using MIS. The major issue solved in these works was the differencies in the mathematical framework used. Indeed, to use MIS, the two techniques need to be expressed in the same framework. For example, Georgiev et al. [GKDS12] chose to express the photon mapping framework in the path integral one. Some results from this paper are shown in fig. 3.13.

Figure 3.13 – On the left: a comparison between BDPT and SPPM. For the same computation time, these two techniques perform differently in the image space. BDPT performs better in all cases except for the caustics viewed through the mirror. On the right: the combined techniques achieved lower variance when using MIS. These images are courtesy from Georgiev's article [GKDS12]

In the same spirit, Krivanek et al. [KGH+14] use the same approach to combine different estimators in participating media rendering. However, they combine 4 different rendering techniques.

3.5 Discussion

Up to now, we have focused on the different path space sampling techniques: path tracing, light tracing, BDPT, photon mapping and SPPM. These techniques use different Monte Carlo estimators to achieve lower variance. However, there exists other ways to speedup the rendering algorithms.

For example, some works have been interested in real-time global illumination techniques. A recent state of the art [RDGK12] summarizes all the recent research aiming at real-time performance. One of these techniques is virtual point light (VPL) [Kel97] based method which is a GPU friendly version of a subset of BDPT. Indeed, VPL-based techniques use the GPU raster operation to evaluate the visibility in the connection operation. However, to get artefact-free images, a large amount of VPL is necessary. This problem of computing visibility over a large set of lights is also known as *many light source* problem. In order to achieve real-time performance, several solutions are possible. The first one is to reduce the cost of the visibility evaluation by using some approximations [RGK+08]. The second is to use a specific technique to cluster the VPLs: matrix row sampling [HPB07], light-cuts techniques [WFA+05, WABG06, WKB12].

Another category of techniques to speed up the rendering are *caching techniques*. These techniques are similar to photon mapping techniques but they store more information about the rendering. They allow to reuse information already stored into a set of records. Caching techniques store indirect lighting information into records. Similarly to photon mapping, these records are inserted into a spatial data structure. At each intersection point, between a view ray and the scene, the illumination is computed by interpolating the lighting stored in the neighbouring records using gradient information. A lot of improvements of this type of technique have been proposed: different BSDF handlings [WRC88, KGPB05], different record shapes [RCB11] or different interpolation schemes / gradient [WH92, JZJ08b, SJJ12].

Moreover, caching techniques can be used to cache other sampling information. One recent work [VKŠ+14] uses the caching technique to store an explicit knowledge of the incident radiance (or importance) distribution. Its aim at constructing a pdf that matches the shape of the integrand. This technique produces good results (low variance) but still has issues in corners due to the caching technique. Moreover, this technique is complex to implement, we will see in the next chapter a simpler technique to automatically construct the pdf implicitly.

Markov Chain Monte Carlo 4

1 Introduction

In the previous chapter, we have presented different global illumination techniques. These techniques often rely on local path sampling strategies and use importance sampling to reduce the variance. The aim of these different techniques is to get a density function the closest as possible to the integrand. However, building such a density function is challenging and complex.

Another way to sample the path space is to use Metropolis-based algorithms [MRR+53]. These algorithms have the ability to easily construct a density function proportional to an importance function I defined by the user. This importance function does not need to be normalized and can be any function different from zero especially when the integrand value is non-zero. Metropolis techniques are based on a Markov Chain process and generate a sequence of samples proportional to the importance function.

In computer graphics, Metropolis-based techniques are called "Metropolis Light Transport" techniques (MLT). We have dedicated a chapter to this technique because several contributions in the manuscript are based on MLT. So, in this chapter, we will give a quick overview of the MLT techniques and discuss their differences and their strengths.

2 Overview of the MLT algorithm

Veach et al. [VG97] were the first to apply Metropolis algorithm in the computer graphics field. The aim of MLT methods is to generate more contributive paths compared to classical Monte Carlo approaches in scenes with complex visibility. Each path $\overline{\mathbf{x}}$ (as defined in the previous chapter) or X is defined by a sequence of $\vec{x}_0, \vec{x}_1 \cdots \vec{x}_k$ points on the scene surfaces or inside a participating media (see chapter 2). The aim of the Metropolis algorithm is to generate a sequence of paths $X_0, X_1, \cdots X_N$ proportional to an importance function I[1]. Each path of this sequence is a state of the Markov chain. Those states are distributed proportionally to the importance function according to the pdf $p(X)$

$$p(X) = \frac{\mathrm{I}(X)}{b} \tag{4.1}$$

[1]Note that this is true only if the Markov chain reaches the stationary distribution (the mutation does not change the state distribution). However, for simplicity reason we will omit this detail for the rest of this chapter. For more information, the reader can refer to Veach PhD thesis [Vea97]

where X is a state of the Markov chain, b the normalization factor of the importance function and $p(X)$ the pdf proportional to the importance function. The normalization factor is not necessary during the Metropolis process. However, it is needed to express the probability density function of the samples generated by MLT. The expression of the normalization factor b is:

$$b = \int_{\mathbb{P}} \mathrm{I}(\overline{\mathbf{x}}) \mathrm{d}\mu(\overline{\mathbf{x}}) \tag{4.2}$$

where $\mathbb{P}$ is the path space, also called the Markov chain space (or domain).

Metropolis algorithm is based on a random walk. At each step of the random walk, the algorithm generates a proposed path Y by applying a random mutation to the path X_{i-1}, also called the current state. To be able to generate the sequence of paths $X_0, X_1, \cdots X_N$ with a distribution proportional to the importance function, each proposal has a chance to be rejected. If the proposal is accepted, the next state X_i will be equal to the proposal Y. If the proposal is rejected, the next state X_i will be equal to the current state X_{i-1}. Metropolis algorithm is Markov chain process because the next state X_i generation depends only on X_{i-1}.

Algorithm 1 Metropolis Light Transport

```
1:  X_0 ← InitialPath()
2:  for i = 1...N do
3:      Y ← Mutate(X_{i-1})
4:      a ← AcceptProb(X_{i-1}→Y)
5:      if a > rand(0,1) then
6:          X_i ← Y
7:      else
8:          X_i ← X_{i-1}
9:      end if
10:     RecordSample(X_i)
11: end for
```

Algorithm 1 summarizes the MLT algorithm. At the beginning of the algorithm (line 1, algorithm 1), we need to select the initial state X_0 of the random walk. This initial state needs to be chosen as proportional to the importance function I to avoid *start-up bias*[2]. However, the value of this importance function is unknown over the path domain. The solution, proposed by Veach et al. [VG97], is to generate a collection of random paths $X_{0,1}, \cdots X_{0,K}$ where K is the number of paths. For each path, we evaluate the importance function value $\mathrm{I}(X_{0,k})$ and pick one of them proportional to its importance value. The selected path is used to initialize the random walk. This technique is similar to the re-sampling procedure [TCE05] used in global illumination.

Once the initial state is selected, the random walk starts and, at each step, a proposed path Y is generated (line 3, algorithm 1). This proposed path Y is

[2]At the early stage of the random walk, the sequence of samples depends on the initial state choice. However, this bias vanishes after a certain amount of steps because the Markov chain "forgets" the initial state.

generated by applying a mutation to the previous state X_{i-1}. The probability of the mutation to generate the proposed Y is equal to $\mathrm{T}(X_{i-1} \to Y)$, where T defines the mutation. This mutation can be randomly selected among a collection of mutation operations. The most important things is that this set of mutations needs to be ergodic. This property means that the random walk requires a non-zero probability to visit a state with a non-zero importance function value. If this property is not respected, the rendering can be false and some contributions may be missing (due to a lack of exploration).

Once the proposal path Y is generated, we need to compute the probability to accept this proposal (line 4, algorithm 1). We use the formulation from Peskun [Pes73] giving the acceptance probability

$$a(X_{i-1} \to Y) = \frac{\mathrm{I}(Y) \cdot \mathrm{T}(X_{i-1} \to Y)}{\mathrm{I}(X_{i-1}) \cdot \mathrm{T}(Y \to X_{i-1})} \tag{4.3}$$

to accept the proposal Y when the current state is X_{i-1}.

After computing this probability, we use it to make a random decision to accept or not the proposal (line 5 to line 9, algorithm 1) as the new state X_i. Then, we add the contribution of the new state X_i to the pixels (line 10, algorithm 1).

Problem statement In classical Monte Carlo rendering algorithms, we use different estimators for each pixel j. By doing this, we can control the number of samples for each pixel and control the variance. However, to be efficient, MLT needs long random walk in the state domain. So, it is not possible to use different chains for each pixel.

To be able to render an entire image with only one chain, we change the path contribution f_j to a pixel j and decompose it in two terms:

$$f_j(\overline{\mathbf{x}}) = h_j(\overline{\mathbf{x}}) f(\overline{\mathbf{x}}) \tag{4.4}$$

where h_j is the filtering function attached to the pixel j and f is the path contribution. MLT techniques explore all the path space and we filter the contribution of the Markov chain using h_j. At the end, the final image can be computed with the following estimator:

$$I_j = \frac{1}{N} \sum_{i=0}^{N} \frac{h_j(X_i) f(X_i)}{p(X_i)} \tag{4.5}$$

where $X_0, X_1, \cdots X_N$ is the sequence of paths used for all the pixels. We can rewrite the previous formula using eq. (4.1) as:

$$I_j = \frac{b}{N} \sum_{i=0}^{N} \frac{h_j(X_i) f(X_i)}{\mathrm{I}(X_i)} \tag{4.6}$$

where $\mathrm{I}(X_i)$ is the importance function value for the state X_i and b is the normalization factor used to scale the final rendered image.

Discussion Note that the MLT sampling advantage (explore the domain locally) could introduce a disadvantage in term of convergence speed. Indeed, the samples (states) could be correlated. Independent samples in Monte Carlo guarantee that the standard deviation σ is equal to $\sigma_p/\sqrt{N}$ with N samples and σ_p is the standard deviation when considering the samples independent.

$$\sigma \leq \sigma_p \cdot \sqrt{\frac{1 + 2\sum_{k=1}^{N} R(k)}{N}} \tag{4.7}$$

where $R(k)$ is the upper bound of the correlation between $\mathrm{I}(X_i)$ and $\mathrm{I}(X_{i+1})$. Thus, a too high correlation during the paths generation increases the variance: in this case, the acceptance ratio is too high (too small mutations) or too small (risk to stay in the same state of the Markov Chain). The acceptance ratio is the ratio of mutated states that entail a change of state in the Markov Chain.

3 Practical aspect

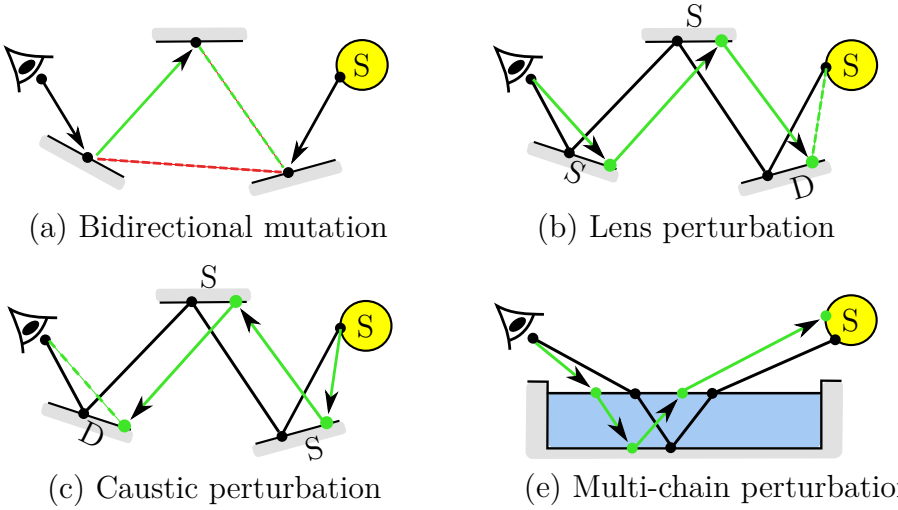
In the previous section, we have described the general metropolis algorithm. In computer graphics, several different Metropolis light transport methods (MLT) have been proposed. Each of these techniques introduces several changes on:

- State space or Markov chain domain: in computer graphics, several domains are possible. We will describe each of them and discuss their strengths and drawbacks.
- Mutations: several mutations are possible to generate a proposal. These mutations are related to the chosen state space. This is why we will present the state space and the mutations in the same subsection.
- Importance function: its choice is important to the rendering algorithm. Several choices are possible and will be discussed hereafter.
- Rendering technique: the choice of the rendering technique will have an impact on above points. We will give an overview of the different rendering techniques.

3.1 State domain and mutations

Path domain Veach et al. [VG97] propose to use the path space directly. This means that each state of the Markov chain will represent a path $\overline{\mathbf{x}}$ with vertices $\vec{x}_0, \vec{x}_1 \cdots \vec{x}_k$. Each vertex defines the path geometry (position, direction, BSDF, etc.). Mutation on path domain uses these sets of information to determine an efficient mutation strategy. Moreover, a mutation can only change a part of the path, which makes this technique computationally efficient.

However, all these strengths comes with a major drawback: the complexity. Indeed, Veach et al. proposed a MLT algorithm with a BDPT. Some light phenomenon are difficult to sample. Veach et al. proposed to use the following set of mutations (see fig. 4.1):



(a) Bidirectional mutation (b) Lens perturbation

(c) Caustic perturbation (e) Multi-chain perturbation

Figure 4.1 – All the possible mutations from Veach et al. [VG97] technique. Some of the mutations are specialized to sample specific light path structures. New parts generated by the mutations are shown in green. Characters "S" and "D" mean specular or diffuse surfaces.

- Bidirectional mutation: this mutation is responsible for large changes to the path, such as modifying its length. First, the subpath to delete is chosen with some probability. Then, a new subpath is generated randomly by adding vertices to the camera and the light paths. Note that the mutations have some probability to delete the path completely and regenerate a new one. This property is important because it makes the set of mutations ergodic.

- Lens and caustic perturbations: these mutations are more specific. Indeed, they will be responsible for the path regeneration from the camera (lens perturbation) or the light sources (caustic perturbation). This is done by regenerating only specular vertices by changing the original path direction.

- Multi-chain perturbation: This mutation can be seen as a generalization of the two above perturbations. This perturbation is designed to handle $(D|L)DS^+DS^+E$ paths.

Then, for each current state X_{i-1}, a mutation operation needs to be chosen. We select one of them randomly. However, if a mutation (lens, caustic or multi-chain perturbation) requirement over path structure is not met, this mutation has zero probability to be picked.

However, the previously described set of mutations is not sufficient when paths have several specular chains. Moreover, the path changes due to these perturbations (lens, caustic and multi-chain) cannot create a mutated path without changing one of the end points (on the light or on the camera). This can be an issue when, for example, a light source is highly directional (in case of sun light).

To address this issue, Jackob et al. [JM12] have proposed a more complex mutation called "Manifold exploration" (fig. 4.2). Their technique is capable of generating a new path with the same end points[3]. To do it, they take advantage of the half

[3]Note that more complex paths with several specular successive bounces are handled as well.

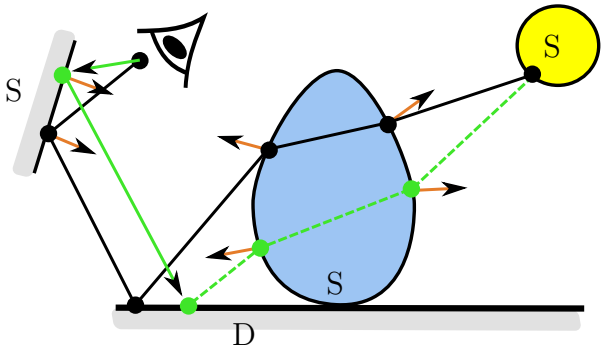


Figure 4.2 – "Manifold Exploration", the mutation proposed Jackob et al. [JM12]. The mutation takes advantage to the half vector constraint (orange vectors) due to specular reflection to create a sub-space and use it to explore the sub-space. The mutation works as follows: first a new sub-part of the path is generated (green plain line) by path tracing. Then, a Newton method generates the rest of the path (green dotted line) by using the half vector constraint. At the end, a new path is generated with the same end points, which is not the case of the multi-chain perturbation.

space constraint on the path vertex. Indeed, for specular interaction, the half vector needs to be aligned with the normal of the surface, which restricts the space of possibilities. By using this constraint, the authors express a sub-space (manifold) and use Newton method to explore this sub-space.

Half-vector domain Kaplanyan et al. [KHD14] have proposed an improvement of Manifold Exploration (ME) for glossy surfaces. Indeed, in ME, glossy surfaces can be also handled by using a special trick. The trick is to consider all glossy surfaces as "specular" by fixing the half vector value. However, to be able to explore the glossy lobe, one of this half vector is perturbed. The problem of this trick is that the mutation does not explore the path space well and creates correlated samples.

Kaplayan et al. want to solve this issue by creating a mutation strategy that perturbs all the half vectors at a time. To do this, the authors propose a new path space named "half-vector space". In this space, the path is expressed as two endpoints and halfway vectors.

Then, the authors use this new space for designing a new mutation. This mutation consists of two steps:

1. perturb half-vectors;

2. find a new path: this is done similarly to ME. However, the new space is not compatible with ray tracing. So, the technique transforms the path from the new space to the old one to trace it.

At the end, the path proposed by this technique undergoes more changes than Manifold exploration. Note that, if the path contains only specular interactions, this technique is similar to Manifold exploration. However, this new mutation strategy has additional strengths such as:

- Importance-sampling all BSDFs: this technique takes advantage of the fact that a lot of BSDF models are based on the half vector formulation. This makes

possible importance sampling of BSDF by making the half-vector perturbation proportional to the half vector distribution.

- Stratification: the technique also makes it possible to change one of the end point for stratification purpose.

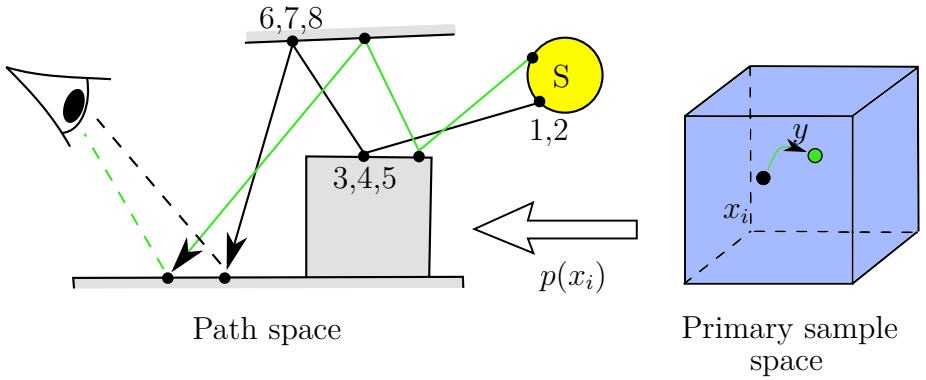


Figure 4.3 – The idea of Kelemen et al. [KSKAC02] is to use the mapping between random numbers (in primary sample space) and the path space. Indeed, a random number vector x_i (numbers $1, 2, .., 8$ in the figure represent the indices of the generated random numbers in the primary space) is used as a Markov state. Kelemen et al. propose a MLT algorithm that only perturbs the random number vector x_i to get y and uses this new vector to trace a new path (in green).

Primary sample domain In the previous MLT techniques, a geometric domain (path or half-vector) was used for the Markov chain. This domain gives full information about the path geometry and allows to design a specific mutation. However, this entails a complex mutation function. As a consequence, those methods are complex to implement and can be inefficient when all the path configurations (depending of geometry and BSDF settings) cannot be handled.

To address this issue, Kelemen et al. [KSKAC02] proposed a more simple MLT technique based on the primary sample domain. This technique, named "primary space samples MLT" (called also PSSMLT) uses random numbers as Markov chain state. Indeed, in Monte Carlo rendering, the light paths are generated using a stochastic process. This process generates random samples that are used to create a random path. So, there is a mapping between the random number space and the path space. This mapping and the designed mutations are used to perturb the random number values (fig. 4.3).

More precisely, in global illumination, we try to solve the following equation:

$$\Phi_j = \int_{\mathbb{P}} f_j(\overline{\mathbf{x}}) \mathrm{d}\mu(\overline{\mathbf{x}}), \tag{4.8}$$

where f_j is the contribution of the path $\overline{\mathbf{x}}$.

Where MLT uses the primary sample space, we need to change the integral space:

$$\Phi_j = \int_{\mathcal{U}} f_j(p^{-1}(u)) \cdot |\frac{\mathrm{d}p^{-1}(u)}{\mathrm{d}u}| \mathrm{d}u \tag{4.9}$$

where $\mathcal{U}$ is the primary sample space, u an random number vector and p^{-1} the sampling technique that takes the random number vector and produces the associated path. However, in this case, the Jacobian is equal to:

$$\left|\frac{\mathrm{d}P^{-1}(u)}{\mathrm{d}u}\right| = \frac{1}{p(u)} \tag{4.10}$$

where $p(u)$ is the pdf to construct the path $\overline{\mathbf{x}}$.

Working on the primary sample space makes the design of mutations simple. In this case, two mutations are proposed:

- "small step" mutation: we make a small perturbation of the random number value. The aim of this mutation is to create a path that is slightly modified so as to explore the domain locally.
- "large step" mutation: the problem of the previous mutation is that we are not sure to have ergodicity (due to the limited range). Moreover, in global illumination, it is very likely that light-paths of non-zero importance form island in the path space. So, the authors proposed another mutation, called "large step" mutation, to ensure ergodicity. This mutation amounts to create new random values from a uniform distribution. This allows the algorithm not to be trap inside peaks of the importance function.

A probability p_{large} is used to choose one of the mutations. This probability tells us about the chance to pick the "large step" mutation or not.

Note that PSSMLT is plug-able with any path construction techniques (path tracing, BDPT, photon mapping, etc.). However, in complex rendering techniques, such BDPT, several strategies are possible to build a path. However, by just playing with random number values, the algorithm does not have control on the choice of how to build a complete path. Hachisuka et al. [HKD14] solve this issue by proposing a MLT algorithm close to PSSMLT. In their technique, a state is defined by its random number vector and a number that depending on a path building strategy. When a mutation is invoked, random numbers are perturbed as well as the path strategy number.

Discussion We have presented different sampling domains for MLT. Apart from the half-vector space used only for designing a mutation operation, there are two main spaces:

- Primary sample space: by only mutating random numbers, PSSMLT is easy to plug into different integration techniques (path tracing, BDPT, SPPM... etc.). Indeed, this method can be used as a black box that produces random numbers. This property makes this MLT process quite robust. However, this technique cannot generate precise knowledge about the current state of the random walk and cannot apply special mutation. Moreover, in this technique, we need to rebuild the path from scratch to evaluate the importance function. Finally, PSSMLT will share the same limitation as the integration technique used. For example, if the integration technique cannot handle some path configuration

(like path tracing techniques with SDS interactions), PSSMLT will not be able to explore it efficiently.

- Path domain: in this domain, the MLT process has access to a lot of information, which makes an efficient mutation scheme possible. Moreover, mutations require to recompute only a part of the mutated path, which makes this technique very efficient. However, the problem of this domain is the variety of the situations that are difficult to handle well with a set of mutations. This is why this technique is less robust than PSSMLT (in case of robust integration technique, like photon mapping).

Note that in the path domain, we did not cover all the mutations. Indeed, different authors have proposed new mutations depending on the integration technique. For example, Fan et al. [FCL+05] proposed a special MLT algorithm in case of Photon mapping with final gathering. Another example is the work of Segovia et al. [SIP07] who designed a special MLT algorithm for VPL rendering technique. They used another Metropolis algorithm named Multi-try metropolis [LLW00].

Moreover, in this section, we did not cover all the possible Markov chain domains used in computer graphics. For example, we did not talk about the work of Cline et al. [CTE05] that consists in expressing the MLT algorithm as an energy flow problem. By doing so, the authors make it possible to use smaller Markov chains and relax the ergodicity requirement. Finally, Lehtinen et al. [LKL+13] have proposed a rendering system that computes only the image gradient. Then, they use the gradient information to reconstruct the 2D image.

3.2 Importance functions

Importance functions are used by MLT to determine which domain is more important to sample. It is a scalar function but its expression may vary from a method to another. Indeed, the importance function does not need to be normalized. The only requirement is that this function needs to be evaluated in all parts of the Markov Chain domain and be non zero when the path contribution is non zero. Moreover, the expression of the importance function depends on the rendering technique (path tracing, photon mapping, etc.).

Path based rendering techniques In the case of global illumination, Veach et al. [VG97] and Kelemen et al. [KSKAC02] use an importance function $I(X)$ based on the radiance reaching the image plane. Some problems occur when using this importance function. Indeed, the Metropolis-Hasting algorithm distributes samples proportionally to the radiance distribution, that is to say low radiance zones receive few samples. To solve this issue, Veach [Vea97] first proposes to only focus on the computation of indirect illumination (the direct component is computed with a classical Monte Carlo method). Second, the expected values of the acceptance probability is used to account for the contribution of the rejected state of the pixels. This technique is also called "waste recycle" in the mathematical literature.

Veach et al. [VG97] also propose a two-step Metropolis sampling. In a first step, the algorithm computes the radiance reaching the image plane with a low resolution

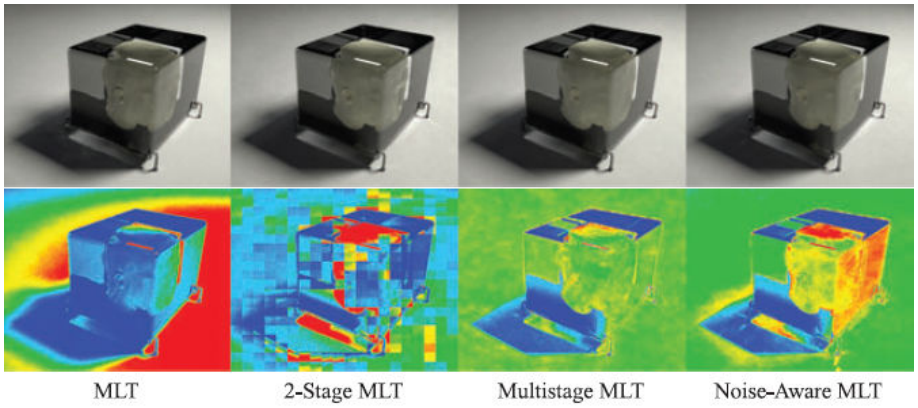


Figure 4.4 – Different results for the same MLT process using different importance functions. On the top row: the rendered image. On the bottom row: the sampling rate on the image plane. This figure stems from Hoberock et al. article [HH10].

image. This estimation is then used as an inverse importance function, as:

$$\mathrm{I}(X) = \frac{f(X)}{f_0(X)} \tag{4.11}$$

where $f_0(X)$ is radiance due to state X for the low resolution and computed at the first step. $f(X)$ is the contribution of state X for full resolution. However, this solution is very parameter sensitive and the first estimation could be not precise and create a bad importance function (which will create worst images compare to those only using the luminance as importance function).

Hoberock et al. [HH10] generalize this approach with a multistep algorithm by increasing the rendering resolution iteratively. Their technique does not require any user parameters and produce good results.

Moreover, in the same paper, Hoberock et al. [HH10] propose a perceptual importance function. This importance function uses the threadhold versus intensity rules to determine the proportion of samples needed to achieve less noisy images. To be able to do that, they design an iterative process that estimates the Markov Chain variance over an image region. A rendering example using all these importance functions is shown in fig. 4.4.

Photon mapping rendering techniques In the case of *progressive photon mapping*, Hachisuka et al. [HJ11] define the importance function as the visibility function of photons (1 if the photon is visible else 0). A binary function allows simplifications in the Metropolis-Hasting parameters. Moreover, the photons coming from the uniform distribution are already distributed proportionally to the importance function. This avoid the problem of start up bias.

Note that, in photon mapping, the photons are naturally distributed proportionally to light flux. So, this importance function only distributes the photons proportionally to their flux only on visible surfaces. The extension to participating media is discussed in chapter 6.

Chen et al. [CWY11] propose another importance function to get the same density of photons over visible surfaces. This importance function is based on the

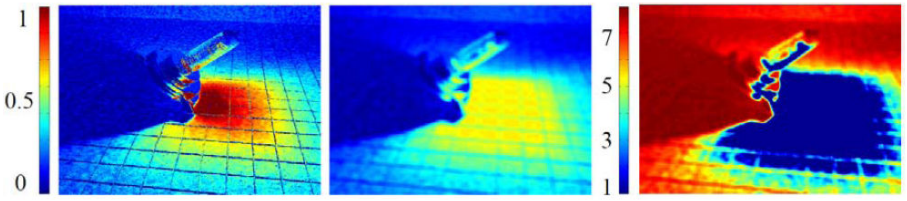


Figure 4.5 – From left to right: the initial photon density, the filtered initial density and the inverse of the second one. The rightmost image shows the importance used. Image steming from Chen et al. article [CWY11]

inverse density of photons. This density is estimated during some precomputation steps before the Metropolois sampling. In this precomputation steps, several filtering operations are needed to remove spikes in the importance function (fig. 4.5). In chapter 7, we will introduce more general importance function formulas trying to equalize the relative error.

Discussion Compared to classical Monte carlo rendering techniques, MLT techniques do not ensure stratification over the image space. However, the formulation of the importance function controls the amount of samples received on the image plane. In general, importance functions are defined in the image space, except for photon mapping, for which importance functions are expressed in the 3D space (over the gather point).

3.3 Other mathematical tools

In the previous sections, we discussed different aspects related to MLT such as: the Markov chain domain, mutations and importance functions. However, other mathematical concepts have been applied in MLT rendering algorithm. In this section, we provide details of some of these concepts.

Automatically tuning parameters The problem of MLT is the number of parameters. Indeed, this number can be high because each mutation has its own parameters. Those values are crucial for rendering efficiency. Indeed, some of these parameters have a strong influence on the acceptance rate (which is related to the sample correlation).

Each Markov Chain has to explore the state space through mutations. However, the mutation size is crucial for the exploration efficiency. Indeed, a small mutation size induces a high correlation ratio and, conversely, a big mutation size creates rejections in the acceptance function. A mathematical solution called *Adaptive MCMC* [AT08] allows to control this mutation size by introducing a new Markov Chain

$$\theta_{t+1} = \theta_t + H(t, \theta_i, X_t, ..., X_1) \tag{4.12}$$

where θ_{t+1} is the new mutation size and H the variation function of the mutation size. For the algorithm to be convergent, the variation function must converge to 0:

$$lim_{t\to\infty} H(t, \theta_t, ...) = 0 \tag{4.13}$$

Hachisuka et al. [HJ11] exploit this function to converge to an optimal acceptance ratio $A^* = 23.4\%$ [R+11]. Indeed, the mutation size is directly related to the acceptance ratio. Controlling this acceptance ratio allows to control the mutation size.

Zsolnai et al. [ZSK13] have proposed an automatic method to compute p_{large} for PSSMLT. In their work, they derived an adaptive formula based on large step mutation performances.

Genetic algorithms can also be used to tune the mutation size. Lai et al. [LFCD07] have proposed a rendering process based on this idea. Their algorithms can automatically select a good mutation size in a set of possible values.

Annealing and Replica exchange In case of spiky importance functions, a Markov chain has some difficulties to explore the state domain. It is the case of path tracing techniques, where specular or glossy interactions may trap the Markov chain. To solve this issue, the constraint on the importance function is relaxed, which makes the importance function flatter. However, to converge to the right solution, the function needs to converge to its "true" importance function. In the mathematics literature, this process is referred to as "Annealing" procedure. This technique has been used by Kaplanyan et al. [KD13b] to relax the directional restriction from specular BSDFs or point light sources.

Another approach is to use several importance functions. Each importance function will be ordered by levels: the first level will be the flattest function and the highest level will be the right function. With each of these importance functions, a Markov Chain is associated. However, it is possible to mix the samples from all these functions by using a *Replica Exchange* procedure. *Replica Exchange* procedure exchanges the current state of two different importance functions. In order to obey the Metropolis-Hasting rules, the probability to change these two states is equal to:

$$r(X, Z) = \frac{\mathrm{I}_X(Z)\mathrm{I}_Z(X)}{\mathrm{I}_X(X)\mathrm{I}_Z(Z)} \tag{4.14}$$

where $r(X, Z)$ is the probability to swap the two current states [4]. This approach has been used by Kitaoka et al. [KKK09] in computer graphics. In their work, the authors use 4 different levels.

Hachisuka et al. [HJ11] use the replica exchange for a visbility importance function. They use two importance functions: one constant over the domain and another one on the photon visibility. The first importance function is responsible for uniform sampling. This chain is equivalent to "large step" mutations in PSSMLT [KSKAC02]. But, this approach is more elegant when it comes to use uniform samples to estimate the normalization factor.

Connection to MIS Note that it is possible to combine different estimates using MIS. For example, it is possible to combine the contribution of different Markov chains (associated with different importance functions, different state domains).

[4] Note that more close the ratio is to 1, more there is state exchange. This is the reason why this exchange is only produced over two close levels

This is the case in the work of Kitaoka et al's [KKK09] where the 4 different level contributions are mixed using MIS. Another application of MIS in MLT is in Kelemen et al. [KSKAC02] work. Indeed, they combine MLT contributions with uniform samples generated by the "large step" mutation.

Part II

Efficient and robust rendering techniques

Introduction

In this part, our new rendering techniques are detailed. First, we will present rendering techniques that support participating media. Developing efficient rendering techniques, which compute multiple light interactions within such media, is a difficult task. Several methods presented in the previous chapters (path tracing, bidirectional path tracing, photon mapping, etc.) handle naturally the participating media. However, these rendering techniques do not take advantage of the higher integration dimension brought by the computation entailed when rendering participating media. Moreover, there are several ways to improve the efficiency of a rendering algorithm: using dedicated materials (GPU) or more advanced mathematical techniques/models. In chapter 5, we will present our GPU algorithm based on Fattal's [Fat09] CPU method. Indeed, his technique has been designed to run only on CPU. We have brought several modifications to this method to make it massively multithreaded (for running on GPU). Moreover, to overcome the memory constraint due to GPU limitation, we propose a streaming algorithm that streams the participating media during the computation.

However, this technique makes several assumptions: small number of lighting directions and the participating media do not contain any objects of the scene. These assumptions make this technique limited in practice. In chapter 6, to overcome these limitations, we will present a second technique, which is more general and runs on CPU, but may handle any type of interactions within the 3D scene. We propose a new data structure to store camera paths that interact with the media. This new data structure is organized similarly to a KD-tree structure. This makes possible to efficiently query the visibility of a photon or a light beam. Moreover, we use this visibility information to drive the path sampling thanks to a Metropolis-based algorithm which allows to render complex scenes in terms of visibility and light/matter interactions.

The human visual system is sensitive to relative differences in luminance but light transport simulation algorithms, based on Metropolis sampling, produce results with a non-uniform relative error distribution. To solve this problem, we propose an importance function, for Metropolis photon tracing, that ensures a good stratification sampling of photons, leading to pixel radiance estimated with equal relative error (chapter 7). We propose a hierarchical scheme for a progressive construction of the importance function from paths sampled during the rendering. Unlike previous works that defined an importance function in the image plane, our method operates in the 3D space (defined on spatial regions). This allows to take advantage of illumination coherence to compute robust estimates of the importance function while adapting to geometry discontinuities. We apply our photon tracing algorithm to progressive photon mapping and show that it considerably outperforms alternative

approaches in terms of image quality.

Light propagation maps on GPU 5

Volume interactions in participating media (presented in chapter 2, section 3, page 26) are computationally intensive. This is due to all possible light interactions (scattering, emission and absorption) that occur at every point inside the medium. These interactions are modeled by the RTE (Radiative Transfer Equation, eq. (2.22)) [Sub60] and different Monte Carlo methods are used to solve this equation. Some of them are specially designed to cope with the high integration dimension (when compared to surface rendering) [JZJ08a, JNT+11, NNDJ12a]. However, in case of a walkthrough, these techniques are view-dependent and need to perform from scratch new computations for each frame.

Irradiance/radiance caching could be an efficient rendering technique well suited for walkthrough. Indeed, if the lighting is constant, the created records can be used for all the frame during the walkthrough. However, with this kind of technique the creation of the cache cannot be easily parallelized.

Discrete Ordinate Methods (DOM) [SH01] rely on the discretization of the 3D and direction spaces to handle the different light interactions with participating media. Using this discretization, DOM techniques solve the radiative transfer equation iteratively by simulating local interactions. These methods are computationally expensive. However with this discretization, this type of method is well suited for an implementation on GPU. One of these methods proposed by Fattal [Fat09] resorts to Light Propagation Maps (LPM) to compute efficiently local interactions. It reduces the artifacts inherent in the DOM-based techniques. However, because of the various constraints imposed by a GPU (for e.g.: parallel execution, branching condition, etc.), the LPM approach as proposed by Fattal cannot be directly ported to GPU.

In this chapter, we will present a GPU implementation of Fattal's technique. Using a novel data organization, we transform the LPM approach to make it amenable to GPU implementation. Moreover, with a novel streaming mechanism, we make the resulting algorithm scalable and hence capable of processing volumes of any size on a GPU regradless of its memory capability. Our method is fast and scalable. We report more than 20× speed improvement by using our method as compared to Fattal's original method. Using our approach we are able to render $64 \times 64 \times 64$ dynamic volumes with multiple scattering of light at interactive speed for complex lighting. We are also able to render participating media of any size regardless of the memory capacity of the GPU.

1 Previous works

There are several categories of numerical techniques to solve the RTE equation (Monte Carlo, DOM techniques, etc.). We have rapidly presented some of them in

the previous chapters. For a better overview, the reader can refer to [CPCP+05, PPS97] for a survey on participating media rendering techniques. Two main approaches, handling multiple scattering, are either deterministic or stochastic (Monte Carlo). In this chapter, we focus only on deterministic methods and particularly on the Discrete Ordinate Methods (DOM) [SH01].

Discrete Ordinate Methods These methods rely on the discretization of 3D and direction spaces. They solve the RTE iteratively through local interactions. However, the DOM techniques suffer from artifacts named false scattering and ray effect (fig. 5.1). Several techniques have been proposed to reduce these shortcomings. For example, the technique proposed by Languenou et al. [LBC94] uses ray casting to solve the boundary conditions and to compute single scattering. Then, they use local interactions between voxels to compute the multiple scattering component. Fattal proposed another approach [Fat09], based on a fine sampling of the light propagation directions and a coarse sampling of the radiance stored at each voxel. We will explain that method in more detail in section 2.

GPU-based algorithms Zhou et al. [ZRL+08] developed a technique achieving real-time animated smoke rendering including multiple scattering. Their approach is based on the decomposition of the input smoke animation into a sequence of points with a radial basis function and a residual field. They use low ordered spherical harmonics to store the lighting information. They handle real-time manipulation of viewpoint, smoke attribute and lighting. But they cannot achieve fast smoke simulation due to the high preprocessing time needed to build their data representation.

Multiple scattering can also be approximated by diffusion equation [Ish78] which consists of a 2 coefficient spherical harmonic expansion of the radiance field. This method was introduced in computer graphics by Stam [Sta95]. Bernabei et. al [BHPB+12] implemented a parallel lattice-boltzmann [GRWS04] solution of the diffusion equation for rendering heterogeneous refractive media. Szirmay-Kalos et. al [SKLU+09] accelerated the iterative solution to the diffusion equation by making an initial guess based on a homogeneous medium assumption. This method has been further extended in [SKLU+11]. Wang et. al [WWH+10] also implemented a parallel solution to diffusion equation however they used a tetrahedral mesh instead of cubic grids for representing the volume, allowing to render arbitrarily shaped objects.

Englhardt et al. [ENSD12] presented a stochastic method based on instant radiosity. The authors use VPL (Virtual Point Light) to compute single scattering and multiple scattering as well. However, to avoid the singularity in the geometry factor, they need to clamp the VPL contribution. The problem of this solution is that some energy gets lost. To address this issue, the authors proposed to apply an approximate compensation bias step to correct the clamping and to get good looking results. Their technique is fast and can allow a surface to contribute to the radiance of the participating medium.

2 Fattal's algorithm

We recall here, the *radiative transport equation* (RTE) which models all the light interactions with a participating media (section 3, page 27). Light can be absorbed and/or scattered at every point in the volume:

$$\frac{\mathrm{d}L(\vec{y}\rightarrow\vec{\omega})}{\mathrm{d}\vec{y}} = \sigma_a(\vec{y})L_e(\vec{y}\rightarrow\vec{\omega}) - (\sigma_a(\vec{y}) + \sigma_s(\vec{y}))L(\vec{y}\rightarrow\vec{\omega}) + \sigma_s(\vec{y})L_i(\vec{y}\rightarrow\vec{\omega}) \tag{5.1}$$

where $L(\vec{y}\rightarrow\vec{\omega})$ is the radiance ($W \cdot m^{-2} \cdot sr^{-1}$) leaving a point $\vec{y}$ in direction $\vec{\omega}$. $L_e(\vec{y}\rightarrow\vec{\omega})$ is the self-emitted radiance, and is zero for non self-emitting media. $\sigma_a(\vec{y})$ and $\sigma_s(\vec{y})$ are the absorption and scattering coefficients that characterize the volume. The right most term $L_i(\vec{y}\rightarrow\vec{\omega})$ corresponds to multiple scattering where all the incoming directions are scattered in direction $\vec{\omega}$. This term is used to integrate all the incoming radiance from direction $\vec{\omega}_i$, times the phase function $\rho(\vec{y}, \vec{\omega}_i\rightarrow\vec{\omega})$[1]:

$$L_i(\vec{y}\rightarrow\vec{\omega}) = \int_\Omega \rho(\vec{y}, \vec{\omega}_i\rightarrow\vec{\omega})L(\vec{y}\leftarrow\vec{\omega}_i)\mathrm{d}\sigma(\vec{\omega}_i) \tag{5.2}$$

In [Fat09], Fattal proposed Light Propagation Map as a solution to the RTE equation. This method falls into the category of Discrete Ordinates Methods (DOM). This type of techniques relies on the discretization of two domains:

1. Spatial domain D: it characterizes the volume. The discretization splits the domain into a set of voxels. Each voxel has spatial indices x, y, z and it is denoted C_{xyz}. This set of voxels partitions of the space as $\bigcup C_{xyz} = D$. Moreover, the scattering and absorption coefficients are constant in the voxel and will be denoted $\sigma_s(C_{xyz})$ and $\sigma_a(C_{xyz})$ respectively.

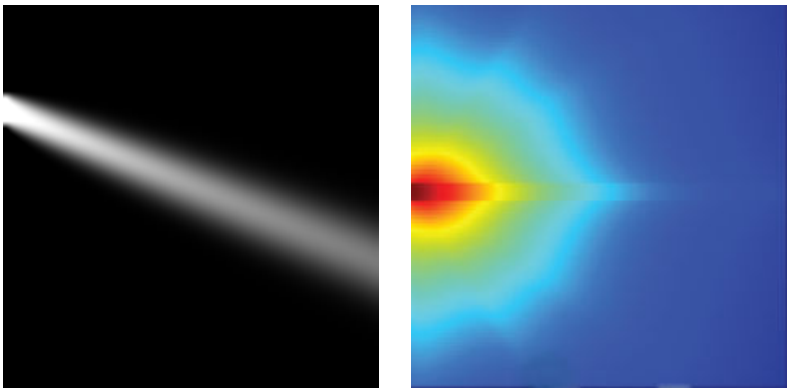
2. Direction domain Ω: the unit sphere Ω around the center of the voxel is sudivided into a set of directions Ω^d. The union of the solid angles around the sampled directions is equal to Ω: $\bigcup|\Omega^d| = \Omega$

The goal of this DOM technique is to approximate the radiance $L(\vec{y}\rightarrow\vec{\omega})$ (Eq. 5.3) by an average scattered radiance L^d_{xyz} in each voxel C_{xyz} and for the set of directions Ω^d:

$$L^d_{xyz} \approx \frac{\sigma_s(C_{xyz})}{V^d_{xyz}} \int_{C_{xyz}} \int_{\Omega^d} \left(\int_{S^2} L(\vec{y}\leftarrow\vec{\omega}_i)\rho(\vec{y}, \vec{\omega}_i\rightarrow\vec{\omega})\mathrm{d}\sigma(\vec{\omega}_i) \right) \mathrm{d}\sigma(\vec{\omega})\mathrm{d}\vec{y} \tag{5.3}$$

where L^d_{xyz} is the radiance in the voxel xyz along the direction Ω^d. $V^d_{xyz} = \Delta_V|\Omega^d|$ is the product of the volume of the voxel C_{xyz} and $|\Omega^d|$ the solid angle. In this representation, the emission, absorption and scattering coefficients are assumed to be constant in each voxel.

[1] In the rest of the chapter, the normalization of 4π is integrated to the phase function. This is not the case in the original Fattal's paper where the author performs an explicit division by this normalization factor.



(a) False scattering **(b)** Ray effect

Figure 5.1 – Because of the discretization, DOM techniques can be source of two artifacts: false scattering and ray effect. (a) False scattering makes impossible to maintain a sharp profile of a light beam. (b) Ray effect occurs when one cansee the discretization of directions. These problems are reduced in the Fattal's approach.

The main problem with the DOM techniques is that they suffer from two main shortcomings, namely *false scattering* and *ray effect* (fig. 5.1). The common solution to reduce these two artifacts is to increase the space and direction discretization resolutions. However, this approach requires huge amount of memory and hence limits the method practically.

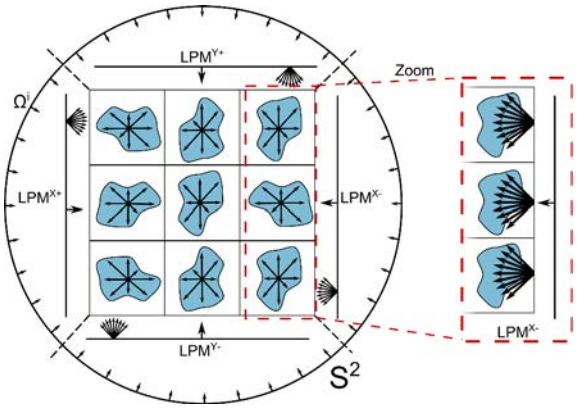


Figure 5.2 – The unit sphere Ω is discretized into a set of m or n directions Ω^i. For each voxel, Fattal computes the scattered radiance for m directions with $m << n$, n being the number of propagation directions. An LPM is an array of point elements, each emitting light in n directions sampling the solid angle $\frac{4\pi}{6}$ of the sphere Ω.

To overcome this problem, Fattal proposes a two-level sampling: a fine sampling of 3D space and direction for the light propagation and a coarse sampling for storage. To reduce the memory capacity during the propagation step, Fattal uses an iterative approach by subdividing the direction domain. To this end, he uses 2D ray maps called *light propagation maps* (LPM) which have a high spatial resolution and represent a subset of the finely sampled propagation directions (fig. 5.2). At each iteration, 6 LPM are needed to propagate the light for each face of the medium. Starting with a face of the coarse grid, this 2D map propagates step by step in one voxel plane at a time and the propagation results are stored at the center of the

voxels. So, the LPM is basically a high resolution 2D array, each of its element (r, s) stores a set of rays originating at (r, s) and having directions sampled from the unit sphere Ω.

The propagation process must compute the interaction between the currently propagated LPM and the volume. For the first propagation, the energy carried by each ray is initialized with the boundary conditions. The boundary conditions correspond to the incoming radiance arriving at the volume boundary. Then the rays are propagated along their directions using ray marching. When a ray leaves the medium at the boundary point A (before reaching the opposite face), then a new ray of the same direction with zero radiance is generated from an opposite point B as shown in fig. 5.3. The aim is to have the same number of rays in each part of the volume.

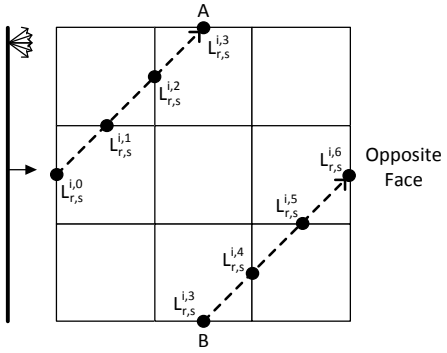


Figure 5.3 – $L_{r,s}^{i,t}$ is the ray radiance at the (r,s) element of the LPM, i the ray direction defined by Ω^i and t the number of the voxel traversed by the ray. Given a direction, the ray can leave the medium at a point A. In this case, we need to create new ray at the point B to cover the whole domain.

Now let us give more details on the Fattal's approach. Let $L_{r,s}^{i,t}$ be the radiance of a ray originating at a point (r,s) of the LPM in direction Ω^i at the p^{th} propagation step from all the faces back and forth, t being the intersection number of the ray with the faces of the current traversed voxel. Let the voxel (x,y,z) be the t^{th} voxel encountered by the ray. The radiance $L_{r,s}^{i,t}$ leaving the t^{th} voxel is given by

$$L_{r,s}^{i,t} = \underbrace{L_{r,s}^{i,t-1} e^{-\triangle l_q \cdot \sigma_t(C_{xyz})/\omega_z^n}}_{\text{energy lost due to extinction}} + \underbrace{U_{xyz}^{d,p} \times \frac{(1 - e^{-\triangle l_q \cdot \sigma_t(C_{xyz})/\omega_z^n})}{\sigma_t(C_{xyz})}}_{\text{energy gained due to scattering}} \tag{5.4}$$

where $U_{xyz}^{d,p}$ is the stored unscattered radiance for direction d (equal to the closest propagation direction i), p is the p^{th} propagation (scattering order), $\triangle l_q$ the distance traversed by the ray in the voxel and $\sigma_t(C_{xyz})$ the extinction coefficient in the traversed voxel.

Given a propagation direction, when a ray adds its radiance contribution to the currently traversed voxel, this contribution must be scattered into all directions. Fattal stores the current scattering contribution in a variable named U that is used in future propagation steps. As we mentioned earlier, Fattal chooses different sampling frequencies for I and U but for the reason of clarity, in the rest of this chapter we

will assume that they are the same. U and I are updated as:

$$\begin{aligned} R &= (V^d_{xyz})^{-1} A_{r,s} F^{i,d} L^{i,t}_{r,s} (1 - e^{-\triangle l_q \cdot \sigma_s(C_{xyz}))/\omega^n_z}) \\ L^d_{xyz} &= R \quad (Eq.5.3) \\ U^d_{xyz} &= R, \end{aligned} \tag{5.5}$$

where $A_{r,s}$ is the area of the (r,s) LPM sample. $F^{i,d}$ is phase function precomputed value equal to:

$$F^{i,d} = \int_{\Omega^i} \int_{\Omega^d} \rho(\omega, \omega') d\omega d\omega' \tag{5.6}$$

where superscripts i and d are for the propagation direction and storage direction respectively, "d" being the direction closest to the direction represented by "i".

Recall that one iteration represents the light propagation from each element (r, s) of the LPMs. It corresponds to one scattering while multiple iterations correspond to multiple scattering. Fattal proposes to terminate the iteration process when the unscattered light is low in every voxel. I and U are initialized with self-emission radiance (if any) of each voxel. Moreover in pratice, we keep only two U buffers, the $(p-1)^{th}$ and the p^{th} U buffers and use a swap mechanism to reuse them.

Fattal's method is summarized by Algorithm 1.

Algorithm 2 Fattal's original algorithm

```
// p is iteration number (scattering order)
Initialize p_prev to 0, the index of the previous iteration
Initialize U^{d,p_prev}_xyz and I^d_xyz with medium emitted light
while max_xyz |U^{d,p_prev}_xyz| < ε && p>0 do
    // Initialisation U buffer of the p_cur^th iteration
    p_cur = (p_prev + 1)%2 // Current iteration index
    Initialize U^{d,p_cur}_xyz to 0
    for each LPM do
        for each propagation direction i do
            for each element (r,s) of the current map do
                if p == 0 then
                    Initialize L^{i,0}_{r,s} with the boundary conditions
                else
                    Set L^{i,0}_{r,s} to 0
                end if
                for each intersected voxel t do
                    Update the ray radiance L^{i,t}_{r,s} from U^{d,p_prev}_xyz (Eq. 5.4)
                    Update U^{d,p_cur}_xyz and I^d_xyz (Eq. 5.5)
                end for
            end for
        end for
    end for
    p_prev = p_cur
end while
```

3 New Method: Parallel and Scalable LPM

3.1 Parallelization

A simple strategy to parallelize Fattal's algorithm would be to assign a computation thread to each ray of the LPM. However, multiple rays affect the values stored in a voxel (see Fig. 5.4). So, this creates a synchronization problem. It may be possible to create synchronization barriers on the writable information and continue with the original simple approach. However, the algorithm will be less efficient because the latency, needed for the synchronization, will be significant as compared to the computation time.

In CUDA[2], updating the I and U voxel values could be performed using atomic operations on data of floating point type. However, using atomic operations has a significant cost because of the branching condition and the multiple global memory transactions (rather than cache accesses as in our approach thanks to data locality), as shown in Fig. 5.9.

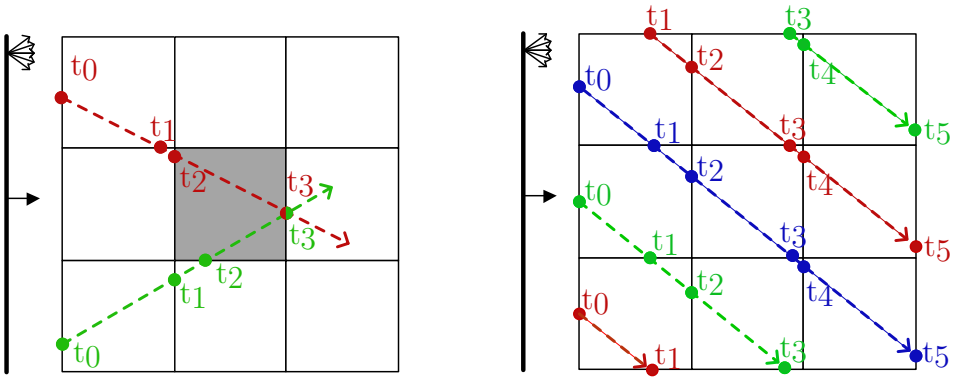


Figure 5.4 – On the left, we have the original Fattal's algorithm where we put one thread per ray (represented with different colors). The problem is that it produces synchronization problem for writing into the grey voxel. On the right, we only propagate rays having the same direction, thus guarantee only one ray write per voxel.

We address this synchronization related problem by dividing the original propagation step into two steps (illustrated in fig. 5.5):

1. Propagation step: we group together all the rays of same propagation direction. Thus we guarantee that each voxel is traversed by only one ray at a time. So we create a temporary buffer to store the radiance brought by a ray when it goes through a voxel.

2. Collecting step: we use all the temporary buffers to update the $U_{xyz}^{d,p}$ and I_{xyz}^{d} values. In order to simplify the discussion, we assume that the LPM spatial resolution is the same as the volume face spatial resolution.

[2] CUDA version used to develop this project was 4.X. So performance assumptions/comments on atomic operations and streaming process are based on this CUDA version.

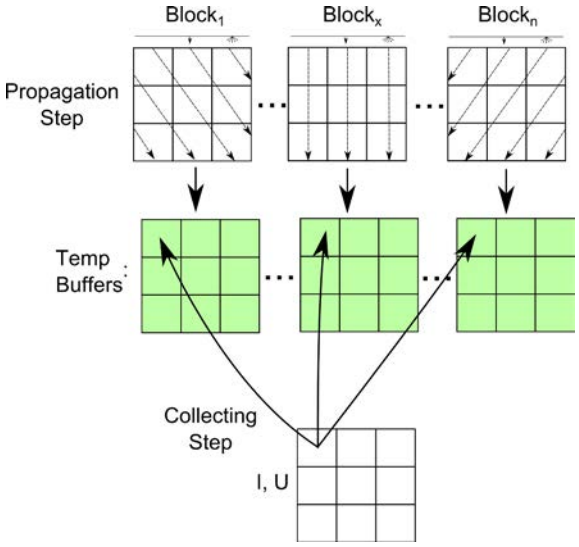


Figure 5.5 – Summary of the GPU algorithm. Each Block in the propagation step manages one temporary buffer. Then, in the collecting step, each voxel reads temporary buffers to update I and U.

We create two GPU kernels: one for the propagation step and the other for the collecting step. One propagation step corresponds to one LPM propagation and requires propagation of multiple blocks. Each block corresponds to all the elements of the LPM and to only one propagation direction. There are as many blocks as directions. We assign a thread to each element (r,s) of the LPM. In this way, there is no execution divergence between threads in the same block. The role of a thread is to trace a ray with origin (r,s) and direction i and to compute its contribution to the traversed voxels. This contribution is stored into a temporary buffer. The ray contribution can be expressed as:

$$T^i_{xyz} += L^{i,t}_{r,s}(1 - e^{-\triangle l_q \cdot \sigma_s(C_{xyz})/\omega^n_z}), \tag{5.7}$$

where T^i_{xyz} is the temporary buffer value for the direction i in the voxel (x, y, z).

As for the collecting step, we create only one block representing all the volume (the whole participating medium). Each thread of this block is assigned to one voxel. It collects the result of the propagation step for all the directions.

3.2 Streaming

Memory limitation of GPUs severely restricts the size of volume we can handle at a time. Particularly, to run Algorithm 2 on a 256^3 volume with 25 propagation directions per one LPM face requires 3.8 GB memory. Maximum memory available on most of the GPUs is much less than this size. A solution is to stream portions of the volume to the GPU.

To this end, we split the volume V into sub-volumes B_{ijk} defined as

$$\begin{aligned} B_{ijk} = \{C_{xyz} | \quad & x \in [i \times N_{sub}, (i+1) \times N_{sub}], \\ & y \in [j \times N_{sub}, (j+1) \times N_{sub}], \\ & z \in [k \times N_{sub}, (k+1) \times N_{sub}], \} \end{aligned}$$

Algorithm 3 Our algorithm implemented with CUDA. Orange color is used to underline specific CUDA instructions.

```
Initialize p_prev to 0, the index of the previous iteration
Initialize U^{d,p_prev}_{xyz} and I^d_{xyz} with self emitted radiance if any // p = iteration number
for p ∈ [0, N_iterations[ do
    // Initialisation U buffer of the p_cur^th iteration
    p_cur = (p_prev + 1)%2 // Current iteration index
    Initialize U^{d,p_cur}_{xyz} to 0
    for each LPM do
        parallel for each blocks i do
            parallel for each thread (r,s) do
                Create ray and initialize L^{i,0}_{r,s}
                for each intersected voxel t
                    Compute L^{i,t}_{r,s} from U^{d,p_prev}_{xyz} (Eq. 5.4)
                    Store ray contribution into T^i_{xyz} (Eq. 5.7)
                end for
            end for
        end for
    end for
    Synchronize_blocks()
    // Collecting step
    parallel for each voxels (xyz) do
        for each direction i
            Update U^{d,p_cur}_{xyz} and I^d_{xyz} with T^i_{xyz} (Eq. 5.5)
        end for
    end for
    p_prev = p_cur
end for
```

where C_{xyz} is a voxel at the spatial position (x,y,z) and N_{sub} the size of a sub-volume such that $\bigcup B_{ijk} = V$. Then we group together sub-volumes into slices. This grouping is based on the orientation of the LPM face. For example, for an LPM face perpendicular to the Z axis, we group together sub-volumes into a slice $S_l = \{B_{ijk} | i \in [0, I_{max}], j \in [0, J_{max}], k = l\}$. In this way, the propagation iteration consists of $\frac{Z_{max}}{N_{sub}}$ steps where Z_{max} is the maximum value of coordinate z. Instead of applying Algorithm 2 to the whole volume, we apply it to one slice at a time. That means propagation and collection for one slice must complete before the next slice is processed by the GPU. The radiance propagated from the LPM through a slice is stored at the outgoing face of the slice and this stored radiance is in turn propagated through the next slice, and so on.

The subdivision into sub-volumes make efficient transfer of slices from the CPU memory to the GPU memory as explained in the next section.

CUDA offers concurrent kernels execution and concurrent data transfers on modern GPU. We exploit this mechanism by creating two streams: one to manage the GPU-CPU transfer, another to compute the multiple scattering solution for one en-

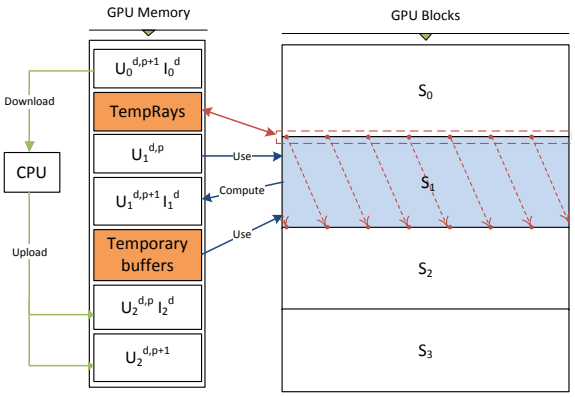


Figure 5.6 – S_i is the i^{th} slice with U_i and I_i the associated data. TempRays is the buffer which stores the radiance at the outgoing face of the previous slice S_0. This radiance is propagated within slice S_1 and finally deposited at the outgoing face of slice S_1. We use two streams represented by green and blue arrows. The green stream is in charge of upload/download data from the GPU memory to the CPU. The blue stream is in charge of the computation of the multiple scattering solution for the slice S_1. The orange buffers are the buffers that are kept in the GPU. Recall that d is the direction number and p the propagation iteration number.

tire slice (Fig 5.6). A stream is a FIFO queue of tasks. The role of the computation stream is to estimate the scattered light within slice S_i, while the transfer stream is in charge of downloading from the GPU the data (I,U) computed for the previous slice S_{i-1}. The computation and transfer streams run in parallel. As soon as the transfer stream completes downloading, its starts uploading the necessary data (I,U) for the next slice S_{i+1}.

In case our solution cannot fit the GPU memory, we split the slice into sub-slices and we stream each sub-slice to the GPU.

4 Implementation and Results

We used CUDA to implement our algorithm. We think that implementing it with OpenCL would be straightforward. We used two graphics cards: GTX 560M with 4 multiprocessors and 2 GB of memory, and GTX 580 with 16 multiprocessors and 1.5 GB of memory. We also implemented the original sequential version of Fattal's algorithm on a CPU i7 -2630QM running at 2GHz.

In our implementation, the user can specify the number of propagation directions for one LPM. Moreover, in case of isotropic phase function, the average radiance I is expressed for 1 direction while the unscattered radiance U is computed for 6 directions. However, in case of anisotropic phase function, the user can specify the number of directions for I and U.

The complexity of our algorithm is equal to $2spn^3$, where s is the scattering order, p the number of propagation directions and n^3 the number of voxels. Regarding the memory occupation, the complexity in bytes is equal to $4*nbSpect*((1+2U+I)n^3+pn^2+pn^3)$ without streaming and $4*nbSpect*((1+2U+I)Sn^2+Spn^2+pn^2)$ with streaming. U and I are the number of unscattered radiance directions and average

radiance directions respectively. S is the slice width along the propagation direction and *nbSpect* the number of wavelengths (RGB in our case).

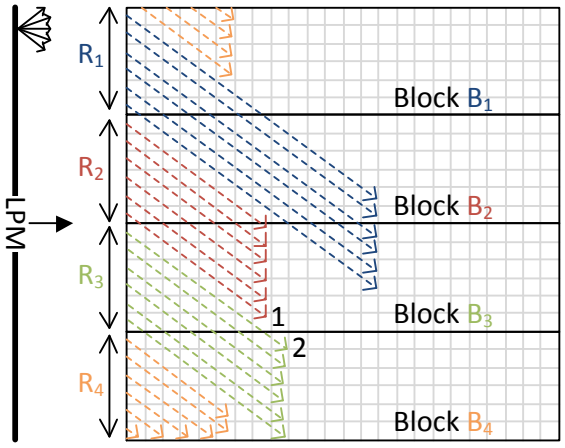


Figure 5.7 – Each block B_i is in charge of propagating rays from a subset R_i of elements (r,s) of a LPM. The rays of same directions at the border of the subset R_i and originating from two contiguous subset R_i may traverse at the same time a same voxel (for example rays 1 and 2).

In order to guarantee only one thread per voxel, all the rays of same direction managed by a same block have to be synchronized whenever they leave their current voxel (Fig. 5.4). This is made possible by using a "___syncthreads()" instruction in their respective threads.

Note that the number of threads in one block is limited due to the number of registers available on a multiprocessor. This is why we reduce the size of a block by bringing down the resolution of one coordinate (r or s) of LPM, which increases the number of blocks for one propagation direction. Consequently, as several blocks may propagate light in a same direction, a voxel, lying at the frontier between two contiguous subsets R_i and $R_{(i+1)}$ of elements (r,s) of the current LPM, may be traversed by at least two rays, one coming from R_i and another from $R_{(i+1)}$ (Fig. 5.7). Each subset R_i is assigned one block B_i. Once again, to guarantee only one thread per voxel, we launch in parallel blocks B_{2i} then blocks $B_{(2i+1)}$. In this way, we avoid to launch contiguous blocks.

In case the LPM resolution is N times higher than that of the voxel grid, to avoid the synchronization problem (due to the traversal of a same voxel by multiples rays) we divide the LPM into N smaller sub-LPM Sl_i and propagate each Sl_i one after the other.

Using $16 \times 16 \times 16$ voxels per sub-volume and 16×16 sub-volumes per slice, the memory needed by our algorithm is far below the available GPU memory (Fig. 5.8).

Furthermore, we use pinned memory as a buffer zone to speedup the data transfer between the CPU and the GPU. When the resulting data are downloaded from the GPU to the CPU, only one transaction is necessary to transfer all the data to the pinned memory. Next, we use *memcpy* operation to update sub-volumes on the CPU memory. Uploading the data onto the GPU memory is performed similarly.

For a volume of 64^3 voxels, 25 propagation directions and 3 bounce multiple scattering, our algorithm takes 417 milliseconds while the Fattal's algorithm running

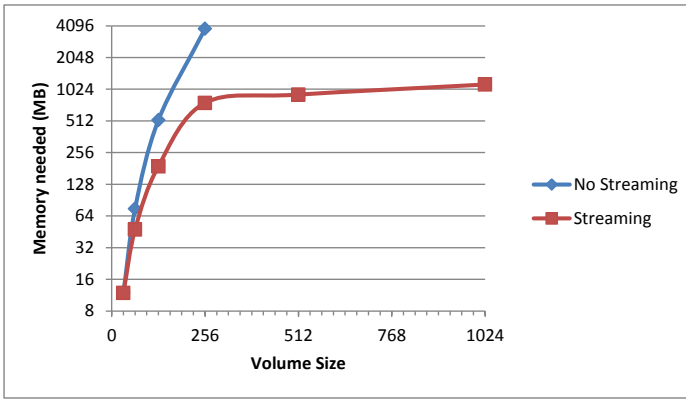


Figure 5.8 – GPU memory requirement for a 25 propagation directions, 6 storage directions (U and I). For the streamed technique, the memory bound of the GPU exceeded. Note that without streaming it is impossible to apply the algorithm on volume sizes larger than 256^3. For the streaming based technique we are far below the available amount of GPU memory. We must point out the fact that the memory requirement of streaming technique increases with volume size (at a much slower rate though).

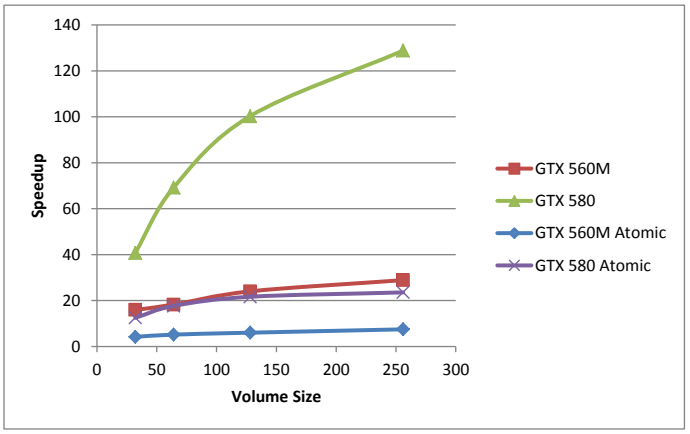


Figure 5.9 – Summary of the speedup between the original CPU algorithm and our implementation on 2 GPUs: GTX 560M and GTX 580. The green and red plots have been obtained with our parallel approach, while the blue and purple ones with atomic operations. We can observe that the obtained speedup increases with the volume size. This is due to the fact that a large number of blocks are created during the propagation step, which makes the GPU computing resources more and more busy. Note the efficiency of our approach compared to an atomic operation-based version.

on the CPU takes 47 seconds. We see in Fig. 5.9 that the speedup grows with the volume resolution. This can be explained by the fact that the more the blocks, the better is the balancing of the multiprocessors loads. Examples of rendered images are given in fig. 5.10. Other results are shown in the additional materials. To validate our GPU-based parallel algorithm, we have computed the RMSE error between the CPU-based solution and our method. We found an RMSE of about 0.005.

Figure 5.10 – Results of two 128^3 participating medium lit by an environmental map. It took approximatively 2300 ms to compute 3 scattering orders with 25 propagation directions.

5 Conclusions & Further works

We propose a novel parallel algorithm to render participating media with multiple scattering. It is a parallel version of Fattal's algorithm. Compared to CPU-based Fattal's algorithm, we obtain a speedup of 1 to 2 orders of magnitude. Our algorithm is capable of interactively rendering volumes of 64^3 voxels. We propose a novel streaming technique based on the concept of sub-volumes, slices, and sub-slices to handle any large size volume.

As future work, our approach could use a hierarchical representation of participating media to handle huge volumes more efficiently. Moreover, a compression approach would allow to reduce the time needed for transferring data from the CPU to the GPU and vice versa. Finally, it would be interesting to extend our approach to handle solid objects inside participating media.

Acknowledgement

The material in this chapter is, in part, a reproduction of the material published in: Adrien Gruson, Ajit Hakke Patil, Rémi Cozot, Kadi Bouatouch and Sumanta N Pattanaik "Light Propagation Maps on Parallel Graphics Architectures", EGPGV 2012 [GPC+12]. This work was partially supported by US National Science Foundation grant IIS-1064427.

Since then, several other papers have been published to address participating media rendering. For example, Hakke-Patil et al. [HPBC+13] have proposed a parallel implementation of the DOM technique proposed by Languenou et al. [LBC94]. In their article, they compare their technique to our method. Their technique allows to obtain a less significant gain when an environment map is used. However, it handles strong directional lighting more efficiently.

Weber et al. [WKSD13] use the instant radiosity technique together with Adaptive Volumetric Shadow Maps (AVSMs) [SVLL10]. Their technique is completely implemented with DirectX. To achieve interactive performance, the authors used only a small number of VPL place adaptively. Moreover, to handle changes in the transfer function, they use a progressive VPL update scheme.

Elek et al. [ERDS14a, ERDS14b] propose a technique that combines VPLs and DOM techniques. Their technique decomposes lighting into virtual directional light sources and VPLs. The contribution of these virtual light sources are propagated using independent discrete propagation volumes.

Progressive volume photon tracing 6

In this chapter, we will present a novel approach to volume rendering based on progressive photon mapping. Rendering participating media (volume data) with multiple scattering in a reasonable time is still a challenge. In the previous chapter, we have presented a GPU-based method faster than Monte Carlo methods. However, this technique suffers from several limitations: no handling of objects inside the participating media, no lighting interaction between a medium and its environment etc. Consequently, it cannot be used for complex scenes.

In order to overcome these issues, we propose a new rendering method based on progressive photon mapping (already presented in section 3.3, page 44). Recall that a photon mapping technique [Jen01] consists of two passes. In the first pass, a number of photons are shot, then stored in a Kd-tree data structure. During the second pass, rays are traced from the camera, and at each intersection point on an enough smooth BSDF, we use the photon map to approximate the incident radiance. This technique has been first applied to surfaces, then extended to participating media (*Volume Photon Mapping* [JC98]). In this chapter, we will focus on photon mapping-based techniques used to render participating media.

As this kind of technique is very demanding in terms of memory storage, a progressive scheme has been proposed [HOJ08, HJ09, KZ11] to cope with this problem. Moreover, when rendering participating media, the integration domain has one more dimension than for surface rendering. By taking advantage of this extra dimension, some techniques propose an efficient Monte Carlo estimator. In addition, to render participating media, the transmittance need to be evaluated by ray marching. However, this operation is costly and needs to be avoided as much as possible.

The goal of our new technique is to propose a novel approach to *progressive photon mapping* for scenes containing both surfaces and volume objects. Our method allows to handle scenes made up of glossy, specular and refractive objects as well as homogeneous and heterogeneous participating media. By making use of two Kd-trees (built in a preprocessing step) to store view beams (camera rays intersecting the medium) and gather points (camera rays intersecting the surfaces). These data structures are independent of the complexity of the scene in terms of number of objects and light sources and they depend only on the image resolution. Moreover, we take advantage of these data structures to drive the photon shooting process by considering the photon visibility as an importance function (similarly to [HJ11]). Finally, we demonstrate that our method can be easily combined with the most recent particle tracing approaches such as [JNT+11] and that it speeds up the rendering process in case of complex lighting[1].

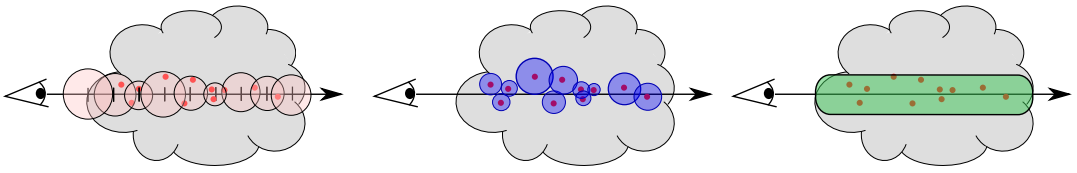
[1] by "complex lighting", we mean all scenes where it is difficult to find a contributive light path.

The next section provides information about the related works while section 2 gives a technical background on photon mapping techniques for participating media rendering. An overview of our approach is given in section 3, while details on our method are presented in section 4. Finally, results are given in section 5 before concluding.

1 Related work

The first method based on photon mapping and coping with participating media is *Volume Photon Mapping* [JC98]. It allows to simulate interactions between photons and participating media and to build a volume photon map. However, its costly rendering step consists in casting a ray from the camera through the volume and in using ray marching to gather the photons (stored in the photon map) close to the ray to compute their contributions (fig. 6.1a). Volume photon mapping yields good results but it suffers from the same limitations as those of surface Photon Mapping: a huge number of photons is needed to render complex scenes. As a large number of photons requires a lot of memory, the rendering quality is limited by the available memory of the used computer. Moreover, ray marching, needed to retrieve nearby photons and media parameters, is very expensive. The smaller the ray marching steps, the better the results, but also the longer the rendering. So, *Volume Photon Mapping* has two limitations: high computation time and large memory requirement.

The *Beam Radiance Estimate* [JZJ08a] method is one way to reduce the rendering time. In this method, each photon has an influence area which is a disk of variable radius depending on the photons density (fig. 6.1b). An isolated photon is assigned a large radius while a smaller one is assigned to photons that are close to each other. To estimate the radiance of a ray cast from the camera, all the photons, within a distance from the ray that is smaller than their radius, are considered. This method speeds up the rendering process and also adapts to the density distribution within the volume and to the lighting conditions. When rendering scenes with sparse photon distribution, the *Beam Radiance Estimate* gives better results than the usual ray marching but remains costly when used for complex scenes. Indeed, the contribution of a ray needs the traversal of a hierarchy of spheres which takes time for a high number of photons. Moreover, this technique is also limited to the memory available on the computer.



(a) Volume Photon Mapping **(b)** Beam Radiance Estimate **(c)** Our method

Figure 6.1 – Different ways to gather photons (points representation, for other presentations see the related work section). Regular ray marching (a) may take into account several time a same photon whereas *Beam Radiance Estimate* (b) and our method (c) gathers all photons once and for all.

(Stochastic) Progressive Photon Mapping [HOJ08, HJ09] (PPM or SPPM), ad-

dresses the memory limitation problem. It is a progressive rendering method consisting in computing a noisy image that is improved after each pass. At each pass, only a limited number of photons are emitted then discarded when running the next pass. Therefore, only the photons emitted at the current pass are saved in memory. To each visible point (aka gather point) is assigned a disk, and only the photons within this disk are used to compute the radiance at this point. The radius of a disk decreases after each pass to reduce the bias of this method. *A Probabilistic Approach* [KZ11] (APA) of photon mapping is also a progressive method that handles scenes containing participating media. It allows to render many noisy images of a same scene that are merged to get a final image. Their approach is more general than SPPM but less adaptive to the local photon density.

Another interesting approach, called *Progressive Photon Beams* [JNT+11] (PPB), is close to ours but proceeds differently. It consists in shooting photon beams (instead of photons) from the light sources and storing them in a BVH data structure. For radiance estimation, the visible photon beams are decomposed into those that are handled using GPU through rasterization, and those handled by the CPU ray tracer.

Another volume rendering method [NNDJ12b] relies on VPLs (Virtual Point Light) [Kel97]. The method considers the light paths as VRLs (Virtual Ray Lights) rather than photon beams. Some improvements of this method have been proposed in [NNDJ12a]. This is an interesting method that proceeds differently from our approach.

Discussion: While the above methods provide good results, they still suffer from efficient sampling issue. All the above related techniques use local path construction to build light paths. However, this strategy can fail in case of difficult visibility from the light sources. In our approach, storing view beams in a Kd-tree allows to extend to participating media the robust adaptive photon tracing approach proposed by Hachisuka and Jensen [HJ11]. These authors employ the photon path visibility as an importance function to better sample the path space. They also use adaptive Markov Chain Monte Carlo to determine the best mutations strategies. In addition, their approach relies on the replica Exchange Monte Carlo to avoid that path sampling gets stuck at local peaks of the target function (Already discuss in chapter 4).

2 Background

The radiance reaching a point inside a participating medium is computed by solving the radiative transfer equation [Sub60] (already presented in section 3, page 27). Given a ray starting at a point $\vec{x}_e$ and going through the medium until a point $\vec{x}_s$, the RTE provides the radiance at $\vec{x}$ in direction $\vec{\omega}$ as follows :

$$\begin{aligned} L(\vec{x}\leftarrow\vec{\omega}) &= \underbrace{\tau(\vec{x}_e\leftrightarrow\vec{x}_s)L(\vec{x}_0\rightarrow\vec{\omega})}_{L_s(\vec{x}\leftarrow\vec{\omega})} \\ &+ \underbrace{\int_{\vec{x}_e}^{\vec{x}_s}\tau(\vec{x}_e\leftrightarrow\vec{y})\sigma_s(\vec{y})L_i(\vec{y}\leftarrow\vec{\omega})\mathrm{d}\vec{y},}_{L_m(\vec{x}\leftarrow\vec{\omega})} \end{aligned} \tag{6.1}$$

Table 6.1 – Definition of quantities used in this chapter

	Symbol	Description
Physical params	Φ_i, $\vec{x}_i$, $\vec{\omega}_i$	Flux, position and incoming direction of photon i.
	$\sigma_a(\vec{x})$, $\sigma_s(\vec{x})$, $\sigma_t(\vec{x})$	Absorption, scattering and extinction coefficient at the position $\vec{x}$.
	$\tau(\vec{x}\leftrightarrow\vec{x}')$	Transmittance between the points $\vec{x}$ and $\vec{x}'$.
	$\rho(\vec{x},\vec{\omega}'\rightarrow\vec{\omega})$	Normalized phase function if $\vec{x}$ is inside a medium.
	$f_r(\vec{x},\vec{\omega}'\rightarrow\vec{\omega})$	BRDF if $\vec{x}$ is on a surface.
	$\mathrm{k_s}(d,r)$, $\mathrm{k_v}(d,r)$	Density kernel for the surfaces and the volume. r is the support of the kernel.
Algorithmic values/parameters	P	Pixel used to attach rendering data.
	G_j,B_j	Gather point and collection of beams attached to the pixel P during the j^{th} iteration
	$L_s(G_j), L_m(B_j)$	Radiance at a surface visible point or in a medium beam respectively.
	$L_s(P), L_m(P)$	Cumulative radiance at a visible point or a beam at the current iteration only.
	$N_s(P)$	Number of contributive photons to the pixel P since the beginning of the rendering.
	$M_s(G_j)$	Number of contributive photons to the gather point G_j at the j^{th} iteration only.
	$r_{s,j}(P), r_{m,j}(P)$	Radius of a disk associated with a visible point or of a beam in the j^{th} iteration.
	α	User parameter to control the convergence rate.

where $\vec{x}_0$ is the back surface point and $L_i(\vec{y}\leftarrow\vec{\omega}) = \int_\Omega \rho(\vec{y},\vec{\omega}_i\rightarrow\vec{\omega})L(\vec{y}\leftarrow\vec{\omega}_i)\mathrm{d}\sigma(\vec{\omega}_i)$ the incoming radiance at $\vec{y}$ and scattered in the direction $\vec{\omega}$. Other terms are defined in table 6.1. For the sake of clarity, equation 6.1 doesn't take into account the radiance self-emitted by the medium which is straightforward to compute. We split this equation into two terms: L_s, radiance coming from the back surface, and L_m, radiance from scattering event in the medium. In the classical approach [JC98], two different sets of photons are used: one for surface estimation and one for volume estimation (we assume here to use the same amount of photon N_{photon} for the two sets).

L_s can be computed using density estimation as in classical *Photon Mapping* [Jen01] and is attenuated by the medium:

$$L_s(\vec{x}\leftarrow\vec{\omega}) \approx \frac{\tau(\vec{x}\leftrightarrow\vec{x}_0)}{N_{\text{photon}}} \sum_{i=0}^{N_{\text{photon}}} \mathrm{k_s}(\vec{x}_i-\vec{x}_0,r) f_r(\vec{x}_i,\vec{\omega}_i\rightarrow\vec{\omega})|\vec{n}\cdot\vec{\omega}_i|\Phi_i, \tag{6.2}$$

where r is the kernel support used during the density estimation.

With the set of volume photons, L_m can be expressed similarly to *Beam Radiance Estimate* method [JZJ08a]:

$$L_m(\vec{x}\leftarrow\vec{\omega}) \approx \frac{1}{N_{\text{photon}}} \sum_{i=0}^{N_{\text{photon}}} \mathrm{k_v}(\vec{x}_i-\vec{x}'_i,r)\tau(\vec{x}\leftrightarrow\vec{x}'_i)\sigma_s(\vec{x}'_i)p(\vec{x}_i,\vec{\omega}_i\rightarrow\vec{\omega})\Phi_i, \tag{6.3}$$

where $\vec{x}_i$ the photon position and $\vec{x}'_i$ the projection of the photon position $\vec{x}_i$ on the view ray. Moreover, contrary to [JZJ08a], the radius r is not adapted for each volumetric photons.

3 Overview

Our method is depicted by the global algorithm 4. We have made the choice (explained later) to use a splatting operation of the photon contribution during the light tracing. So, we need to store the camera paths (volume and surface interactions) into data structures and use them during the photon generation.

At each iteration, a preprocessing step is performed and detailed in section 4.1. First, new camera paths are generated (TraceCameraPaths in algorithm 4). Then, with these new paths, we build different hierarchies (BuildHierachies in algorithm 4). For rendering surfaces, our method assigns a disk[2] to every visible point lying on a surface. For volumes, we use beams rather than disks. To each ray of a camera path, cast through a pixel and traversing a medium, we assign a cylinder. This cylinder is called beam from now on. These beams get reflected/refracted if the associated ray hits a surface not enough smooth (*i.e.* specular or fine glossy surface). We build a beam Kd-tree whose leaves are the resulting beams. When a ray hits an enough smooth surface, the resulting hit point is registered and assigned a disk-shaped influence zone. We build another Kd-tree, called gather point Kd-tree, whose leaves are those hit points. As the results of camera path generation, we may have one gather point G_j and a collection of beams B_j for each pixel P.

Then, for the photon shooting step, two different sets of photons are shot. Whenever one of those photons interacts with a volume (or a surface), the top-down traversal of the associated Kd-tree leads to beams (or gather points), which allows to add photon contribution on impacted beams (or gather points) (SplatPhotonContribution function). Then, the photon is discarded instead of stored in a photon map. Further details are given about these updates in section 4.2.1. To be more efficient, every photons are sampled using Metropolis approach (TraceLightPathVisibility function), to better sample the contributive light path domain, similarly to [HJ11].

Once all the photons have been shot, we need to update the pixel radiance by collecting contributions received by associated gather point G_j and beam collection B_j (UpdateRadiance function). Moreover, to have a consistent estimator, we need to decrease the associated gather point (or beam) radius at each iteration (UpdateRadii function). Like in APA [KZ11], we use a same radius $r_{m,j}$ for all the beams. As regards the surface radii, their value depends on the used estimator. If we use (like for beams) the APA estimator, a global radius $r_{s,j}$ is assigned to all the gather points. Otherwise, if we use the SPPM estimator, different radii are assigned to the gather points. In our implementation, we have chosen to use SPPM estimator for surface component. These steps are described in sections 4.3.1 and 4.3.2.

In *Volume Photon Mapping*, once the photon map is built a ray is cast through each pixel and a costly ray marching (or a *Beam Radiance Estimate*) is used to determine the relevant photons used to compute the radiance associated with the ray.

[2] the radius of the disk is equal to the support of the current surface kernel

Rather, in our approach as soon as a photon interacts with a medium, it is assigned to one or more beams by going down the beam Kd-Tree, and its contribution to the scene is computed straightaway. This assignment operation spares the use of ray marching (fig. 6.1).

Algorithm 4 main()

```
1: for int i = 1 to nbPass do
2:     // Step 1: Preprocess
3:     TraceCameraPaths()
4:     BuildHierachies()
5:
6:     // Step 2: Visibility-driven Photon shooting step
7:     for space in {surface,volume} do
8:         for int i = 1 to nbPhoton do
9:             TraceLightPathVisibility()
10:            SplatPhotonContribution()
11:        end for
12:    end for
13:
14:    // Step 3: Collect statistics
15:    UpdateRadiance()
16:    UpdateRadii()
17: end for
```

On the one hand, the use of a beam Kd-Tree makes faster the estimation of the radiance assigned to each beam. On the other hand, photon assignment requires the traversal of this hierarchy which takes time. Overall, our hierarchy-based approach is far faster than ray marching as seen in the results section.

4 Implementation details

For clarity purpose, this section consider only photons. Other particles model (beams [JNT+11]) will show in section 5.

4.1 Preprocessing step

Before starting any rendering we need to initialize all the needed data structures. A view ray, traced from the viewpoint through a pixel (going through the participating media or not), is repeatedly reflected if the surface is not enough smooth. When the view ray crosses a medium it is assigned a cylinder beam. At the end, a camera path is composed of a surface end point G_j (called gather point) and a collection of cylinder beams B_j (fig. 6.2). This collection contains all the line segments of the camera path which intersect a medium. We note this collection $B_j = \{B_{j,1}, \ldots, B_{j,\text{nbB}}\}$ where nbB is the number of beams. When computing the ray paths, the final intersection point with sufficient smooth surface, called gather point, are registered to

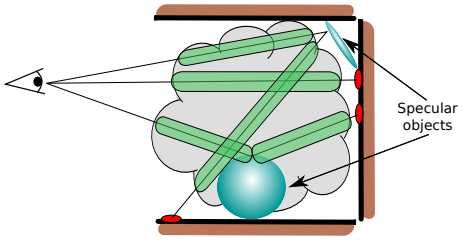
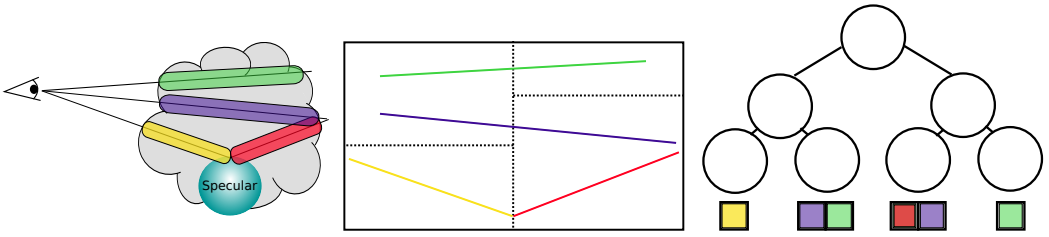


Figure 6.2 – Different possible view rays in a scene. The rays going through the medium are assigned beams (in green), and gather points (in red) are created on sufficient smooth surfaces.



(a) Creation of the view beams **(b)** Splitting planes **(c)** Final tree

Figure 6.3 – Example of a beam Kd-tree building for a set of beams (a). Only the supporting rays are used in the tree (b). Some leaves (c) contains several beams, and beams crossing splitting planes are duplicated in the tree.

compute a gather point Kd-tree as explained hereafter. Moreover, for all beams and gather points, we store the total transmittance value Tr of the camera path before creating it. For example, a gather point will store the total transmittance value of the camera path.

Once the path tracing has been completed, we obtain for all the pixels, a set of beams and a set of gather points. These two data structures are computed in one single pass. We build a Kd-tree from the gather points and a beam Kd-tree from the beams. Each gather point P is assigned a disk of a radius $r_{s,j}(P)$. This radius is computed using ray differential and locally adapted by the photon density (this is explained later). A photon within this disk contributes to the associated gather point. A photon contributes to a beam if it lies in its associated cylinder with a radius $r_{m,j}(P)$. This radius is the same for all the beams.

Recall that the radii associated with disks and beams decrease after each rendering pass. The gather point Kd-tree is constructed using a classical way using a SAH Kd-tree based on the gather point influence zone.

The beam Kd-tree however is created following the method from Havran et al. [HBS04], a beam being represented by a line segment supported by its associated ray (beam axis). The endpoints of this line segment are those of the beam (fig. 6.3b). The first step of building the beam tree is to find the axis-aligned bounding box of all the beam segments, which represents the root of the tree. Then, a splitting plane is chosen along the largest axis of that bounding box. All the beam segments on one side of that plane are assigned to one child, and the beams crossing this plane are assigned to both children. The splitting operation is recursively repeated until one of the stopping criteria is met (fig. 6.3c):

- The number of beams in the current node is below a threshold (typically 32 in our implementation).
- The depth of the current node reaches a maximum value (between 20 and 30 for our scenes).

Each beam is assigned a data structure which contains a pointer to the associated beam computed in the preprocessing step. To compute the contribution of a photon to the beams, the beam Kd-tree is traversed top-down. For each node, If a photon is located on one side of the associated splitting plane and its orthogonal distance to this plane is larger the global beam radius $r_{m,j}(P)$, then the associated sub-tree is traversed, otherwise we traverse the two subtrees. For each leaf node reached by the traversal, we compute the contribution of the photon to the beams corresponding to the beam segments within this leaf. Moreover, to avoid multiple contributions of the same photon to a given beam, we use its projection on the the beam segment. Indeed, if the projected point for a given beam is outside the leaf bounding box, the photon does not contribute to this beam.

4.2 Visibility-driven Photon shooting step

The photons are shot from the light sources using the method presented in [HJ11]. When a photon enters a participating medium, the hierarchy is traversed top-down, and its contribution is brought to the leaf beams containing it. When it hits a surface, we compute its contribution to the gather points by a top-down traversal of the gather point Kd-tree. In this way, as soon as a photon contributes to a gather point or a beam, its contribution is accumulated and the photon is discarded.

We use the photon visibility information as an importance functions (chapter 4) for the Metropolis sampling process. Unlike in Hachisuka et al. [HJ11] technique which handles only surface interactions with one Markov Chain, we use two different Markov Chains, one for the surfaces and another for the participating media. These two Markov chains are independent. For the second Markov Chain, we use the beam visibility as the importance function.

Moreover, we use also Adaptive MCMC to find optimal mutation size automatically. In our case, we also use two independent Adaptive MCMC to tune the mutation size for participating media or surface rendering. Indeed, these two component dimensions can be different, which leads to two different optimal mutation sizes.

On top of that, similarly Hachisuka et al. work [HJ11], we use replica exchange to avoid to get stuck inside importance function peaks. Moreover, we use uniform chains to compute the two normalization factors: $b_{s,j}$ for the surface chain and $b_{v,i}$ for the volume chain.

4.2.1 Radiance update

When a photon hits a surface, we compute its contribution to the gather points (nodes of the view Kd-tree whose disk contains the photon) by a top-down traversal of the gather point Kd-tree. In this way, as soon as a photon contributes to a

gather point or a beam, it is discarded. The radiance of this latter is updated using eq. (6.2):

$$L_s(G_j) += Tr(G_j) \cdot \mathrm{k_s}(\vec{x}_i - \vec{g}_j, r_{s,j}(P)) \cdot \tau(\vec{x} \leftrightarrow \vec{x}_s) \cdot f_\mathrm{r}(\vec{x}_i, \vec{\omega}_i \rightarrow \vec{\omega}) \cdot |\vec{n} \cdot \vec{\omega}_i| \cdot \Phi_i \quad (6.4)$$

where $\vec{g}_j$ is the position of the gather point G_j and $Tr(G_j)$ is the total transmittance carried by the camera path before the gather point G_j. In our current implementation, we use a constant kernel $\mathrm{k_s}(\vec{x}_i - \vec{g}_j, r_{s,j}(P))$.

When a photon lies in a beam $B_{j,k}$, it participates in the update of the cumulative radiance L_m of the beam attached to a collection B_j using eq. (6.3):

$$L_m(B_j) += Tr(B_{j,k}) \cdot \mathrm{k_v}(\vec{x}_i - \vec{x}'_i, \cdot r_{m,j}(P)) \cdot \tau(\vec{x} \leftrightarrow \vec{x}'_i) \cdot \sigma_s(\vec{x}'_i) \cdot p(\vec{x}_i, \vec{\omega}_i \rightarrow \vec{\omega}) \cdot \Phi_i, \quad (6.5)$$

where $\vec{x}'_i$ is the projection of $\vec{x}$ onto the axis of the beam $B_{j,k}$.

In case of heterogeneous medium, for a reason of efficiency each beam data structure stores a lookup table of the transmittance along the beam axis to quickly determine the transmittance $\tau(\vec{x} \leftrightarrow \vec{x}')$.

4.3 Collecting statistics

Once a sufficient number of photons have been shot for the current pass (iteration), the resulting image can be updated. This step consists in updating the radiances attached to the pixel P: $L_s(P)$ from the surface and $L_m(P)$ from the participating media. Moreover, we also update the radius values for the next iteration.

4.3.1 Image update

In our method, we have chosen the APA estimator for beams. However, to one pixel P corresponds a collection of beams B_j whose contributions yield the radiance $L_m(B_j)$. Note that if the camera path does not cross any participating medium $L_m(B_j) = 0$. The beam radiances associated with the pixel P are updated by averaging over the different iterations:

$$L_m(P) = \frac{L_m(P) \cdot (j-1) + b_{v,i} \cdot \frac{L_m(B_j)}{N_v}}{j} \quad (6.6)$$

where N_v is the number of photon used for the volume rendering and $b_{v,i}$ is the normalization factor. This equation is valid for the update of surface radiance if APA estimator is used for surfaces. In our implementation, it is not the case and the SPPM estimator is used for surfaces. The unnormalized accumulated surface radiance is computed as

$$\widetilde{L}_{s,j}(P) = (\widetilde{L}_{s,j-1}(P) + b_{s,i} \cdot L_s(G_j)) \cdot \frac{N_s(P) + \alpha M_s(G_j)}{N_s(P) + M_s(G_j)} \quad (6.7)$$

where $b_{s,i}$ is the normalisation factor. And the final surface radiance for the pixel P for the iteration j is finally computed as

$$L_s(P) = \frac{\widetilde{L}_{s,j}(P)}{\pi \cdot r_{s,j}(P)^2 \cdot N_e} \quad (6.8)$$

where N_e is the total number of paths emitted for surface rendering.

Finally, the final radiance for a given pixel P is equal to:

$$L(P) = L_s(P) + L_m(P) \tag{6.9}$$

4.3.2 Radius update

Finally, in order to achieve convergence, we need to update the global radius assigned to beams as explained in [KZ11]:

$$r_{m,j}(P) = r_{m,j-1}(P) \cdot \sqrt[3]{\frac{j+\alpha}{j+1}}, \tag{6.10}$$

where j is the number of the current iteration.

For the surfaces, the radius reduction formula depends on the choice of the estimator (APA or SPPM). In our method, we chose the SPPM estimator for surfaces, so the surface radius update is:

$$r_{s,j}(P) = r_{s,j-1}(P) \cdot \sqrt{\frac{N_s(P) + \alpha M_s(G_j)}{N_s(P) + M_s(G_j)}} \tag{6.11}$$

Moreover, for the surface radius, we modulate the radius by using ray differential.

5 Results

In this section we show some results obtained with our method using different shooting techniques (without and with metropolis optimisation) and different kinds of particle such as photon or photon beams. In this way, we demonstrate that, although our method has been designed for photon tracing, it has been easily extended to photon beam tracing. We compared our method with PPB, which is the most related method, and APA which is another interesting approach, easy to implement since it is just a loop consisting in running a classical volume photon mapping. We use the following notations concerning different variants of our method:

PPT our approach where photons are traced from the light sources

PPBT our approach where photon beams are traced

PPT_metro our approach with photon tracing and Metropolis

PPBT_metro our approach with photon beam tracing and Metropolis

We have implemented our methods as well as APA and PPB using the Mitsuba renderer framework [Jak10]. The different parameters used for each scene are given in table 6.2 ($\alpha = 0.7$ for all the methods). The results have been obtained on a computer supplied with two 2.4 GHz Intel Xeon E5645 CPU (12 cores). Each method is executed in a multi-threading context to use the 12 cores of the computer. Finally, for each scene, one reference image has been generated using path tracing. We compare the methods in terms of convergence speed.

	# polygons	resolution	# VP	# PB	# SP	time
Dragon smokes	100k	768x768	50k	5k	100k	1h.
Breakfast hall	1600k	1080x1920	100k	10k	100k	10h.
Kitchen	250k	720x1280	100k	10k	500k	10h.

Table 6.2 – Rendering parameters: each method is stopped at the same rendering time (rightmost column). *# VP* = number of volume photons shot per pass in APA and PPT; *# PB* = number of photons beams in PPB and PPBT; *# SP* = number of surface photons shot per pass.

In the *dragon smokes* scene (shown in fig. 6.5), as the lighting conditions are quite simple the metropolis optimisation is not really useful (except for the smoke which covers a small part of the scene). Indeed, the majority of the photon paths are visible, so few of them need mutation. However, as shown in fig. 6.4, our method (with different variants) performs as well as APA or PPB. In the two next scenes, *breakfast hall* and *kitchen* (figs. 6.6 and 6.7), the lighting conditions are more complex. In the breakfast scene, light goes through the windows and the hall is filled with a homogeneous medium. The outside is represented by a horizontal large plane a small part of which is visible through the windows. This scene is challenging for Metropolis as well as for uniform sampling based methods. Indeed, for gather points lying outside, replica exchange builds a lot of useless paths that mutate outdoors while they are less contributive to the final image. This is why PPT without Metropolis performs as well as PPT_metro in the beginning of the progressive process (fig. 6.4). We will present a solution of this problem in the next chapter. As shown in this figure, our method PPT_metro converges faster than APA and PPB. The kitchen scene is lighted, through the double-glazed window, by a clear sky and the sunlight. The kitchen is filled with a heterogeneous medium. In fig. 6.7, we show results obtained with different methods. We can see that Metropolis gives better visual results.

Discussion and future works All the six presented methods suffer from the limitation of photon mapping. This limitation is a starting bias highly depending on the initial parameters (initial radii, number of photons, α, etc.)

Our experiments showed that AMCMC (Adaptive Markov Chain Monte Carlo) converges toward a small mutation size. This is why we initialize AMCMC with a mutation size smaller than the one proposed in [HJ11] (0.1 gives good results). An interesting future work is to propose a better adaptive process to converge faster toward an optimal mutation size. Finally, using only visibility as an importance sampling function can fail in some cases. Indeed, visible photon paths may have a weak contribution to the final image (due to glossyness or transmittance in the volume). Finding a new importance sampling function which takes into account these cases is a challenge. A new importance function will be proposed in the next chapter.

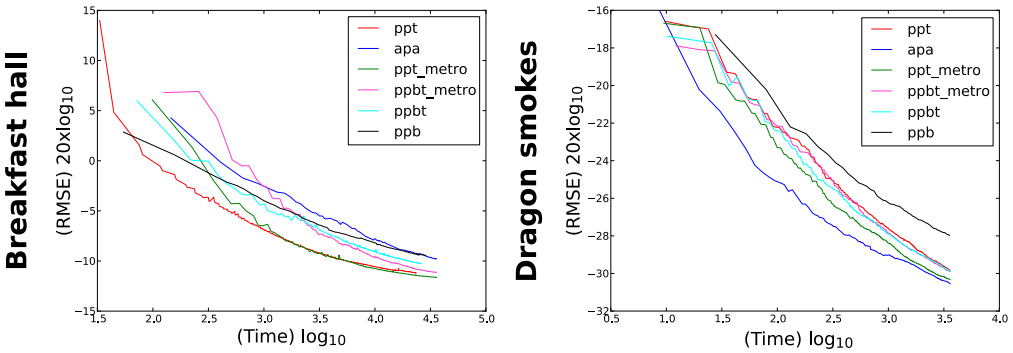


Figure 6.4 – Plots of the RMSE (between a reference image generated using path tracing and all the methods) as a function of time.

6 Conclusion

We have proposed a global illumination method that handles scenes containing surface objects of any material and participating media (homogeneous and heterogeneous). The method computes two data structures in a preprocessing step: gather point Kd-tree and beam Kd-tree. No photon maps, nor volume photon maps are stored in memory. Moreover, the beams that are not visible from the viewpoint are not computed. To increase efficiency, our method uses Metropolis and visibility information to guide the photon shooting process. The results have shown that our approach converges faster to the same solution (image of a given RMSE) than the one obtained with APA [KZ11] and with the CPU version of PPB [JNT+11]. Even though our method has been implemented on the CPU, the obtained results demonstrate that our approach is fast. Another interesting feature of our method is that it could be easily transformed from progressive to interactive. More precisely, the user can set the number of photons to a very high value (this is possible since the photons are not stored), launch the programme, interrupt it whenever he wants to display a resulting image, then restart it to get a better image, and so on.

Acknowledgements

The material in this chapter is, in part, a reproduction of the material published in Charly Collin, Mickael Ribardiere, Adrien Gruson, Remi Cozot and Kadi Bouatouch "Visibility-driven progressive volume photon tracing", CGI 2013 [CRG+13]. Charly Collin was supported in part by NSF grant IIS-1064427.

In this project, I was principally involved in the Metropolis part of the algorithm. Indeed, the idea of this paper were to extend the progressive photon mapping to participating media. However, during the project, several other methods was published [JNT+11, KZ11, NNDJ12b]. So, we exploited the fact that our technique provides the visibility information during the photon shooting process to build a metropolis sampling. With this enhancement, the technique is able to sample difficult visibility scenes where other methods fail.

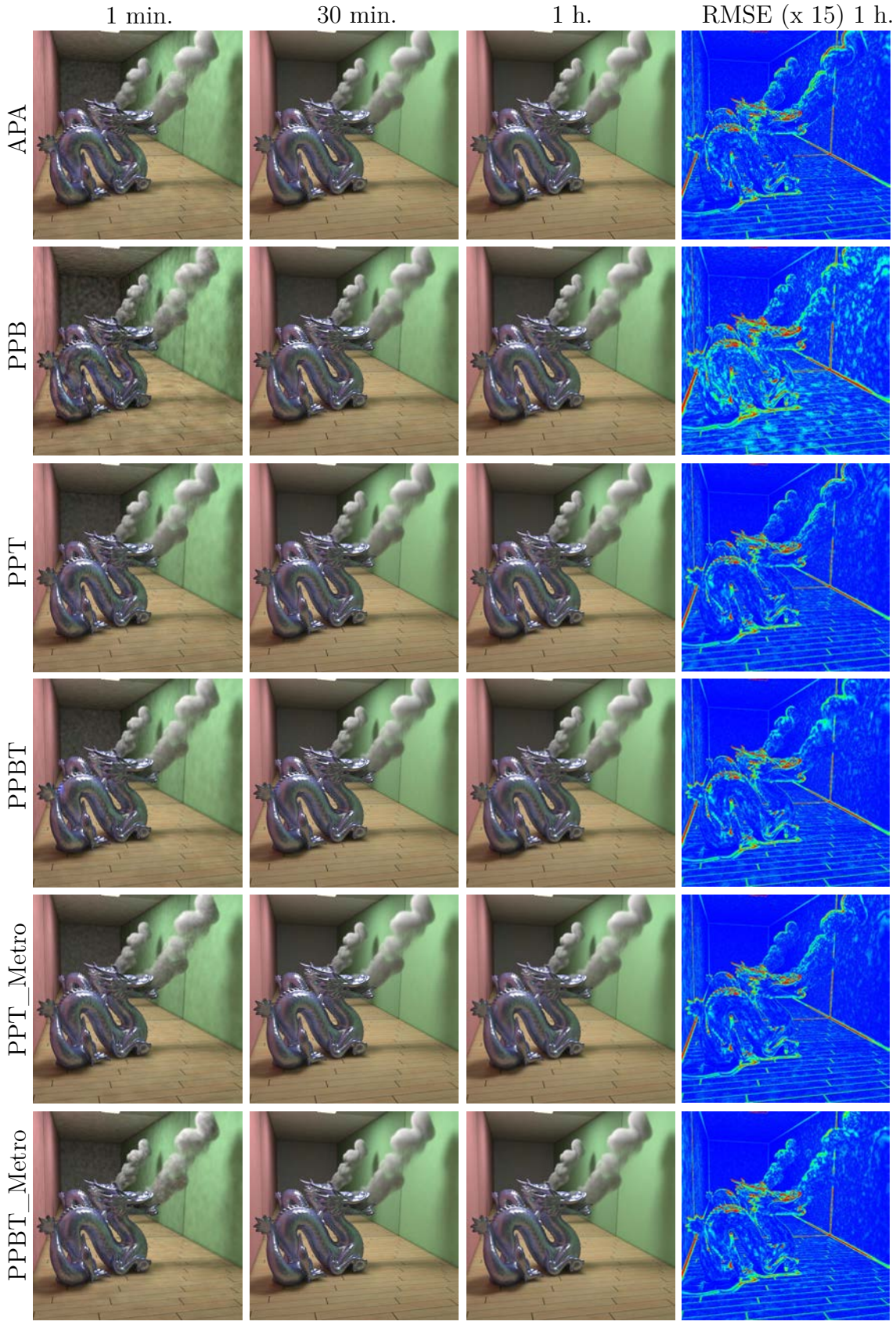


Figure 6.5 – Results obtained for the "dragon smokes" scene. The rightmost column shows the RMSE in false colors.

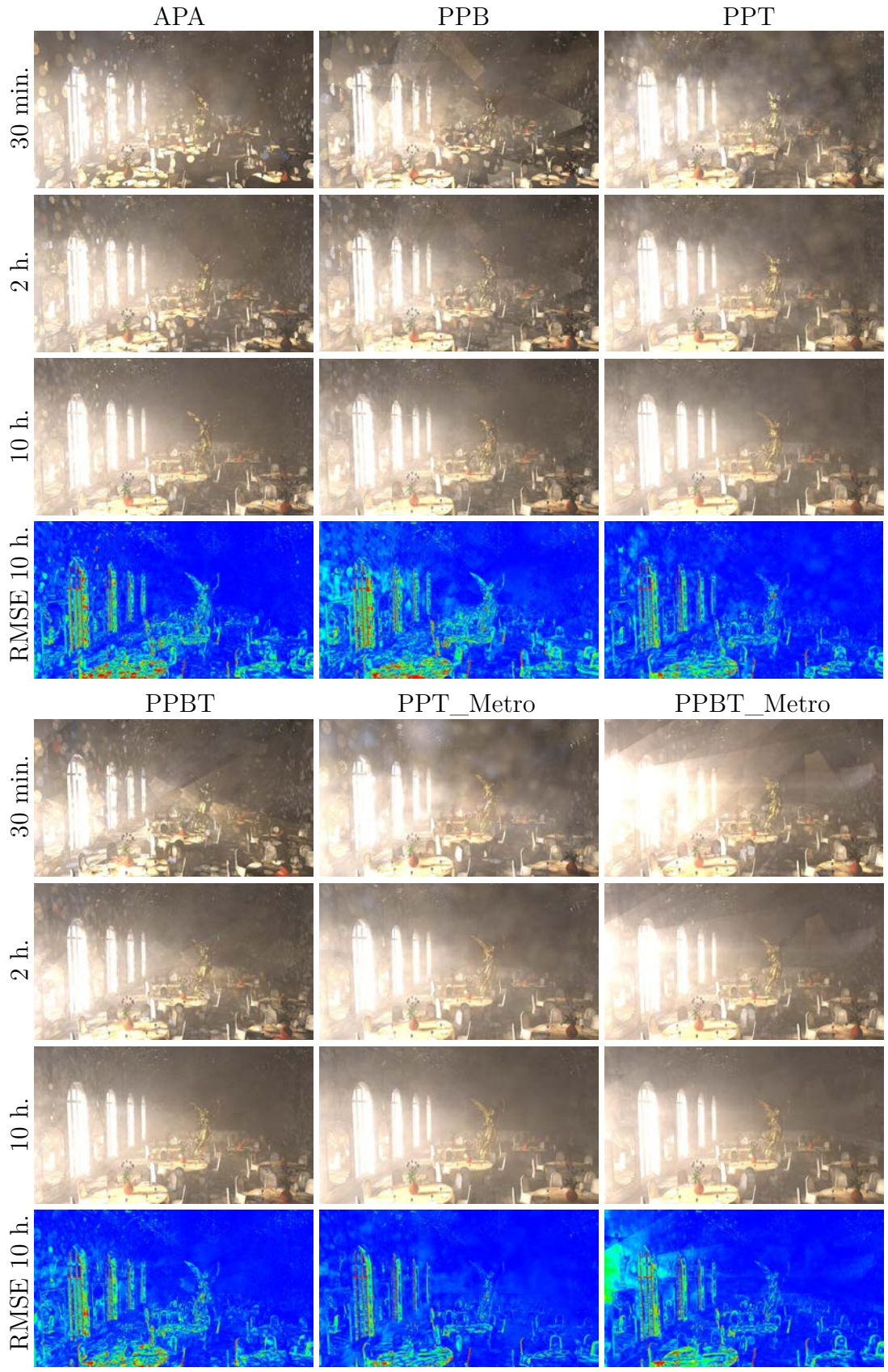


Figure 6.6 – Results obtained for the breakfast hall scene (courtesy of Greg Zaal).

Figure 6.7 – Results obtained for the kitchen scene (courtesy of Jay-Artist).

A spatial importance function for MLT

7

The recent widespread adoption of Monte Carlo light transport simulation in the practice of realistic rendering has revealed a number of limitations of the existing algorithms. One of the most pressing issues is their low efficiency in scenes with complex visibility, where just a small fraction of emitted light contributes to the image, and often only after multiple interactions with surfaces. Such scenes are difficult especially for the popular photon density estimation and other bidirectional algorithms [GKDS12, HPJ12]. While those methods often excel at dealing with complex light transport, a good sampling strategy for subpaths originating from the light sources is critical to their success. Indeed, when the sampling of light subpaths is blind to the camera location, most calculation effort may end up being wasted on processing paths that make no image contribution [HJ11, VKŠ+14].

Our work focuses on improving light subpath sampling for photon density estimation. Specifically, we use Metropolis sampling to guide light subpaths toward the camera. While some previous work has taken this route [FCL+05, CWY11, HJ11], the existing algorithms usually result in a highly non-uniform distribution of the generated paths vertices, or "photons", and in turn also of the image error. This is undesirable since a few high-error regions force the rendering calculation to continue for a long time.

To address this problem, we derive an *importance function* (or, target function) for the Metropolis sampler that provably equalizes the *relative* error over a number of radiance measurements. By using our importance function for Metropolis light subpath sampling, the generated path vertices cover the regions of interest more evenly and, in turn, yield a uniform relative error distribution. Equalizing relative, as opposed to absolute, error is desirable when the estimates are directly displayed, because the human visual system is sensitive to relative differences in luminance.

We develop a hierarchical scheme for progressive construction of the importance function from paths generated during rendering. Unlike in previous work, where an importance function for Metropolis Light Transport was defined in image space [Vea97, HH10], we define ours over scene surfaces. This allows us to better exploit illumination coherence for a robust estimation of the importance function while adapting to geometric discontinuities. To sample from this importance function, we develop a replica exchange Metropolis sampler following multiple Markov chains. We show that our light subpath sampling algorithm produces a significant improvement in image quality when applied in progressive photon mapping [HJ09]. Our main contributions are:

- A derivation of an importance function that provably ensures equal distribution of relative error among a number of Monte Carlo estimators that share a common set of samples.
- Application of this importance function to light subpath tracing in photon density estimation.
- A scheme for estimating the importance function progressively during rendering based on a spatial hierarchy.

1 Related Work

We review light transport simulation algorithms designed to handle scenes with difficult visibility, with a focus on approaches based on Metropolis sampling.

Metropolis Light Transport (MLT). MLT [VG97] was the first use of the Metropolis-Hastings (MH) algorithm [Has70] in light transport simulation. A major advantage of the MH algorithm is its ability to generate samples (light paths) proportionately to an arbitrary scalar *importance function* (a.k.a. target function), which may include the – otherwise difficult-to-sample – path visibility. That is to say, it never produces paths blocked by geometry as valid samples. This unique feature makes MLT and related methods well suited to rendering occluded scenes. Yet, Vorba et al. [VKŠ+14] have demonstrated some important limitations of the current MLT-based solutions.

Veach and Guibas [VG97] use as the importance function the luminance value of the pixel contribution of a full transport path. A well-known problem of this importance function is a low number of paths contributing to dim image regions, resulting in their large relative error. Veach addressed this issue with his two-stage MLT [Vea97] where a low-resolution image is first rendered and the inverse of the pixel values is then used to rescale the original importance function for the actual rendering. The idea is that if all pixels receive the same expected number of samples, relative error will be equalized. We provide a formal justification and a generalization of this idea suitable to our photon tracing approach.

Hoberock and Hart [HH10] point out some important deficiencies of two-stage MLT that may, in fact, deteriorate the image quality compared to plain MLT. They propose a multi-stage MLT algorithm, where an estimate of the rendered image is continually refined during rendering and used to progressively update the importance function. Our work is based on a similar idea but we define our importance function over scene surfaces so that we can better benefit from illumination coherence.

In gradient-domain MLT [LKL+13], gradient of path contributions is used as the importance function so that image discontinuities are better explored than flat areas. Multiplexed MLT [HKD14] uses an importance function given by the path contribution modulated by the multiple importance sampling weight [Vea97] of the technique used to generate the path. That way, appropriate techniques are selected more often.

The Metropolis-Hastings algorithm generates a Markov chain of samples (in our case paths), where a new sample is generated by a *mutation* of the current one. Intricate path mutation strategies have been devised to help the MLT algorithm deal with complex light transport, such as specular-diffuse-specular interactions. But even the most advanced mutation strategies known to date [JM12, KHD14] may still fail to converge to the desired result in an acceptable time [VKŠ+14].

Metropolis sampling in photon density estimation. Unlike MLT and the derived algorithms, photon density estimation handles complex specular-diffuse-specular transport gracefully thanks to its inherent subpath reuse and regularization properties [KD13b]. In combination with other bidirectional path sampling strategies, it makes for algorithms robust to various lighting and material settings [GKDS12, HPJ12, KGH+14]. It is for this reason that we choose photon density estimation as the basis of our approach.

As discussed in the introduction, a good sampling strategy for subpaths originating from light sources is critical to a good performance of photon density estimation, and ours is not the first work to apply Metropolis-Hastings sampling to generate those light subpaths. Fan et al. [FCL+05] runs the original MLT algorithm and uses selected vertices of the sampled paths as photons. More closely related are the works of Hachisuka and Jensen [HJ11] and Chen et al. [CWY11]. Both algorithms are designed for use in stochastic progressive photon mapping [HJ09] and their purpose is to guide the light subpaths toward a set of measurement points distributed on scene surfaces. Our work follows this general scheme. The two methods differ mostly by the importance function for the Metropolis-Hastings sampler. Hachisuka and Jensen use the path visibility, that is, a binary variable indicating whether or not the path contributes to any of the measurement points. Chen et al. modulate the visibility by a somewhat arbitrary function constructed from local photon density estimated in a pilot photon tracing pass. A major shortcoming of those methods is their tendency to generate paths that lead to a highly varying error of radiance estimates at the measurement points. Our proposed importance function also incorporates the path visibility but we modulate it to ensure a uniform relative error distribution.

Hachisuka and Jensen [HJ11] also introduce some important optimizations of the Metropolis-Hastings sampler itself, such as adaptive mutation size [AT08] and replica exchange [Nea96, KKK09] that we adopt and extend in our work.

Local path sampling. Metropolis sampling is not the only approach to guide light subpaths toward the camera. Another option is to devise suitable local sampling pdfs for constructing light subpaths vertex-by-vertex. Those pdfs can be constructed adaptively based on the observed contributions of previously generated paths [**?**], or using directional density estimation from importance particles distributed in the scene in a preprocessing phase [PP98, BRDC12]. We compare our results to a recent work from the latter category [VKŠ+14] in Section 5.

2 Overview

The objective of our work is to develop a photon tracing algorithm that produces uniform *relative* error of radiance estimates on a set of measurement points distributed on scene surfaces. This goal is motivated by the sensitivity of the human visual system to relative, rather than absolute, luminance deviations. We build on the works of Hachisuka and Jensen [HJ11] and Chen et al. [CWY11], where the Metropolis-Hastings algorithm generates samples in the primary-sample space$\mathcal{U}$ [KSKAC02], which are then transformed into the path space Ω to produce actual light subpaths. We achieve the desired uniform error distribution by importance sampling in the primary-sample spaceusing a suitably defined scalar *importance function* $\hat{I} : \mathcal{U} \to \mathbb{R}$. Our contributions consist in providing a formal derivation of the importance function (Sec. 3) and developing a practical and robust algorithm for estimating and sampling from this importance function in the course of rendering (Sec. 4). The rest of this section provides an overview of our approach.

Our importance function (IF) is given by the inverse of the expected number of photons contributing to each measurement point. Such an IF is unknown at the outset and computing it is as hard as rendering the image itself. For this reason, we progressively refine estimates of the IF during the rendering process. This is done by collecting and maintaining suitable statistics from the generated path vertices (photons) over a set of spatial regions.

The estimated IF is subject to variance, especially in early stages of calculation. To obtain robust estimates despite this variance, we initially average the statistics used to calculate the IF over large regions. The regions are refined as the calculation progresses and the variance of their statistics decreases. This process progressively improves our approximation of the ideal IF.

A distinguishing feature of our approach is that the IF is defined over the *spatial* regions organized in a spatial hierarchy. This allows us to take advantage of illumination coherence in the regions while adapting to geometric discontinuities. We show that in our setting this is an important improvement over calculating the importance function in the image plane [Vea97, HH10].

Excessive sample correlation could result from using the basic Metropolis algorithm to sample from our IF because of its multimodal shape and relatively high dynamic range. To ensure good exploration without "getting trapped" in the IF modes, we design a replica-exchange Metropolis algorithm following multiple parallel Markov chains.

We apply our photon tracing algorithm in stochastic progressive photon mapping [HJ09] and we show that it outperforms existing photon tracing algorithms in terms of image quality by a large margin (Sec. 5).

3 Importance Function

This section discusses the importance function itself; a rendering algorithm based on this function is then described in Sec. 4.

Problem statement. Our goal is to calculate a number of *radiance measurements*, that is, integrals of the form

$$R_k = \int_\Omega h_k(\bar{\mathbf{x}}) f(\bar{\mathbf{x}}) \mathrm{d}\mu(\bar{\mathbf{x}}), \tag{7.1}$$

where $\bar{\mathbf{x}} \in \Omega$ is a light transport path and $h_k(\bar{\mathbf{x}})f(\bar{\mathbf{x}})$ is the measurement contribution of that path. The function $f(\bar{\mathbf{x}})$, common to all the measurements, is given by the product of the emitted radiance at the first path vertex, scattering terms at the interior vertices, and geometry and visibility terms for the path edges [Vea97]. The function $h_k(\bar{\mathbf{x}})$, which is specific to each measurement, specifies the kind of measurement taken.

We consider photon tracing in the context of stochastic progressive photon mapping [HJ09], so the measurements R_k are radiance estimates at a number of *measurement points* G_k on scene surfaces. These are created by tracing paths from the camera until a sufficiently diffuse surface is encountered, where a measurement point is then deposited. In this setting, the functions $h_k(\bar{\mathbf{x}})$ are given by the product of the density estimation kernel and the BRDF at the measurement points.

A simplifying assumption we make for the derivation below is disregarding the BRDF dependence: we assume that $h_k(\bar{\mathbf{x}})$ is only the density estimation kernel. As a result, our importance function does not adapt to the BRDF and measurement points on glossy surfaces will have higher error than those on diffuse ones. Generalizing our derivation to directionally dependent BRDFs is left for future work.

With this formulation, our goal is to construct estimators $\langle R_k \rangle$ of the measurements, so that their relative error, that is, their *normalized standard deviation* (NSD) $\mathrm{nsd}(\langle R_k \rangle) = \sqrt{\mathrm{var}(\langle R_k \rangle)/R_k^2}$, is the same for all k. We achieve this by importance sampling in the primary-sample spaceusing a suitable importance function.

Light transport in primary-sample space. We use Metropolis-Hastings sampling in the *primary-sample space* $\mathcal{U} = \sum_{i=1}^{\infty}[0,1)^i$ [KSKAC02], where the measurements R_k are given as

$$R_k = \int_{\mathcal{U}} \hat{h}_k(\bar{\mathbf{u}}) \hat{f}(\bar{\mathbf{u}}) \left| \frac{\mathrm{d}\mu(\bar{\mathbf{x}})}{\mathrm{d}\bar{\mathbf{u}}} \right| \mathrm{d}\bar{\mathbf{u}} = \int_{\mathcal{U}} \frac{\hat{h}_k(\bar{\mathbf{u}}) \hat{f}(\bar{\mathbf{u}})}{\hat{p}(\bar{\mathbf{u}})} \mathrm{d}\bar{\mathbf{u}}. \tag{7.2}$$

The mapping $\bar{\mathbf{x}} = P^{-1}(\bar{\mathbf{u}})$, $\bar{\mathbf{u}} \in \mathcal{U}$ from the primary-sample spaceto the path space is given by the inverse cdfof the probability distribution used for importance sampling on the path space. The corresponding path pdf$p$ is the product of local pdfs for light emission sampling, BRDF sampling, and Russian roulette, used to generate the path vertices from a vector of "random numbers" $\bar{\mathbf{u}}$. Here we have introduced the notation $\hat{h}_k(\bar{\mathbf{u}}) = h_k(P^{-1}(\bar{\mathbf{u}}))$, and similarly for $\hat{f}$ and $\hat{p}$. We use the same convention also for other functions, notably the importance function $\hat{I}$.

Equalizing relative error. Veach [Vea97] argues that equal relative error of pixel measurements in MLT can be achieved by making the expected number of contributions to each measurement the same. We provide a more formal derivation and a generalization of this result below. We derive an importance function in the

primary-sample space, $\hat{I} : \mathcal{U} \to \mathbb{R}$, that ensures uniform relative error in a more general setting, and we show that this goal is achieved by equalizing the number of contributions only under a specific set of assumptions.

Our importance function can be intuitively understood as compensating for the non-uniform mapping between the primary-sample spaceand the paths space, and also for the non-uniform size of the density estimation kernels at the measurement points. The varying kernel size stems from the usual practice where it is determined by the projected pixel area.

Importance function derivation. Going back to Eq. 7.2, all the measurements R_k can be estimated simultaneously using Monte Carlo quadrature with a single, shared set of samples, in our case light subpaths. To ensure uniform relative error across the measurements, we employ importance sampling in the primary-sample spaceby using random variables U from the pdf$\hat{q}(\bar{\mathbf{u}}) = \hat{I}(\bar{\mathbf{u}})/b$ with $b = \int \hat{I}(\bar{\mathbf{u}})\mathrm{d}\bar{\mathbf{u}}$. That is, the pdf$\hat{q}$ is a normalized version of the importance function $\hat{I}$ that we seek to derive. This yields a classic one-sample Monte Carlo estimator of the measurements R_k:

$$\langle R_k \rangle = \frac{\hat{h}_k(U)\hat{f}(U)/\hat{p}(U)}{\hat{q}(U)} \tag{7.3}$$

We now derive the pdf$\hat{q}$ such that the relative error given by $\mathrm{nsd}(\langle R_k \rangle) = \sqrt{\mathrm{var}(\langle R_k \rangle)/R_k^2}$ is the same for estimators of all the measurements R_k. To simplify the derivation, let us assume that the functions $\hat{h}_k$ have disjoint supports (i.e. they are non-zero at different parts of the domain), and that the pdf$\hat{q}$ is constant in the support of the individual $\hat{h}_k$'s, i.e. $\hat{q}(\bar{\mathbf{u}}) = \hat{q}_k$ for $\bar{\mathbf{u}} \in \mathrm{supp}(\hat{h}_k)$. The variance of $\langle R_k \rangle$ is given by $\mathrm{var}(\langle R_k \rangle) = E[\langle R_k \rangle^2] - R_k^2$, and under the above assumptions, the second moment becomes

$$E[\langle R_k \rangle^2] = \frac{1}{\hat{q}_k} \underbrace{\int_{\mathrm{supp}(\hat{h}_k)} \left[\hat{h}_k(\bar{\mathbf{u}})\hat{f}(\bar{\mathbf{u}})/\hat{p}(\bar{\mathbf{u}})\right]^2 \mathrm{d}\bar{\mathbf{u}}}_{E[\langle R_k \rangle^2_{\mathrm{uni}}]}. \tag{7.4}$$

That is, $E[\langle R_k \rangle^2]$ is equal to a rescaled version of $E[\langle R_k \rangle^2_{\mathrm{uni}}]$, the second moment of the MC estimator that uses uniform sampling on the primary-sample space. The NSD can now be written as

$$\mathrm{nsd}(\langle R_k \rangle) = \sqrt{\frac{1}{\hat{q}_k} \frac{E[\langle R_k \rangle^2_{\mathrm{uni}}]}{R_k^2} - 1}. \tag{7.5}$$

This equation implies that to make the NSD of all the estimators equal, it is sufficient to set

$$\boxed{\hat{q}_k = \frac{1}{b} \frac{E[\langle R_k \rangle^2_{\mathrm{uni}}]}{R_k^2} = \frac{\hat{I}_k}{b}} \tag{7.6}$$

Recall that the normalization constant b ensures that $\hat{q}$ integrates to one as a whole. With this definition of $\hat{q}_k$, we have $\mathrm{nsd}(\langle R_k \rangle) = \sqrt{b-1}$, so the NSD is indeed equal for all k.

Let us summarize this result. Under the assumption that individual integrands in Eq. 7.2 do not overlap (i.e. have disjoint support), relative error can be evenly distributed among estimators of the individual measurements, Eq. 7.3, by using importance sampling in the primary-sample spacewith an importance function I that is constant in the support of the individual integrals and is given by Eq. 7.6.

A similar derivation with an equivalent result can be carried out directly in the path space, without resorting to the primary-sample space. We choose the above derivation because it allows to clearly tease apart importance sampling in the primary-sample space(the pdf$\hat{q} = \hat{I}/b$ proportional to the importance function $\hat{I}$) from importance sampling in the path space (the pdf$p$).

Application to photon tracing. We now use the general result in Eq. 7.6 to write the importance function for sampling light subpaths in progressive photon mapping. Recall that the functions h_k are the density estimation kernels at the measurement points. We assume piece-wise constant kernels, i.e., $h_k(\bar{\mathbf{u}}) = 1/s_k$, with $s_k = \pi r_k^2$, for $\bar{\mathbf{u}}$ such that the last vertex of the corresponding path $\bar{\mathbf{x}} = P^{-1}(\bar{\mathbf{u}})$ is within the radius r_k from the measurement point G_k, and $h_k(\bar{\mathbf{u}}) = 0$ otherwise. Furthermore, with good path space importance sampling we can assume that the pdf$p$ in Eq. 7.2 is approximately proportional to the contribution function, i.e. $\hat{p}(\bar{\mathbf{u}}) \approx \hat{f}(\bar{\mathbf{u}})/c$. With those assumptions, the second moment $E[\langle R_k \rangle^2_{\text{uni}}]$ given by Eq. 7.4 becomes equal to $c^2/s_k^2 \int_{\text{supp}(\hat{h}_k)} \mathrm{d}\bar{\mathbf{u}}$ and $R_k^2 = (c/s_k \int_{\text{supp}(\hat{h}_k)} \mathrm{d}\bar{\mathbf{u}})^2$. Eq. 7.6 then simplifies to

$$\hat{I}_k = \frac{1}{\mathbb{P}^{\text{uni}}_k} \qquad \text{with} \qquad \mathbb{P}^{\text{uni}}_k = \int_{\text{supp}(\hat{h}_k)} \mathrm{d}\bar{\mathbf{u}} \tag{7.7}$$

By writing the above integral for $\mathbb{P}^{\text{uni}}_k$ directly in the path space, we get $\mathbb{P}^{\text{uni}}_k = \int_{\text{supp}(h_k)} p(\bar{\mathbf{x}})\, \mathrm{d}\mu(\bar{\mathbf{x}})$. This means that $\mathbb{P}^{\text{uni}}_k$ is the total probability that a path generated by *uniform sampling in the primary-sample space* will make a non-zero contribution to the estimate at the point G_k. The net result of sampling in the primary-sample spaceproportionately to an importance function given by the inverse of $\mathbb{P}^{\text{uni}}_k$ (Eq. 7.7) is that *all measurement points get an equal probability of receiving a non-zero contribution*. In other words, relative error is equalized by equalizing the number of contributions.

Discussion. The above result is in line with the argument that Veach [Vea97] made in the context of MLT. While our derivation might appear as a lengthy way of arriving at this result, its value lies in clearly identifying all the assumptions, while also deriving the more general result in Eq. 7.6. Let us point out that without assuming "ideal" importance sampling in the path space (i.e. when $\hat{p}(\bar{\mathbf{u}}) \neq \hat{f}(\bar{\mathbf{u}})/c$), no direct relation between the relative error and number of contributions exist. Eq. 7.6 gives the importance function in such a case. Nonetheless, in our algorithm, described in the next section, we rely on the importance function given by Eq. 7.7.

4 Algorithm

In this section, we develop a practical algorithm for estimating and sampling from our importance function (IF) in the course of rendering. The algorithm works in iterations and refines the IF estimates as it progresses. Iteration i uses the IF $\hat{I}_i$ for path sampling, while collecting statistics that will be used to calculate an updated IF $\hat{I}_{i+1}$ for the next iteration. In this way, the importance function approaches the ideal IF given by Eq. 7.7.

To calculate the IF according to Eq. 7.7, we need to estimate the probability $\mathbb{P}_k^{\text{uni}}$ for each measurement point G_k. The estimates of $\mathbb{P}_k^{\text{uni}}$ need to be carried over from one iteration to another so that their variance can eventually vanish. This is complicated by the fact that each iteration uses a new, independently generated set of measurement points. Stochastic progressive photon mapping [HJ09] solves this by maintaining statistics associated with image pixels. We adopt the idea of statistics maintained over the iterations. A major departure of our approach consists in associating those statistics to *spatial regions* C, as opposed to pixels. We refer to an importance function calculated using the statistics from the spatial regions as a *spatial importance function*. The spatial IF is more robust to illumination changes caused by geometry discontinuities because it averages statistics over compact spatial regions, where illumination can be expected to show coherence.

4.1 Importance function calculation

To calculate the importance function, we associate with each spatial region C a statistic $\kappa(C)$ that persists over the iterations and is continually updated. This statistic maintains a running estimate of the probability $\mathbb{P}_k^{\text{uni}}$ averaged over the measurement points G_k that have so far been generated in this region.

Let us start with the estimation of the probability $\mathbb{P}_k^{\text{uni}} = \int_{\text{supp}(\hat{h}_k)} \mathrm{d}\bar{\mathbf{u}}$ (Eq. 7.7) for a single measurement point. We use the paths generated in iteration i to evaluate a statistic $\psi_i(G_k)$ for each measurement point G_k. This statistic serves as a Monte Carlo estimate of the probability $\mathbb{P}_k^{\text{uni}}$. If we used uniform sampling in the primary-sample space, this could be calculated as a fraction of paths that make a non-zero contribution to G_k. However, this would be a bad estimator because such paths are extremely rare in scenes that we consider. Luckily, we can obtain a more accurate estimate by using paths sampled by the Metropolis sampler from our current estimate of the importance function $\hat{I}_i$. Using those paths, the statistic $\psi_i(G_k)$ is calculated as

$$\psi_i(G_k) = \frac{1}{N} \sum_{\bar{\mathbf{u}}_j \in \mathcal{S}} \frac{1}{\hat{I}_i(\bar{\mathbf{u}}_j)/b_i} \tag{7.8}$$

where N is the number of paths sampled in each iteration and the sum runs over paths that fall into the support of the kernel at G_k, i.e. $\mathcal{S} = \{\bar{\mathbf{u}}_j \mid P^{-1}(\bar{\mathbf{u}}_j) \in \text{supp}(h_k)\}$. The normalization by $b_i = \int I_i(\bar{\mathbf{u}})\bar{\mathbf{u}}$ is carried out at the end of the iteration because b_i is not known earlier.

To estimate the overall probability $\mathbb{P}^{\text{uni}}$ for a region C, we sum $\psi_i(G_k)$ over all

measurement points contained by that region:

$$\psi_i(C) = \sum_{G_k \in C} \psi_i(G_k). \tag{7.9}$$

Finally, the value of the κ statistic for the next iteration is calculated

$$\kappa_{i+1}(C) = [\kappa_i(C) + \psi_i(C)] \frac{s_{i+1}(C)}{s_i(C)}. \tag{7.10}$$

Here $s_i(C)$ is the sum of the kernel sizes s_k at the measurement points contained by the region C in iteration i:

$$s_i(C) = \sum_{G_k \in C} s_k. \tag{7.11}$$

Two components of the importance function. We could use the inverse of the κ statistic to define a piece-wise constant importance function over the entire space. However, this would be a rough estimate of the ideal IF. We show how to take advantage of the region statistics without forcing the IF to be constant in each of them. This provides a more accurate IF approximation.

We use the fact that the probability $\mathbb{P}_k^{\text{uni}}$ at any measurement point can be approximated as a product of local area density of path vertices $D^{\text{uni}}(G_k)$ (under uniform sampling in the primary-sample space) and the kernel size s_k. Since the kernel sizes s_k are known, they can be factored out in the IF calculation and provide the desired modulation within each region.

The local density $D^{\text{uni}}(G_k)$ at any measurement point is approximated by the average density $D^{\text{uni}}(C)$ over the spatial region containing G_k. At iteration i, the average density in a region can be calculated from the κ statistic of that region as

$$D_i^{\text{uni}}(C) = \frac{1}{N_i(C)} \frac{\kappa_i(C)}{s_i(C)}, \tag{7.12}$$

where $N_i(C)$ is the number of iterations in which the region C contained at least one measurement point.

Importance for a measurement point. We could now calculate the IF at any measurement point $\hat{I}_i(G_k)$ as the inverse of $D^{\text{uni}}(G_k)s_k$. However, both the local density $D^{\text{uni}}(G_k)$ and the kernel size s_k can have a high dynamic range. Using them directly could result in an IF that would be difficult to explore for the Metropolis algorithm. For this reason, we compress the dynamic range of both the inverse density and the inverse kernel size. The final formula for the IF at a measurement point G_k then reads:

$$\hat{I}_i(G_k) = \underbrace{\left[\epsilon_1 \frac{1}{\frac{D_i^{\text{uni}}(C(G_k))}{\max_l\{D_i^{\text{uni}}(C_l)\}} + \epsilon_1}\right]}_{\text{average density in region}} \cdot \underbrace{\left[\epsilon_2 \frac{1}{\frac{s_k}{\max_l\{s_l\}} + \epsilon_2}\right]}_{\text{kernel size at meas. point}} \tag{7.13}$$

Here we have used a shortcut $C(G_k)$ to denote the region that contains the measurement point G_k. Division by the maxima is used to normalize the range of the

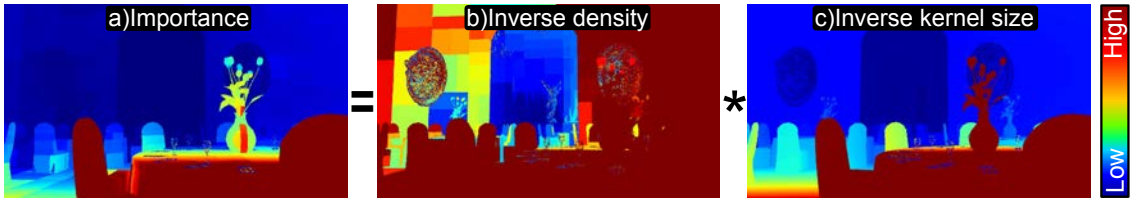


Figure 7.1 – The importance function $\hat{I}(G_k)$ for a measurement point (a) is given by the product of two components: the inverse of the average photon density in the spatial region that contains the measurement point (b), and the inverse of the density estimation kernel size at that point (c). The images show the importance for measurement points associated with the image pixels.

respective quantities: $\max_l\{D_i^{\text{uni}}(C_l)\}$ is the maximum of all the region densities and similarly, $\max_l\{s_l\}$ is the maximum kernel size over all measurement points. Both those maxima consider only regions and measurement points in the current iteration. The parameters ϵ_1 and ϵ_2 ensure that the range of the respective component of the importance function remains within the interval $[\epsilon/(1+\epsilon), 1]$, where $\epsilon \in \{\epsilon_1, \epsilon_2\}$. We use $\epsilon_1 = \epsilon_2 = 10^{-4}$ in all our results. Fig. 7.1 illustrates the two components of the importance function.

Importance for a full path. To compute the importance function $I_i(\bar{\mathbf{x}})$ of an entire light subpath $\bar{\mathbf{x}}$, we take the maximum of the IF values for the measurement points $\mathcal{K}$ impacted by the path vertices:

$$I_i(\bar{\mathbf{x}}) = \max_{k\in\mathcal{K}}\{\hat{I}_i(G_k)\}, \tag{7.14}$$

Taking the maximum is motivated by the preference for exploring difficult regions.

4.2 Spatial region definition and refinement

Statistics in small spatial regions could be subject to high variance, especially in the early stages of calculation. As a result, the estimated importance function could contain some artificial spikes that would negatively affect its ability to equalize error. We adopt a natural solution where variance in the early iterations is reduced by averaging the statistics over larger spatial regions. The regions are then refined as the calculation progresses.

To define the spatial regions C, we build a spatial hierarchy (a kD-tree in our case) prior to rendering, as shown in Fig. 7.2. Each node of this structure corresponds to one spatial region with its associated statistic $\kappa(C)$. The set of regions used to define the importance function $\hat{I}_i$ in iteration i is given by a cut through the tree.

Region refinement strategy. We refine the spatial regions C as the rendering progresses so as to improve the accuracy with which we model the ideal importance function. We base our region refinement policy on a simple argument about the expected variance of the regions' κ statistics. The goal is to avoid using regions with a high variance of their κ statistic, otherwise our IF would be inaccurate and possibly counterproductive.

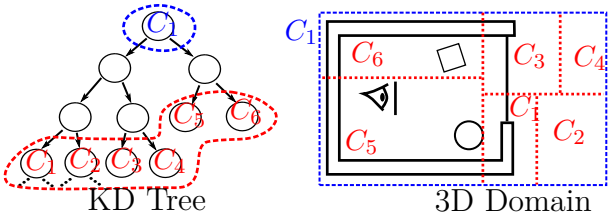


Figure 7.2 – The spatial regions used to maintain the statistics for importance function estimation are defined by the nodes of a spatial hierarchy constructed prior to rendering. For any given algorithm iteration, the current set of regions is given by a cut of the tree.

Since κ is given by a weighted sum of Monte Carlo estimators ψ_i (Eq. 7.8 and 7.9), its variance is inversely proportional to the number of photons that the corresponding region receives. We construct our hierarchy such that each node receives about half the photon count of its parent, and thus has about twice the parent's variance. As a result, to avoid increasing variance, the number of iterations for each next refinement must at least double. While this simple scheme worked well in our tests it could be improved by using an adaptive refinement strategy based on actual, rather than expected, variance estimates in the regions.

Hierarchy construction. As mentioned above, our goal is that a region associated with a hierarchy node receives about half the number of photons of its parent. To construct such a hierarchy, we exploit the fact that our importance function equalizes the number of photons received by the individual measurement points. The hierarchy construction starts by distributing a pilot batch of measurement points in the scene. We then construct the hierarchy in a top-down manner in such a way that after a split both children have about the same number of measurement points: in other words, we use the median-split rule.

Image-based importance function. So far, we have considered that a region is a part of the 3D space. We could define an image-based importance function by changing the notion of what a "region" means. For an image-based IF, a region is a part of the image plane: a measurement point belongs to that region if it was generated by tracing a path through a pixel in the region. The region hierarchy is then a regular kD-tree in the 2D image space. An image-based IF defined in this way is closely related to the multi-stage MLT algorithm of Hoberock and Hart [HH10]. We show in our results that our new spatial definition yields an importance function that better equalizes image error.

4.3 Algorithm Overview

Alg. 5 provides an overview of our algorithm. Only steps related to importance function calculation are shown. Steps necessary for image rendering are similar as in regular stochastic progressive photon mapping [HJ09].

Before the rendering, we build the hierarchy of regions (line 1). The algorithm then proceeds in iterations, each of which starts by distributing a new set of mea-

surement points by tracing camera subpaths through image pixels (line 4). After that, we calculate the average density $D_i^{\text{uni}}(C)$ for all regions (line 5), which allows us to evaluate the importance function for all the measurement points (line 6). We now run our Metropolis algorithm, described in Sec. 4.4, that samples photon paths using the importance function $\hat{I}_i$ while updating the statistic $\psi_i(G_k)$ for the measurement points (line 9). After the path sampling has finished, we gather the measurement point statistics and calculate the statistic $\kappa_{i+1}(C)$ for all regions (line 12). Note that the multiplication by $s_{i+1}(C)$ in Eq. 7.10 is carried out at the beginning of the next iteration, when this value is known. In the final step, we evaluate our refinement criterion and possibly subdivide the hierarchy (line 13).

Algorithm 5 Overview of our algorithm.

```
1:  ConstructRegionHierarchy()
2:  for i = 0 ... N_iter − 1 do
3:      // Step 1: Initialization
4:      DistributeMeasurementPts()
5:      UpdateRegionDensity()                    ▷ Eq. 7.12
6:      AssignImportanceToMP()                   ▷ Eq. 7.13
7:
8:      // Step 2: Photon tracing
9:      RunMetropolisPathSampling()              ▷ Eq. 7.8
10:
11:     // Step 3: Statistics update
12:     UpdateRegionStatistics()                 ▷ Eq. 7.10
13:     RefineClusters()
14: end for
```

4.4 Sampling form the importance function

We conclude the algorithm description by presenting the variant of the Metropolis algorithm that we use for sampling from our importance function. As we show in Fig. 7.1, the importance function has a relatively high dynamic range and could have several modes. Such functions are hard to explore via basic Metropolis sampling because the generated Markov chain tends to get trapped in the modes and produce highly correlated samples. To ensure good exploration, we employ a replica-exchange Metropolis algorithm [Nea96, KKK09, HJ11]. A general idea of replica-exchange Metropolis is to follow several Markov chains in parallel with a probabilistic exchange of states between any two chains. Using "tempered" importance functions for some of the chains ensures good exploration with low correlation even if one chain follows a function with a complex shape.

We build upon the work of Hachisuka and Jensen [HJ11] and we refer to that paper for more details. Our design features three chains. The first chain uses uniform, independent proposals and a constant importance function. This chain behaves in the same manner as in Hachisuka and Jensen's work. The second chain uses as its importance function the inverse of the kernel size (the second factor in

Eq. 7.13, see also Fig. 7.1). Finally, the third chain follows our complete importance function given by Eq. 7.13. Symmetric mutations in the primary-sample spacewith adaptive mutation size are used for chains two and three [HJ11].

There are two major advantages to using this three-chain design over a simpler one that only combines uniform sampling with sampling from our importance function. The first benefit, as already discussed, is less sample correlation. A no less important advantage is a more robust calculation of the normalization factor $b_i = \int I_i(\bar{\mathbf{u}})\bar{\mathbf{u}}$. While this integral can be easily estimated by using paths from the first chain, the estimate often suffers from a high variance because of the complex shape of our importance function. We instead calculate the normalization factor by using samples from one chain to estimate the normalization of the next chain [KKK09]. This approach provides a substantially more stable estimates.

Finally, our three-chain algorithm samples paths from several different distributions given by the respective target functions. To use all those samples while minimizing the variance of the result, we combine them using Multiple Importance Sampling [Vea97].

5 Results

In Figure 7.8 we compare the results of stochastic progressive photon mapping (SPPM) with three different methods for sampling photon paths: our method, Hachisuka and Jensen [HJ11] and the on-line learning of Vorba et al. [VKŠ+14]. We implemented the methods in the Mitsuba renderer [Jak10]. They share the same code for distributing and looking-up the measurement points, the photon search radius is initialized using ray differentials to approximately the projected pixel size, and the radius reduction parameter is set to 1. We emit 10 million photons paths per iteration for all the methods. All the scenes were rendered for one hour at a resolution 960×540 on 2 x Intel(R) Xeon(R) CPU E5640 @ 2.67GHz using 8 logical threads.

Since we use photon mapping to render all images, we left out all the purely specular paths. These paths can be easily rendered by brute-force Monte Carlo path-tracing.

In the **Dinner hall** scene in Figure 7.8 only the room that is far from the camera is strongly illuminated. This pose a problem for existing algorithms as they do not distribute light samples efficiently between bright and visually important dim parts of the scene. Our method clearly outperforms the state-of-the-art algorithms.

The **Villa interior** scene is illuminated by a strong sunlight, however, parts of the scene are very dark thanks to the complex visibility of the scene. Both our competitors succeed in removing noise from brightly illuminated areas, but retain lots of error in the dimmer parts of the image. Our algorithm on the other hand ensures that image error is reduced equally in the whole image as can be clearly seen from the normalized standard deviation (NSD) graph.

Finally, in the **Canyon** scene the light arrives from only small set of directions through holes between the rocks. Although the light doesn't have too travel through glass as in the two previous scenes, the visibility remains very difficult. Our method

delivers almost noise-free image, which can't be told about our competitors.

Relative error and number of contributions. The objective of our method is to equalize relative error. The top row of Figure 7.3 shows that our method indeed achieves this goal. We can see that unlike in the methods of Hachisuka and Jensen [HJ11] and Vorba et al. [VKŠ+14], our relative error images are almost constant. In the bottom row, we confirm that the relative error is proportional to the number of contributions received by the measurement points.

Algorithm components. We now illustrate the impact of the various components of our approach on the image quality. We start by demonstrating in Figure 7.4 the advantage of our spatial definition of the importance function over a simple image-based approach. Figure 7.5 demonstrates that without our 3-chain replica exchange approach, the algorithm is not able to explore the state space well and produces outlier pixel values due to the Markov chain "getting stuck". In addition, the normalization factor in the simple 2-chain approach is noisy, which can lead to visible brightness difference in the images. Finally, Figure 7.6 shows how combining samples from the different Markov chains using MIS reduces noise compared to a simple averaging.

Performance in simple scenes. Advanced photon tracing methods such as ours are not needed in simple scenes where most of the paths generated by uniform sampling in primary-sample spacecontribute to the image. While our method has some computational overhead and slightly increased variance due to sample correlation, we shown in Figure 7.7 shows that even in those cases, our method does not significantly impair performance compared to plain SPPM.

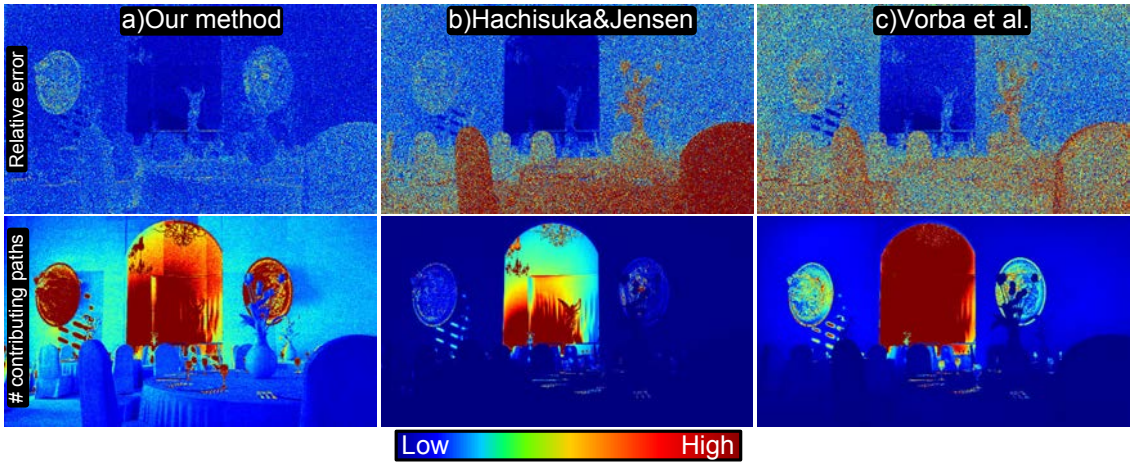


Figure 7.3 – Top row: Our method produces uniform distribution of relative error across the image plane (a). Error distribution is highly non-uniform in the images generated with the methods of Hachisuka and Jensen [2011] (b) and Vorba et al. [2014] (c). Bottom row shows the average number of contributions received by the measurement points. We see that the relative error is indeed directly proportional to this quantity.

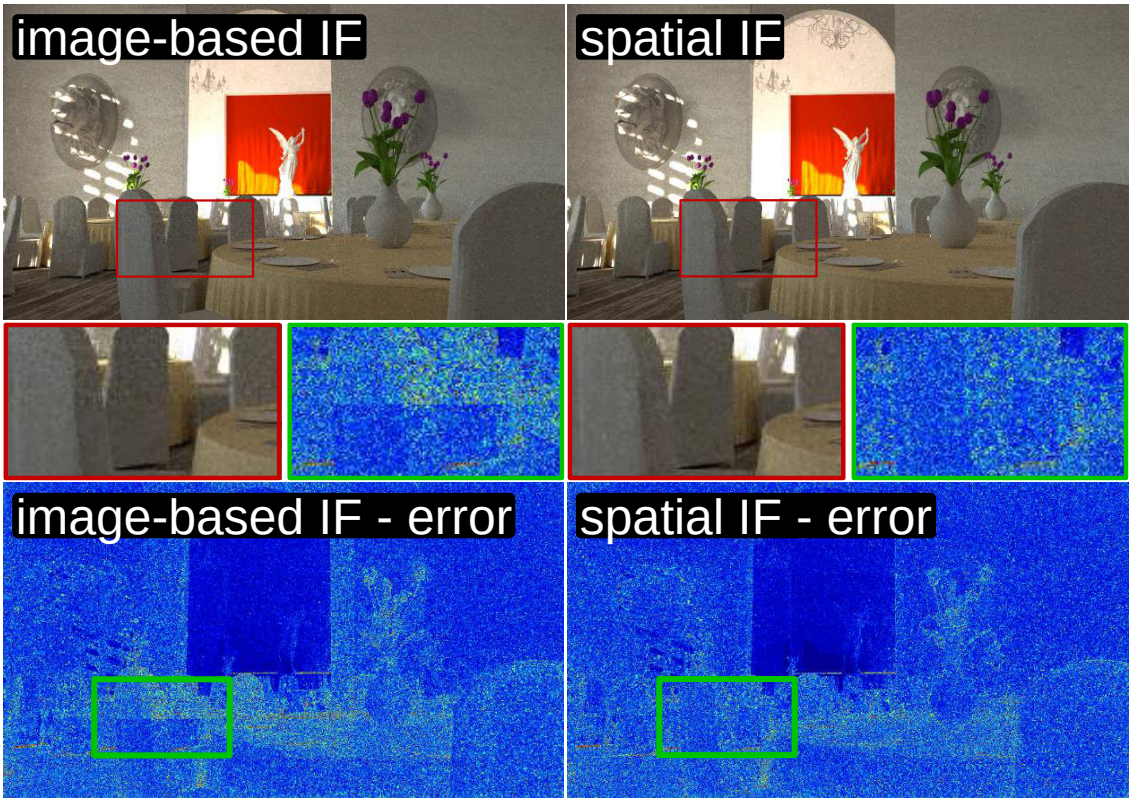


Figure 7.4 – Comparison of our method utilizing imaged based importance function (left) and spatial based importance function (right). The bottom row shows relative error of the upper row images. Using spatial based importance function removes rectangular artifacts, which are present when imaged based importance function is used.

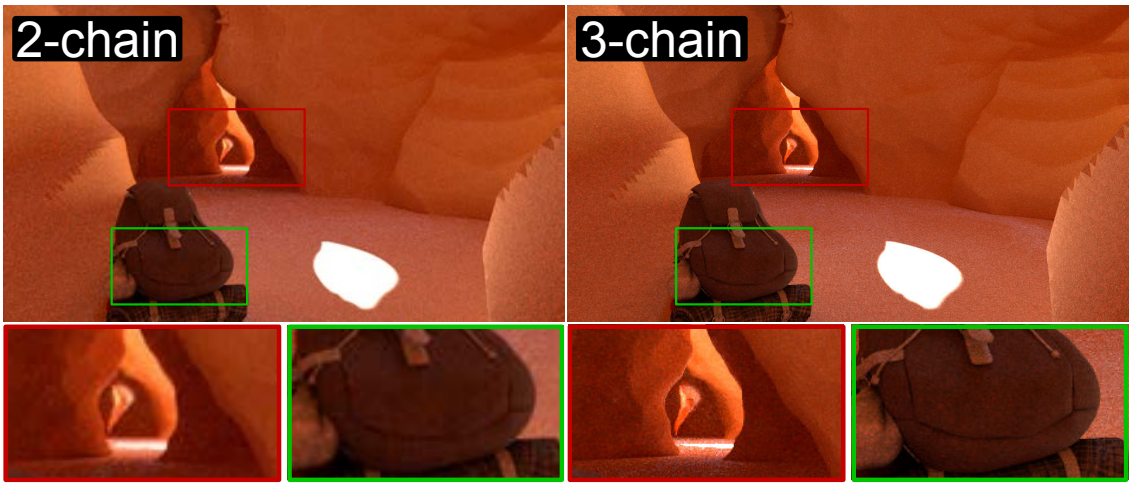


Figure 7.5 – Comparison of our method using only two Markov chains (left) and using all three Markov chains (right). Adding one extra Markov chain into our algorithm limits the possibility of "getting stuck" in one state. This helps to remove both fireflies and noise from the image.

Figure 7.6 – Comparison of our method without (left) and with (right) utilization of multiple importance sampling (MIS). MIS is used to weigh the different Markov chains in our 3-chain algorithm. Using MIS helps to reduce noise and bright spots in the generated images.

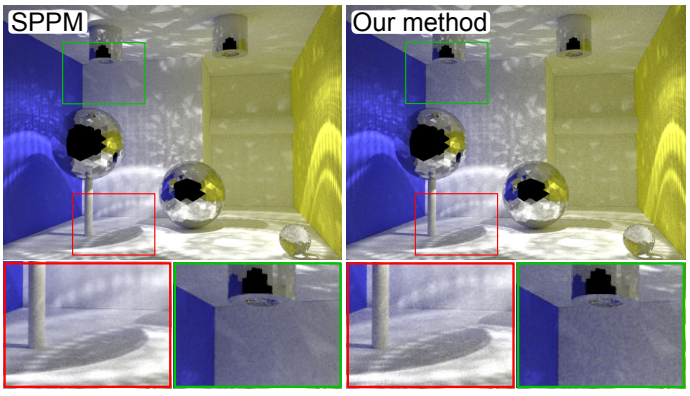


Figure 7.7 – Our method does not significantly impair the performance of plain SPPM in simple scenes, where most of the paths generated by uniform sampling in the primary-sample spacecontribute to the image. To generate those images, we ran SPPM and our method for 5 minutes.

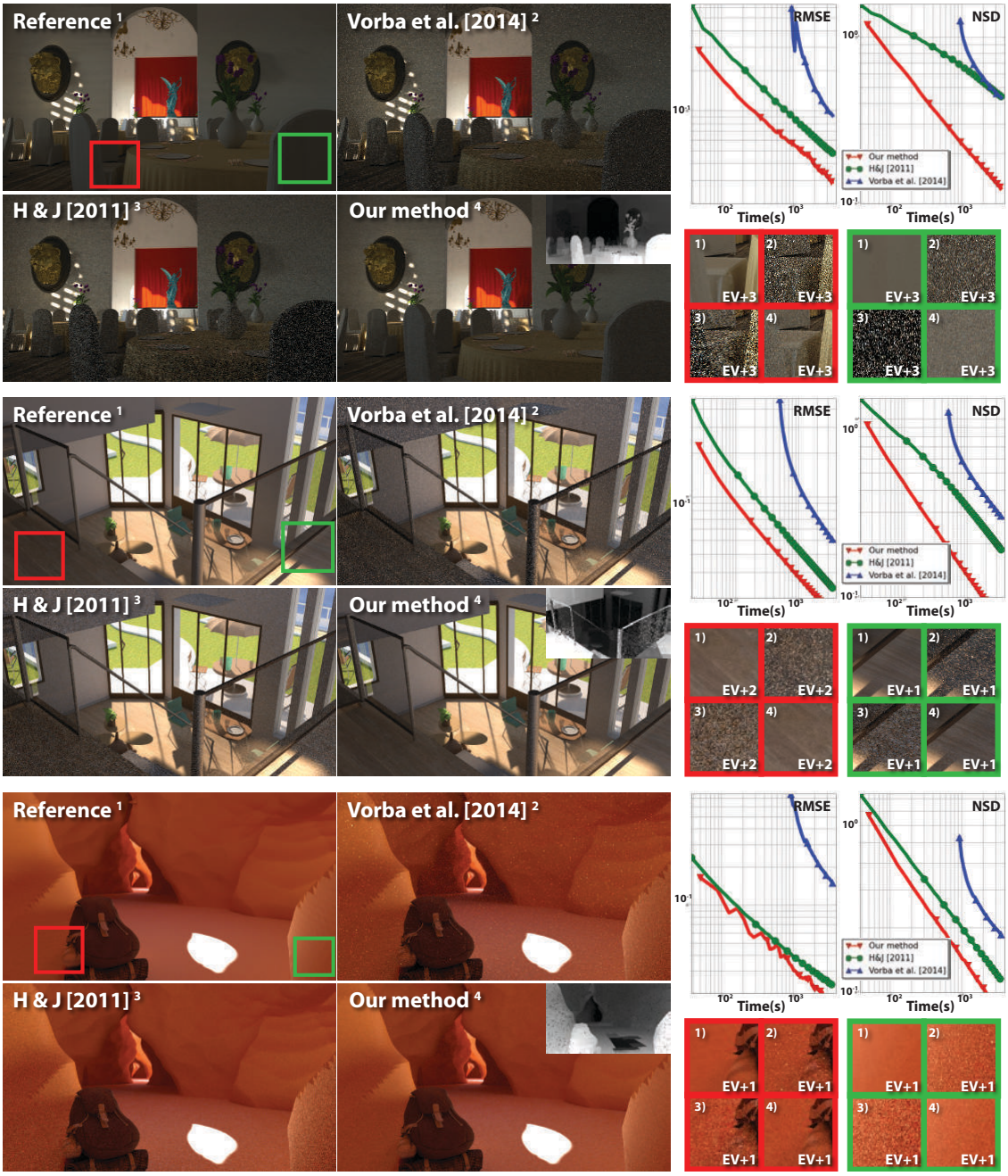


Figure 7.8 – In this figure we compare three different approaches to photon mapping including our method, method of Hachisuka and Jensen [HJ11] (b) and Vorba et al. [VKŠ+14] (c). We provide equal time comparison (1 hour). To make the differences between the algorithms more visible we change the brightness of the insets positioned on the left side (the exact change in exposure is shown inside each inset). For each image we also show its root mean square error (RMSE) and normalized standard deviation (NSD) over time in log-log plots. We also display spatial importance function associated with our method (displayed at upper left corner of each image generated by our algorithm).

6 Limitations and Discussion

In our current implementation, statistics from differently illuminated parts of the scene, such as interior and exterior, can be averaged in one spatial region. This could negatively affect the ability of the resulting importance function to equalize error. Considering surface orientation or using a spatial subdivision that better aligns with the scene geometry to define the regions would alleviate this problem.

We currently calculate an independent estimate of the normalization factor in each rendering iteration, which can only be done by sampling a relatively high number of photon paths per iteration. Devising a truly progressive normalization factor estimate would enable running a higher number of "smaller" iterations, thus improving sampling of effects on camera subpaths such as glossy reflections.

The objective of equalizing relative error is well suited when the individual measurements are directly related to the resulting pixel values, as in progressive photon mapping. However, this criterion is certainly not optimal when the light subpaths are used in a more complex way, such as in combined bidirectional algorithms [GKDS12, HPJ12]. It would be interesting to investigate suitable sampling distributions that would equalize relative pixel error in such algorithms.

We have focused solely on the primary-sample space importance function. But Vorba et al. [VKŠ+14] have shown that the path space importance sampling itself (i.e. the pdf$\hat{p}$ in Eq. 7.2) can be altered to guide more paths toward the camera. It would be interesting to see if a combination of the two approaches could further improve the results. Note that in this case, the photon "flux" (i.e. the ratio $\hat{f}/\hat{p}$ in Eq. 7.2) would not be roughly equal and, as a result, equalizing the number of contributions to the measurements would no longer produce uniform error. Instead, our general result in Eq. 7.6 would be a suitable basis for calculating the primary-sample space importance function.

7 Conclusions and Future Work

We have presented a spatial importance function for Metropolis photon tracing along with a robust scheme for calculating the function during rendering. Its objective is to equalize the relative error of a number of radiance measurements estimated using a shared set of photon paths. We have identified the assumptions under which this goal can be achieved by making the number of contributions to each measurement equal. But our derivation also exposes a more general form of the importance function with potential application in advanced importance sampling schemes.

Additionally, we have proposed an approach for progressive refinement of the importance function estimates in the course of rendering. Our spatial, as opposed to image-based, definition of the importance function provides an important advantage in this calculation. Finally, we developed a replica-exchange Metropolis algorithm that can sample from the importance function without excessive sample correlation. With this approach we have been able to demonstrate good results in scenes with difficult visibility and high luminance range, that are difficult for the existing photon

tracing methods.

Acknowledgments

The Mirror Ballsscene is courtesy of Toshiya Hachisuka and the Doorscene is courtesy of Jaakko Lehtinen. The work was supported by Charles University in Prague, project GA UK No 1362413, by the grant SVV2014260103, and by the Czech Science Foundation grant P202-13-26189S.

At the early stage of this project, the idea has been submitted to a french workshop: A. Gruson, R. Ribardiere, R. Cozot and K. Bouatouch, "Rendu Progressif basé Metropolis-Hasting dans des scènes à contextes topologiques multiples", REFIG, 2014. However, this is different compared to the technique presented in this chapter. First, the spatial subdivision was manually performed (resulting in a set of 3D cells, each cell is called spatial context). Second, we used visibility-based importance function similarly to Hachisuka et al [HJ11]. Each spatial context is assigned a different "local" importance function with its own Markov chain. This way, different AMCMC [AT08] processes are used to learn optimal mutation sizes. However, this approach suffers from noise appearing at the borders between spatial contexts. For this reason, we left out this approach and proposed the method presented in this chapter.

Part III

Computer-aided global illumination techniques for artists

Introduction

In the previous part of the manuscript, we have presented some rendering techniques we have designed and implemented. Remember that the output of a rendering technique is an image with a certain aesthetics. The aim of the artist is to ensure that the desired aesthetics matches his/her intent. To change the image aesthetics, the artist must:

- change the 3D scene (light position, materials used, etc.) and recompute the image.
- or apply, in a post-production step, image-based transformations such as color transformation, color transfer, blur addition, compositing, etc. For example, in fig. 7.9, the final output has been modified through a style transfer.

In this part, we will focus on some techniques used by the artist so that the final image matches his/her intent.

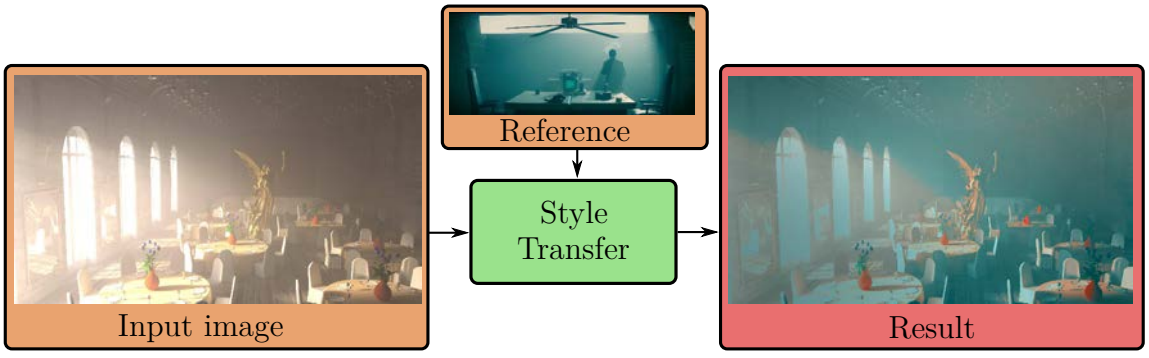


Figure 7.9 – Example of style transfer technique. The input image is an image generated using a rendering algorithm. The artist provides a reference image that matches his/her intent. The algorithm takes these two inputs and transfers the style of the reference image to the input image (color transfer). To perform this color transfer, additional information is needed such as the main illuminant color [NKB14].

First, we will present a new technique to estimate the main illuminant color for a 3D scene (chapter 8). This illuminant can be that of the light source color or the indirect lighting. It can be used during a post-processing step to apply color transformations (white balancing, style transfer (fig. 7.9), etc.). In our method, we evaluate this illuminant by computing the amount of light received by the observer. We proceed as follows. We put, at the camera position, a virtual hemispherical surface that is white and perfectly diffuse. This hemisphere will mimic the human eye. Then, we estimate the main illuminant color by averaging the light flux arriving

at this hemisphere (average irradiance). Compared to the technique from Wilkie et Weidlich [WW09], our technique provides better results.

Second, we will present a technique that optimizes the lighting setup (light source size and flux) in a 3D scene (chapter 9). For that, as input, the artist describes his desired aesthetics by providing several target values (image style, mean luminance on the main object, etc.). Then, these values are used to define an objective function. This function is optimized to determine the proper lighting configuration (light source size and flux) that allows to generate an image that matches the desired aesthetics.

Eye-centred color adaptation in global illumination

8

Figure 8.1 – Global illumination rendering without Chromatic Adaptation (left), with Chromatic Adaptation (middle), carpet's color comparison (right).

1 Introduction

Color adaptation also named color constancy is one of the main ability of the human visual system (HVS) [Fai05]. It makes us perceive colors as constant even though the illumination conditions change. For example we perceive the same white color for a white shirt outdoors at midday or indoors under tungsten lights. From a physics point of view, the white shirt color illuminated by the midday sun is quite physically white, but it becomes physically orange under tungsten lights. This shows that the color perception of human beings is not only determined by the spectral distribution of light.

Global illumination rendering engines [PH10] physically simulate light transport to compute the pixel colors. But due to chromatic adaptation, these colors are not those that a human perceives. Neumann et al. [NCNS03] have pointed out that global illumination images have to undergo chromatic adaptation to recover the perceived color. The effect of color adaptation is demonstrated in fig. 8.1.

Chromatic adaptation strongly depends on the used adaptation color (white balancing). Estimating this adaptation color in case of virtual scenes could appear simple as the geometry, the material appearance as well as the light sources are known data. But, as shown later on, the existing approaches do not provide convincing results for complex lighting conditions (high color variation of light and material). Designing a method to efficiently estimate the adaptation color is still challenging.

In this chapter, we propose a new color adaptation method well suited to global illumination. We consider the eye (viewpoint) as a hemispherical sensor around the eye. The adaptation color is computed as the average irradiance color over this sensor. In this way, flickering artefacts are avoided in case of walkthrough unlike the existing approaches. Moreover, unlike other existing methods, our approach is not limited to the view frustrum, as it considers the illumination from all the scene. Experiments have shown that our method outperforms the state of the art methods. The main advantages of our method are:

- the obtained results are faithful to reality,
- temporal coherence in case of animation,
- easy plug-in in existing global illumination renderers.

The chapter is structured as follows. Section 2 provides a brief overview on chromatic adaptation, while section 3 reviews the related works. Section 4 provides details on our adaptation color method. Some results are presented in section 5. Finally section 6 concludes the chapter.

2 Chromatic Adaptation

In his book [Fai05], Fairchild compares different chromatic adaptation models. Most of the chromatic adaptation models rely on Von Kries' hypothesis which considers that each photoreceptor (L, M, S) adapts linearly independently of each other [VK70]:

$$\begin{bmatrix} L_a \\ M_a \\ S_a \end{bmatrix} = \begin{bmatrix} 1/L_W & 0 & 0 \\ 0 & 1/M_W & 0 \\ 0 & 0 & 1/S_W \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix} \tag{8.1}$$

where L,M,S are the initial cone responses, L_a,M_a,S_a the adapted cone signals and L_w,M_w,S_w the adaptation color, also called illuminant color or white color.

Some improvements have been brought to the Von Kries transform such as non linear adaptations [BW92, CW95, NTS81, Gut91] and degree of adaptation[Bre87, Fai91b, Fai91a]. For example, the chromatic adaptation model [CIE98] modifies the Von Kries transform using an exponential non linearity added to the short wavelength stimuli as well as a parameter D that specifies the degree of adaptation:

$$\begin{cases} R_a = [D(1/R_W) + (1-D)]\, R \\ G_a = [D(1/G_W) + (1-D)]\, G \\ B_a = [D(1/B_W^p) + (1-D)]\, B^p \\ \\ p = (B_W/1.0)^{0.00834} \\ \\ D = F\left[1 - \frac{1}{\left[1+2L_A^{1/4}+\frac{L_A^2}{300}\right]}\right] \end{cases} \tag{8.2}$$

In summary, the chromatic adaptation models provide the perceived color x_a of a stimulus x under given lighting conditions x_W (adaptation color). These models can be expressed as a function of two variables x and x_W:

$$x_a = CAT\,(x, x_W) \tag{8.3}$$

Consequently, chromatic adaptation consists of two steps: estimate the adaptation color from the scene or image, then perform the chromatic adaptation transform (see fig. 8.2). The results of chromatic adaptation strongly depends on the accuracy of the adaptation color.

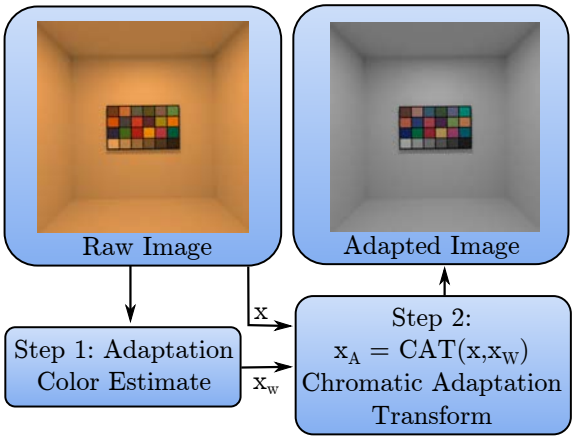


Figure 8.2 – The 2 steps of the chromatic adaptation process.

3 Related works

In the computer vision field, several methods have been proposed to estimate the adaptation color. They are classified according to three main approaches [TTP08]:

1. physics-based (dichromatric models, maxRGB [Lan77], grey-world [Buc80]),
2. statistical (gamut mapping [FHT06], bayesian [GRB+08]),
3. high level methods (high level visual information) [CFB02, VdWSV07].

A comparison of these methods can be found in [Hor06].

Most of the statistical and high level approaches allow to estimate the adaptation color from a set of given illuminants: artificial lighting (tungsten, fluorescent, halogen, etc.), natural lighting (midday sun, rising sun, sunset, cloudy, etc.). Unfortunately, in the context of global illumination these data are not available.

The Grey-World method [Buc80] assumes that the average reflectance in a scene is achromatic, i.e. grey. Then, the color adaptation is computed as the average color of the whole image.

When the image contains large uniformly colored surfaces, the assumption of a Grey-World fails. That is why Weijer et al. [GGW10] proposed a more robust algorithm called Grey-Edge. They use the average of the gaussian derivative of the pixel colors to estimate the adaptation color.

The MaxRGB [Lan77], dichromatic [YCK05] and Retinex [LM71, MPS09, MPS10] methods assume that the brightest pixels correspond to specular reflection of the light sources. Thereby the maximum RGB pixel is used as the adaptation color.

In [FT04, GGW10], it is shown that the above adaptation methods perform similarly. However these methods fail for some test cases such as the one called "world of one reflectance" [RB07]. This test case corresponds to two kinds of scene: (1) a grey world lighted with a colored light and (2) the same world with colored objects lit with a white light. Without chromatic adaptation, the images of the two worlds captured with a camera are the same. However they are actually different when seen by a human viewer.

Color Appearance Model (CAM) [KR09, MFH+02, KJF07] are based on chromatic adaptation. The purpose of the CAM models is to compute the color appearance (brightness, lightness, hue, saturation, chroma and colorfullness) of each pixel of an image. They include a local chromatic adaptation step. For each pixel the adaptation color is computed using either a gaussian low-pass filter [MFH+02, KJF07] or an interpolation between a given achromatic white and the geometric mean of the pixel neighbourhood [RPK+12]. To our knowledge, only few authors address the chromatic adaptation problem in the context of global illumination.

Greg Ward et al. [WEV02] assume that most scenes contain a single dominant illuminant. In this case, they use the main illuminant color as the adaptation color. When a scene does not contain any dominant illuminant, this method can not be used.

Neumann et al. [NCNS03] define the adaptation color as the weighted average of the irradiances of the white surfaces. A white surface close to the camera axis is assigned a higher weight. When a scene does not contain any white surface, this method can not be applied. The authors mentioned that their approach provides unnatural results when the white surfaces are lit with different colors.

Wilkie and Weidlich [WW09] propose a method that overcomes the limitations of the above mentioned approaches. Their method is capable of handling complex lightings conditions. It proceeds as follows. First, it uses a global illumination algorithm to compute and store the incident illumination over all the surfaces. Next, the surface BRDFs are replaced by an achromatic reflectance, then multiplyied by the incident illumination to get a new image. The pixels of this image are assigned a weight depending on the associated BRDF. The adaptation color is the weighted average of the pixel colors. The value of the weight is maximum if the pixel corresponds to a white surface. However, this approach suffers from two limitations. First, as their adaptation color estimate depends on the color of the viewed surfaces and the associated weights, even a small camera displacement can entail high variations of the adaptation color estimate, as detailed later on. This first limitation will be called, from now on, spatio-temporal incoherence. Second, assigning a high weight to white surfaces overestimates their color at the detriment of global illumination.

To sum up, the existing methods suffer from spatio-temporal incoherence because the adaptation color estimate is limited to the field of view. In addition, they provide images that do not look real. The reason is that, when calculating the average color, they (1) assign too high weights to the white surfaces, which favours the color of

these surfaces at the detriment of the global illumination color, and (2) assign the same weight to close and faraway objects (to/from the camera).

4 Our color adaptation method

4.1 Generalization of chromatic adaptation

The basic assumption of chromatic adaptation is that an object is perceived as white whatever the illuminant conditions. The physical color of a white point of a diffuse surface is the color of the irradiance at this point. If we generalize the chromatic adaptation assumption to any diffuse surface, then the perceived radiance color $L_a(\vec{y})$ of a point $\vec{y}$ is the intrinsic reflectance $\rho(\vec{y})$ of the surface. The physical radiance $L(\vec{y}) = [L_L, L_M, L_S]$ of a point $\vec{y}$ of a diffuse surface with a reflectance $\rho(\vec{y}) = [\rho_L, \rho_M, \rho_S]$ and an irradiance $E(\vec{y})$ is given by:

$$L(\vec{y}) = \frac{1}{\pi} E(\vec{y}) \rho(\vec{y}) \tag{8.4}$$

where:

$$[E(\vec{y})] = \begin{bmatrix} E_L(\vec{y}) & 0 & 0 \\ 0 & E_M(\vec{y}) & 0 \\ 0 & 0 & E_S(\vec{y}) \end{bmatrix} \tag{8.5}$$

Equation (8.1) can be rewritten as:

$$L_a(\vec{y}) = \rho(\vec{y}) = \frac{1}{\pi} \begin{bmatrix} 1/L_W & 0 & 0 \\ 0 & 1/M_W & 0 \\ 0 & 0 & 1/S_W \end{bmatrix} [E(\vec{y})] \, \rho(\vec{y}) \tag{8.6}$$

Then, the adaptation radiance $x_W(\vec{y})$ has the same color as the irradiance ones:

$$x_W(\vec{y}) = \frac{1}{\pi} \begin{bmatrix} E_L(\vec{y}) \\ E_M(\vec{y}) \\ E_S(\vec{y}) \end{bmatrix} \tag{8.7}$$

4.2 Eye-centered estimate of the adaptation color

As a single global adaptation color is required for the whole image, the adaptation color is usually computed as the weighted average irradiance ([NCNS03]) or radiance ([WW09]) of the white surfaces within the field of view. In our opinion, the main issues of existing methods stem from the weighting average operation and from the space from which is computed the average values (view frustum or all the 3D space containing the camera). First, the averaging operation assigns a higher weight to the white surfaces. Consequently, the average value overestimates the direct lighting on these white surfaces, which underestimates the indirect lighting color. For example if the scene contains a red spotlight that only lights the only white object within the scene, then the direct lighting due to the red spotlight is overestimated, which

makes the adapted image look unnatural (fig. 8.5). Equation (8.7) shows that there is no reason to assign a higher weight to the color of white surfaces. Second the previous methods do not take into account the distance to the surfaces from the camera when assigning weights since they assign them the same weight. Third, performing an averaging operation only for surfaces lying in the field of view, is not sufficient. Indeed, a small displacement of the camera can result in a high change in the adaptation color. This happens especially when a white surface enters or leaves the field of view.

According to Von Kries hypothesis [VK70] that considers that the photoreceptor are adapted independently of one another, we assume that the adaptation color is the lighting color at the eye location as everyday experience shows that the chromatic adaptation is sensitive to the observer position rather than to the direction he looks at. When the observer is in a room, even tough he sees the illuminant color of another room, he discounts only the illuminant color of the room in which he is located. In other words, the adaptation color depends only on the illuminant color of the room containing the observer.

We also make the assumption that all the directions of lighting arriving at the observer (view angle of 180 degrees) contribute uniformly to its perceived radiance. This means that the chromatic adaptation phenomenon would be sensitive to lighting even when coming from outside the field of view of the camera. This can be explained by the fact that the human eye has an approximately hemispherical field of vision.

Then the chromatic adaptation phenomenon can be modeled by a hemispherical sensor located around the eye. This sensor has an isotropic sensitivity and measures the average color of light arriving at its surface. From a mathematical point of view, it means that the adaptation color x_W^{eye} is the average value of the irradiance colors on a virtual hemisphere located at the eye position and aligned with the gaze direction (fig. 8.3 (a)):

$$x_W^{eye} = \frac{1}{2\pi r^2} \int_{S_\Omega} E(\vec{x}) \mathrm{d}A(\vec{x}) \tag{8.8}$$

where $E(\vec{x})$ is the irradiance at a point $\vec{x}$ on the hemisphere S_Ω of radius r.

The irradiance $E(\vec{x})$ is computed as:

$$E(\vec{x}) = \int_\Omega L_i(\vec{x} \leftarrow \vec{\omega}_i) |\vec{n}_{\vec{x}} \cdot \vec{\omega}_i| \mathrm{d}\sigma(\vec{\omega}_i) \tag{8.9}$$

where $L_i(\vec{x} \leftarrow \vec{\omega}_i)$ is the incident luminance of direction direction $\vec{\omega}_i$ at point $\vec{x}$, and $\vec{n}_{\vec{x}}$ the normal vector at $\vec{x}$.

Our new approach has the following main characteristics:

1. the adaptation color is based on the average irradiance at the eye location rather than on the radiance colors in the field of view, more importance is then given to objects close to the eye,

2. the adaptation color does not focus on the irradiance or radiance color of white surfaces, consequently it does not overestimate local lighting,

3. the adaptation color estimate is not limited to the field of view, it is then more robust to camera displacements.

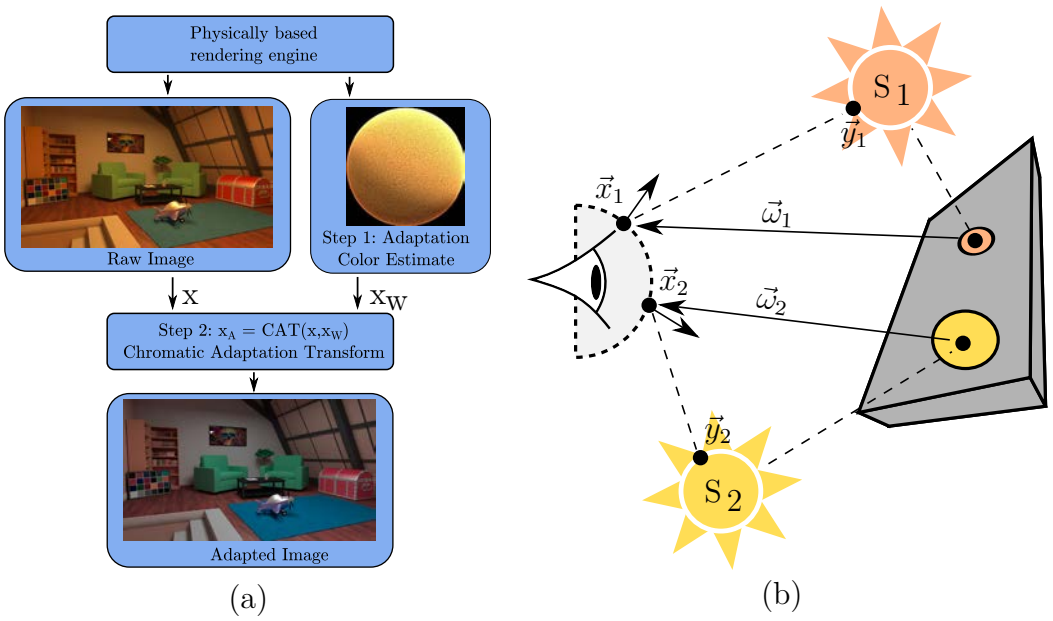


Figure 8.3 – (a) Architecture overview of our chromatic adaptation process. (b) Irradiance color average on virtual hemisphere as color adptation (hemispherical sensor).

The rendering engine computes the average irradiance color over the virtual hemisphere (fig. 8.3 (b)), then the color adaptation transform (CAT) is applied to recover the perceived colors. We use the CIE linear chromatic adaptation transform, called CAT02 [MFH+02], to perform the chromatic adaptation:

$$\begin{bmatrix} X_a \\ Y_a \\ Z_a \end{bmatrix} = M_{02}^{-1} \begin{bmatrix} R_a \\ G_a \\ B_a \end{bmatrix}, \begin{bmatrix} R \\ G \\ B \end{bmatrix} = M_{02} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{8.10}$$

$$\begin{aligned} R_a &= \left(\frac{R_O}{R_W^{eye}} D + (1 - D) \right) R \\ G_a &= \left(\frac{G_O}{G_W^{eye}} D + (1 - D) \right) G \\ B_a &= \left(\frac{B_O}{B_W^{eye}} D + (1 - D) \right) B \end{aligned} \tag{8.11}$$

where $x_a = [X_a, Y_a, Z_a]$ is the adapted color, $[R_0, G_0, B_0]$ the white reference illuminant (D_{65}), $[R_W^{eye}, G_W^{eye}, B_W^{eye}]$ our adaptation color, D is the degree of adaptation and M_{02} the color space transform matrix used in CAT02.

In our current implementation, we use the Mitsuba physically based renderer [Jak10] and its path tracing integrator[1]. In order to estimate the average irradiance over the hemisphere (eq. (8.8)), we split the radiance into direct and indirect components to get $L_i(\vec{x} \leftarrow \vec{\omega}_i) = L_e(\vec{x} \leftarrow \vec{\omega}_i) + L_{ind}(\vec{x} \leftarrow \vec{\omega}_i)$, which leads to:

$$x_W^{eye} = \frac{1}{2\pi r_{S_\Omega}^2} (\phi_{direct} + \phi_{ind}) \tag{8.12}$$

[1]Other integration methods (light tracing, BDPT) can be choose.

with:

$$\phi_{direct} = \int_{S_\Omega} \int_S L_e(\vec{y} \rightarrow \vec{x}) G(\vec{x} \leftrightarrow \vec{y}) \mathrm{d}A(\vec{y}) \mathrm{d}A(\vec{x}) \tag{8.13}$$

$$\phi_{ind} = \int_{S_\Omega} \int_\Omega L_{ind}(\vec{x} \leftarrow \vec{\omega}_i) |\vec{n}_{\vec{x}_i} \cdot \vec{\omega}_i| \mathrm{d}\sigma(\vec{\omega}_i) \mathrm{d}A(\vec{x}) \tag{8.14}$$

where S is all the surfaces comprising the scene. For direct lighting (estimator for section 4.2) we use a simple sampling method by taking a point $\vec{y}_i$ on the light source with the probability $P_a(\vec{y}_i)$ and compute its contribution[2]:

$$\phi_{direct} = \frac{1}{N} \sum_{i=1}^{N} \frac{L_e(\vec{y}_i \rightarrow \vec{x}_i) G(\vec{y}_i \leftrightarrow \vec{x}_i) V(\vec{y}_i \leftrightarrow \vec{x}_i)}{P_a(\vec{y}_i) P_a(\vec{x}_i)} \tag{8.15}$$

where $\vec{x}_i$ is a sample on the hemisphere with the probability $P_a(\vec{x}_i)$, $L_e(\vec{y}_i \rightarrow \vec{x}_i)$ the emitted radiance from $\vec{y}_i$ to $\vec{x}_i$, $G(\vec{y}_i \leftrightarrow \vec{x}_i)$ the geometry factor and $V(\vec{y}_i \leftrightarrow \vec{x}_i)$ the visibility term.

For indirect lighting computation, we can use a well known stochastic ray tracing technique by taking a point on the hemisphere $\vec{x}_i$ and a random incident direction $\vec{\omega}_i$ with the probability $P_\sigma(\vec{x}_i \rightarrow \vec{\omega}_i)$:

$$\phi_{ind} = \frac{1}{N} \sum_{i=1}^{N} \frac{L_{path}(\vec{x}_i, \vec{\omega}_i) |\vec{n}_{\vec{x}_i} \cdot \vec{\omega}_i|}{P_a(\vec{x}_i) P_\sigma(\vec{x}_i, \vec{\omega}_i)} \tag{8.16}$$

where L_{path} is a radiance computed by a path tracer using multiples bounces.

Moreover, the evaluation of the estimators (eqs. (8.15) and (8.16)) is time consuming and depend on the number of samples N. However, the cost of these evaluations is negligible when considering the cost of the global illumination computation for the whole image. Indeed, in our experiments, we used 0.1% of the total samples count used to render an image.

Our adaptation color operation can also be done using commercial global illumination engines which are not open source. In this case, we proceed the following way : (1) we render the 3D scene in which we add a little diffuse white hemisphere located at the camera position just ahead the front clipping plane, (2) we add on orthographic camera looking at the hemisphere, (3) we render an image of the hemisphere and compute the average radiance L_{av} over the hemisphere, (4) finally the average irradiance (πL_{av}) color gives the adaptation color. This implementation works well because adding a small hemisphere does not change significantly the rendering solution.

Our eye-centered estimate of color adaptation gives the expected results (see section 5) in the classic test cases, these results are similar to those obtained with the Wilkie and Weidlich's method. But for scenes with complex lighting our approach provides images that look more natural compared to [WW09].

[2]Note that here no multiple importance sampling is used. However, it is possible to use MIS for the direct estimator as shown in section 3.1.

5 Results

First, we used the benchmark cases listed in [WW09] for color adaptation: direct lighting, world of one reflectance and indirect lighting. In these test cases there is only one illuminant and their corresponding results are known (section 5.1). In addition, we added more complex test cases (section 5.2). By complexity we mean: high irradiance color variations (when the camera moves from one room to another, spotlight), several illuminants with different colors. In these latter cases, the expected results are not available. Therefore we can only discuss if the adapted image looks natural. Finally we also show results for walkthrough (section 5.3).

For each case, we provide three images. The first one, called *raw* image, does not undergo chromatic adaptation, while the two other ones result from chromatic adaptation using the method presented in [WW09] and our approach. As Wilkie's method is the most recent and most efficient for color adaptation in the context of global illumination, we compare it to our approach through different test cases. In Wilkie's method, the parameter w which controls the influence of the "white object information" is set to the value 2 (optimal value according to the authors). In our method, the degree of adaptation D (eq. (8.11)) is set to 1 in order to compute a complete adaptation as in Wilkie's method. From now on, Wilkie's method will be called *WCAM*, which stands for *W*ilkie *C*olor *A*daptation *M*ethod.

5.1 Standard tests cases

The first standard test case consists of a white Cornell box with an orange light source (see fig. 8.4, scene 1). In this case Wilkie's method and ours provide the same adaptation color which corresponds to the expected one.

The second test case is the worlds of one reflectance: White World-Orange Light and Orange World-White Light. Wilkie's method and ours give the expected adaptation colors. The actual object reflectances are recovered in both cases (see fig. 8.4, scenes 2 and 3).

The third case consists of a Cornell box with a diffuse green floor (see fig. 8.4, scene 4). The light source color is white. Due to indirect illumination, a green lighting component is clearly noticeable on the raw image. *WCAM* overestimates this green component, hence the facing wall above the color checker appears slightly purple: (hue=292, saturation=5%). With our adaptation color, the same pixel is still a little bit green: (hue=132, saturation=2%), which looks more natural.

As in all the above test cases, the illuminant color is almost uniform within the scene, *WCAM* and our method provide the same color adaptation. Indeed, as the illuminant color is the same from all the object's scene, computing the adaptation color within either the field of view or a 180 degree frustum lead to the same result. Even for a moderate variation of irradiance color in the scene, our method outperforms *WCAM*, as seen in fig. 8.4.

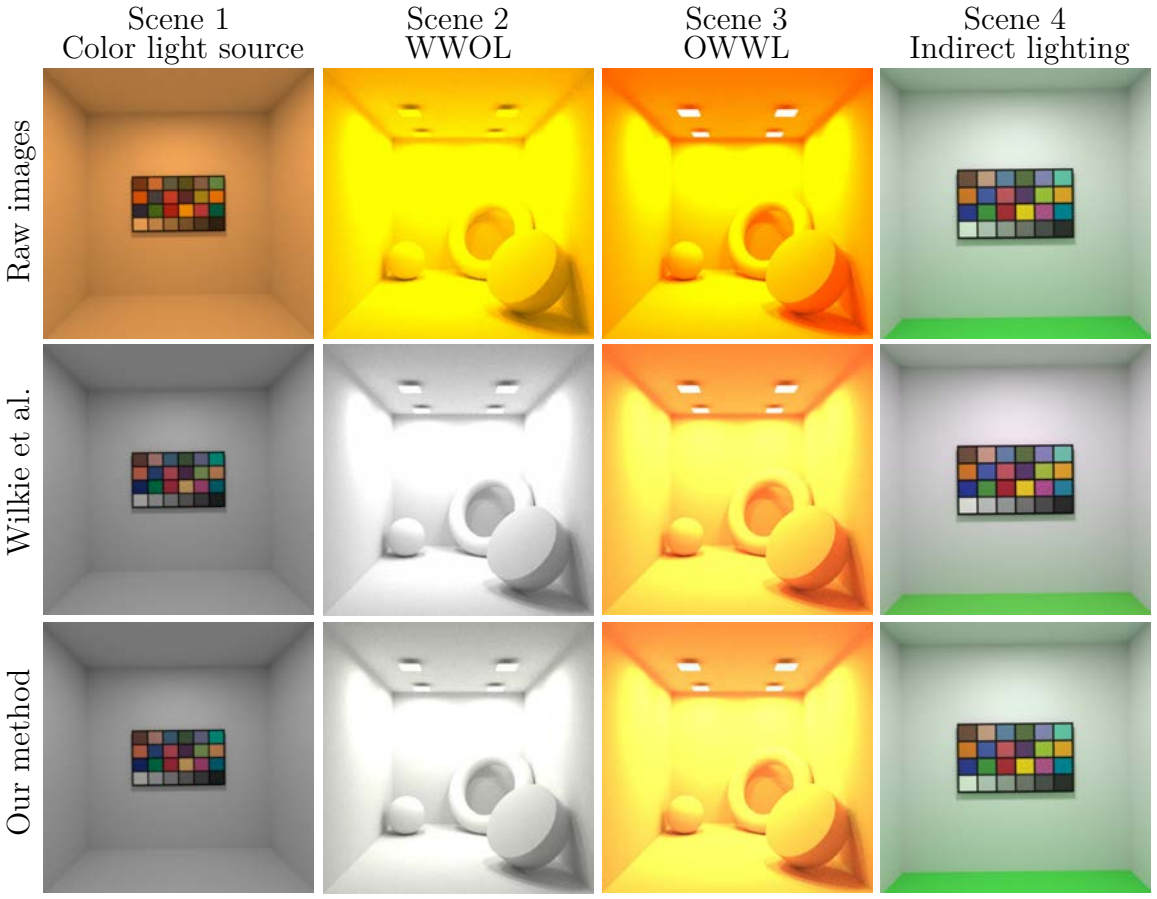


Figure 8.4 – Results in the Wilkie's test cases. 4 different scenes is shown. The first row correspond to the raw image (not corrected), the second row correspond to *WCAM* and the third row correspond to our adaptation color

5.2 Complex tests cases

The first complex test case consists of a room containing a white Buddha statue lit by a red lightspot and a main white light source located on the ceiling (see fig. 8.5). *WCAM* computes a red adaptation color because the white statue is partially lit by a red spotlight. Consequently, after chromatic adaptation, the part of the statue lit by the spotlight gets a white color while the part takes a blue color, which looks unnatural. Instead, our method provides a result that looks more natural. The second test case consists of a scene made of two rooms (fig. 8.6). The first one is lit with four orange light sources while the second is lit with four blue ones. We computed three images (fig. 8.7). In the first one (red view frustum), the second room is not visible. In the second image (green view frustum), both rooms are visible. In the third image (blue view frustum), only the second room is visible while the camera lies in the first room.

In this case, *WCAM* computes three different adaptation colors because it considers only the visible objects to compute the adaptation color. Regarding our method, it also computes three adaptation colors which are very close to each other (fig. 8.7). This can be explained by the fact that our approach is sensitive to where

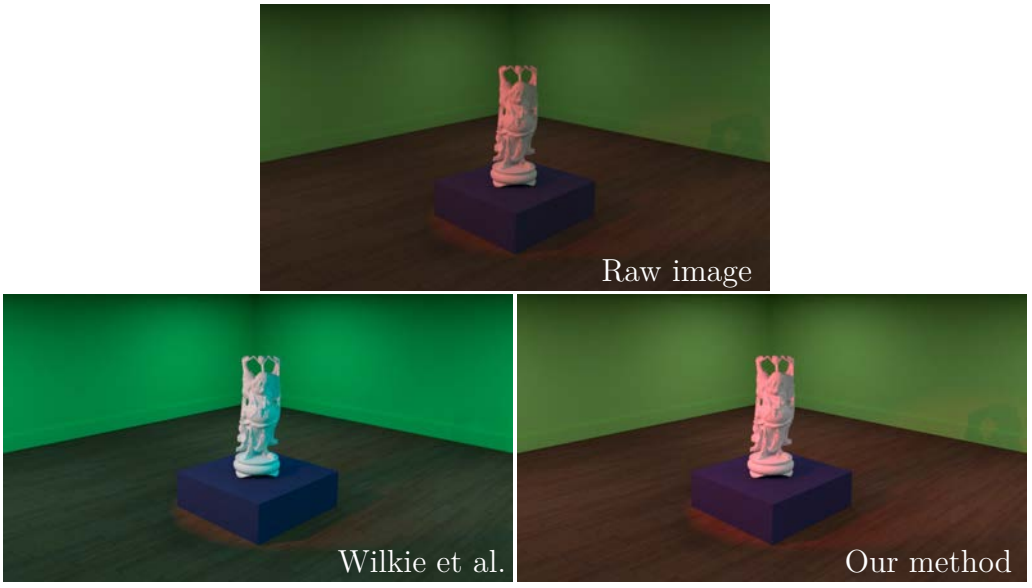
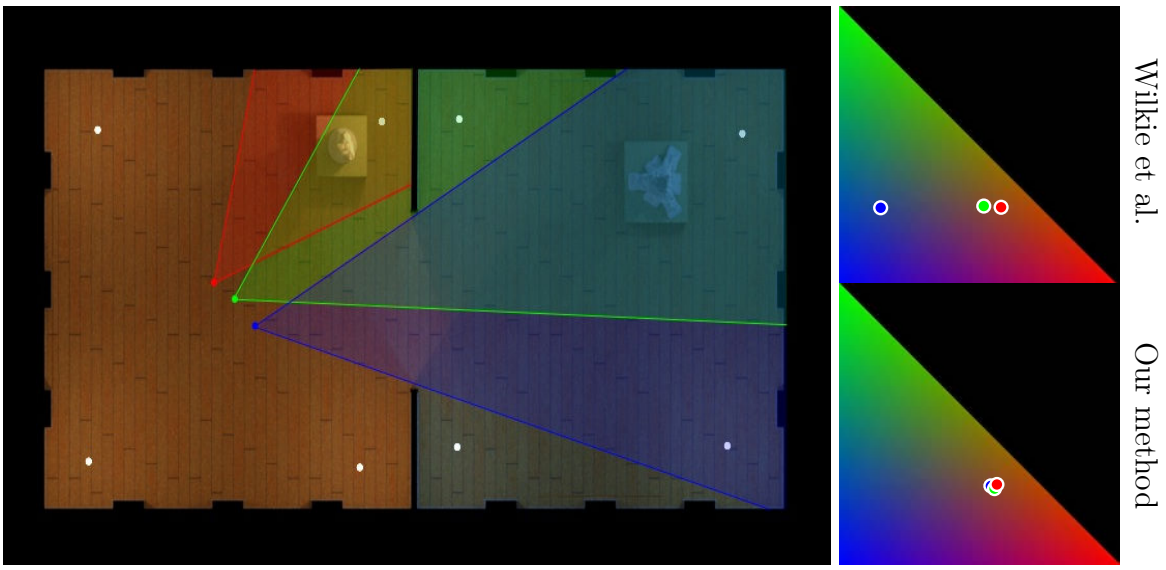


Figure 8.5 – Chromatic adaptation results when a red spotlight partially lits a white statue, raw image (up), using *WCAM* (botoom left), using our adaptation color (bottom right).



(a) Scene configuration (b) Estimates

Figure 8.6 – (a) Map of a 2 room scene and 3 view frustrums (red, blue, green). (b) Chromaticity diagram for RGB color space. The small circles represent the adaptation colors for the 3 frustums, obtained with *WCAM* (top) and our method (bottom). The circle color (red, green, blue) corresponds to the frustum.

the camera is located rather than the visible surfaces. These results shows that lacks spatio-temporal coherency.

5.3 Sequence tests cases

We tested our algorithm on video sequences to demonstrate its ability to meet the constraint of spatio-temporal coherency. The video sequence shown in this chapter

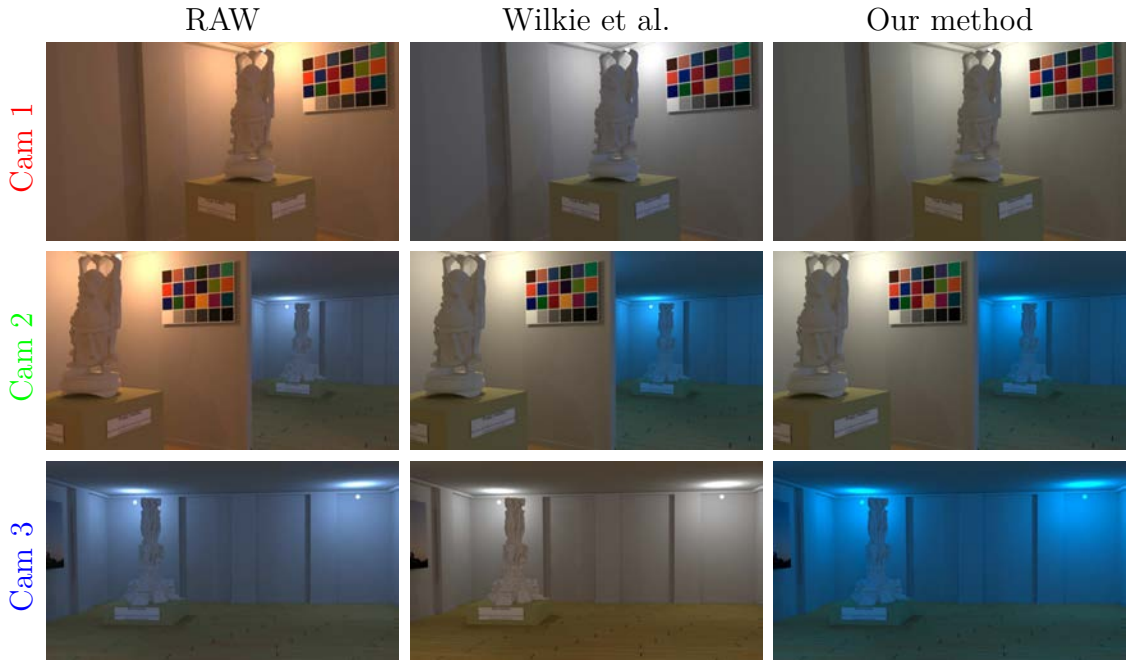


Figure 8.7 – Spatial coherency of the adaptation color estimate in the case of 2 room scene. raw image(left), using *WCAM* (middle), using our method (right)

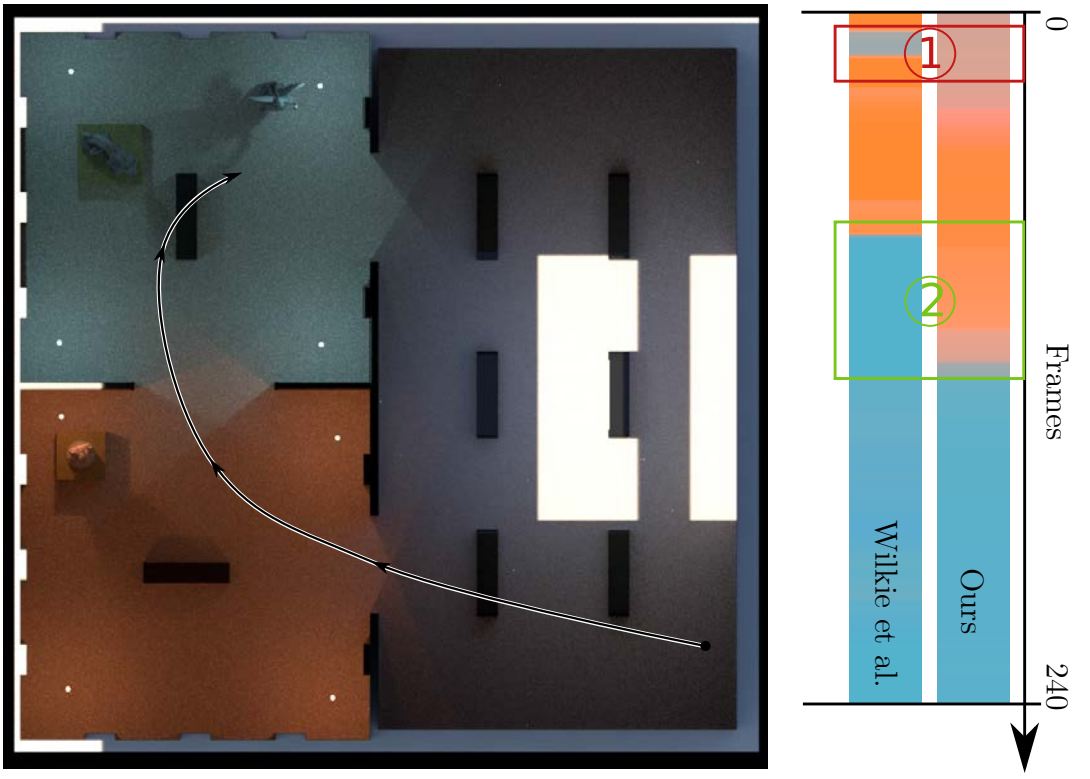
is a walkthrough into a scene composed of three rooms with different illuminant colors (fig. 8.8 (a)). The camera starts in a room lit with a white light, then goes through another room lit with an orange light,to finally reach a room lit with a blue light.

Figure 8.8 (b) shows the adaptation color for each frame of the sequence. Unlike *WCAM*, our method provides a smooth variation of the adaptation color. Note that the *WCAM* results in a sudden change of the adaptation colors, especially at the beginning of the sequence ((1) insert box in fig. 8.8 (b)). In addition, the *WCAM* adapts too early to the illuminant color of the third room ((2) green insert in figure fig. 8.8 (b))

The sudden change in the adaptation color in *WCAM* occurs between frames 5 and 10 (fig. 8.9). The rendered image does not change so much between these frames. Only the specular reflection of one light source on the bench (red insert in fig. 8.9, middle) changes significantly, which highly changes the adaptation color. The reason is that *WCAM* adapts mostly to this specular reflection while our method adapts to the average irradiance color over the hemispherical sensor. The video sequence is provided as an additional material. Other results is shown in fig. 8.10.

6 Conclusion

We have proposed a simple, accurate and spatio-temporally coherent method to automatically estimate the adaptation color for chromatic adaptation in the context of global illumination. Our adaptation color is computed as the average of irradiance color over a virtual hemispherical sensor centred at the camera location and aligned with the camera axis. First, we have demonstrated that our algorithm out-



(a) Camera path (b) Adaptation Color

Figure 8.8 – (a) Map of a 3 room scene and camera trajectory. (b) Adaptation colors for each frame of the video sequence with *WCAM* (left), with our method (right)

performs the state of the art methods especially when the illuminant color varies in a scene. Second, as our method is spatio-temporally coherent, it can be used in video sequences. Third, it can be easily implemented using an open or closed source rendering engine.

Future work would extend the use of hemispherical sensor to tone mapping (also called light adaptation) video sequences, thanks to the spatio-temporal coherence property of the hemispherical sensor. Another research avenue is to propose an alternative (other norms) to just averaging the irradiance color to compute the adaptation color.

Acknowledgements

The material in this chapter is, in part, a reproduction of the material published in Adrien Gruson, Mickael Ribardiere and Remi Cozot "Eye-Centred Color Adaptation in Global Illumination", Pacific Graphics 2013 [GRC13].

Figure 8.9 – Video sequence. Spatio-temporel coherency issue between frame 5 (up) and 10 (bottom), raw image (left), *WCAM* (middle) with specular highlight changing the estimate and ours (right).

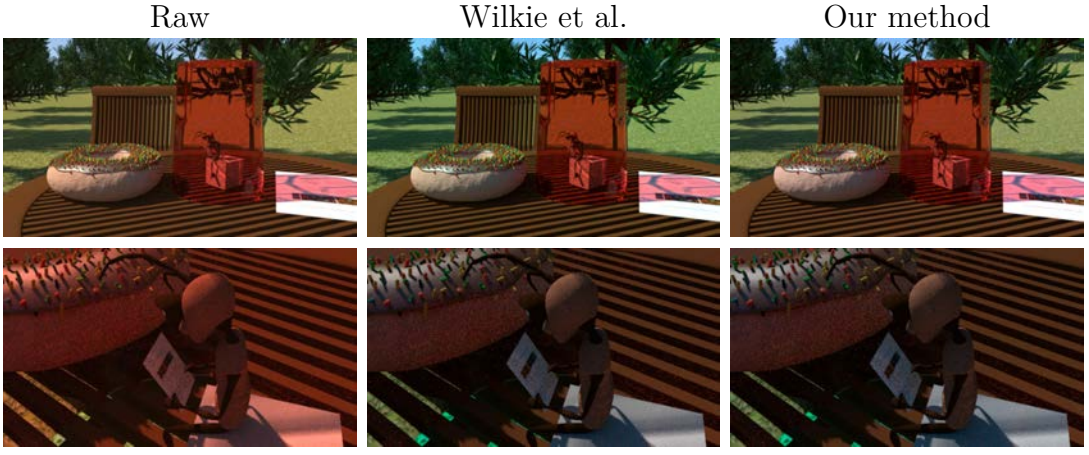


Figure 8.10 – Scene addressing transmission through a glass. On the right side of the top images, there is a red cylindrical glass containing objects (Ant, White cubic sugar). The camera is outside the glass. The same scene is rendered from inside the glass (bottom row of images). This case is similar to Wilkie's test called "Transmissions colors". In the first row, the adaptation color estimated with *WCAM* method has a higher red color value due to the caustic on the white sheet of paper. As for the second row, the sensor is inside the red glass, the incoming illuminant is red, which gives the same result for the two methods.

Automatic aesthetics-based lighting design with global illumination

9

Figure 9.1 – Given an initial scene configuration, the aim of the technique presented in this chapter, is to produce different targeted aesthetics. In this technique, we will mainly address two target aesthetics: High-key and Low-key aesthetics.

1 Introduction

In computer graphics, global illumination rendering methods take into account multiple phenomena such as reflection, refraction and scattering. However, because of these numerous phenomena, it is difficult to know beforehand what the resulting image will look like. Lighting plays an important role in the appearance of a scene and a change in the configuration of light sources can lead to different aesthetics in the final rendered image. For instance, we would like an object of a scene to be bright, with a lot of contrast, while the background should be dark and uniform. Answering such specifications can represent a time-consuming and tedious effort if performed through a trial-and-error process.

To obtain a satisfying result automatically, the user has to express his intent by providing a set of target values as properties of the final image. The goal is to change iteratively the parameters of the 3D scene to fit the user intent. A lot of parameters of the scene can be taken into account. Among these are:

- Position, size, flux of the light sources,
- Reflectance of the materials,
- Distribution of the objects in the scene.

Methods used to compute lighting parameters to fit a given set of constraints are called inverse lighting methods. An example can be found in the work of Shacked et al. [SL01], where an objective function is minimized in order to make an image of a 3D object easy to understand. Our approach is more general, as our objective function accounts for target parameters defined to express the intent of the user: it expresses the distance between the current lighting design and the desired result. This function is minimized to find the optimal set of parameters.

In this chapter, we will mainly address two target aesthetics used in films and photography: high-key and low-key pictures as shown in fig. 9.1. High-key and low-key aesthetics focus on one object of the scene, called henceforth *main object*. High-key images are bright, do not have too much contrast, nor feature dark shadows cast by the main object. Low-key images have a darker tone, their contour lines are highlighted and the background is usually black. They also feature high values of contrast.

We designed a new inverse lighting framework. Given a set of target parameters provided by the user, we set up a platform that can render a scene with global illumination techniques and allow to minimize the objective function to find the desired parameters (light source size, flux, reflectance) to meet the user's intent.

In our case, we use two light sources: a key-light and a fill-light. The key-light is the main light source in the scene. It is used to light the main object. The fill-light is used to control the shadows cast by the main object. The key-light sets the main direction of lighting, and the fill-light lights the main object from a side angle relative to the key-light.

2 Related Works

Finding a light configuration that produces an image with the desired characteristics is an ill-posed problem. It was first introduced as inverse-lighting problem by Kawai et al. [KPC93]. Inverse-lighting methods allow us to compute an ideal lighting configuration for the 3D scene, given a lighting specification. This lighting specification is considered as a target that represents the desired result.

Inverse-lighting methods differ by the type of target they use, the optimization process, but also by the parameters they tune. In the rest of this section, we will distinguish two categories of inverse-lighting methods: image-based and global methods. For more details, the reader can refer to the survey on inverse-lighting problems by Patow and Pueyo [PP03].

2.1 Image-based methods

Inverse-lighting in the context of radiosity has been discussed by Schoeneman et al. [SDS+93]. The user provides a target radiosity vector Ψ, which elements are the desired radiosity of each patch of the scene, by painting on an image of the scene with a brush. The goal of this method is to find the emittance of each light source in the scene. The radiosity equation is solved for each light source independently. When considering each light source at the same time, the radiosity vector $\hat{\Psi}$ can

be defined as a linear combination of the radiosity vectors Φ_i obtained for the light source i:

$$\hat{\Psi} = \sum_{i=1}^{n} w_i \Phi_i \tag{9.1}$$

In order to find the contribution factor w_i of each light source, the objective function $||\Psi - \hat{\Psi}||$ has to be minimized using a least-square method. Light source positions and orientations are fixed, and only their emittance is automatically computed. Moreover, the user has to provide the target image using the painting interface, which can be a time-consuming task.

The SAIL model presented by Zupko and El-Nasr [ZEN09] does not use an image of the scene as a target. Instead, the provided target is an image of an illuminated sphere. The target image is thus independent of the scene. However, generating and understanding such an image can be a difficult task for the user, as the simple geometry of the sphere may only allow a coarse specification of the lighting goals. The method discussed by Zupko and El-Nasr [ZEN09] uses a configuration consisting of two light sources (key light and fill light). It computes a parameter set (key-to-fill ratio, orientation and position of the key light) that minimizes the distance between two vectors of appearance metrics : one vector computed on the illuminated sphere and another computed on the image of the 3D scene.

The main difficulty in a model such as SAIL is to provide an image of an illuminated sphere that communicates the desired lighting configuration. Instead of using a target image, the methods presented in the next sub-section use a set of target values that better characterize the desired configuration.

2.2 Global Methods

The methods in this category do not use an image as a target. Instead, they use a set of target values for descriptors that describe an image's aesthetics. These methods proceed by minimizing an objective function, often a linear combination of several terms.

Kawai et al. [KPC93] use a linear combination of three terms that measure the mean brightness, non-uniformity and peripheral characteristics of the lighting of the scene. The weight assigned to each of these terms is specified by the user so that he can construct his own constraint. The optimization variables can be light source direction, surface element radiosity, reflectivity or emissivity. The objective function is minimized iteratively using the Broyden-Fletcher-Goldfarb-Shanno non-linear optimization method (BFGS) [PW00].

Fernandez et al. [FB14] introduce an efficient way to solve a radiosity-based inverse lighting problem. Desired values for the mean and variance of the radiosity values are provided, and an optimization operation is performed. Many other existing works use radiosity-based rendering methods to optimize the placement of light sources in architectural settings [CSFN11, CdAS12, FB12].

The method introduced by Shacked and Lichinski [SL01] computes the lighting configuration that is optimized for understanding the shape and fine details on the scene's objects. It uses an objective function f_q that is a linear combination of terms

defined as follows:

$$\begin{aligned} f_q &= \sum_{s \in S} w_s f_s \\ S &= [mean, var, hist, grad, edge, dir] \end{aligned} \tag{9.2}$$

The terms f_s are listed below:

- f_{mean} measures the distance between the mean luminance and a target value;
- f_{var} measures the distance between the luminance variance and a target value;
- f_{hist} measures the distance between the luminance histogram and an equalized histogram;
- f_{grad} measures the magnitude of gradients;
- f_{edge} evaluates the appearance of edges;
- f_{dir} measures the elevation of light sources (this term is optional).

The method used in the approach mentioned above for rendering is not physically accurate: it relies on diffuse and specular intensity values for point light sources. These two values are considered as variables to be optimized, along with the direction of the point light sources (θ_i, ϕ_i). Optimization is performed iteratively using a gradient descent method.

In the following section, we will discuss the above presented methods, and present the different properties our method should have.

2.3 Discussion

Providing a target image [SDS+93, ZEN09], with some possible different parametrisation, is a time-consuming process, and although the result may feature the right highlighted areas, it may not communicate the aesthetics expected by the user.

The work presented by Shacked and Lichinski [SL01] uses an objective function to communicate the shape of the main object as well as possible. The object is isolated on a uniform background and is not considered as part of the scene. Also, the rendering method used is not physically correct as it only use OpenGL rendering, hence it does not account for indirect lighting. The objective function is made of several terms that only depend on parameters computed from an image of the scene, together with an optional constraint for light direction. It is then minimized to provide an optimal set of parameters (orientation of light sources, specular and diffuse intensities).

However, this method has been designed to improve the understanding of 3D shapes. The goal of our work is to cover a broader range of aesthetics. The user should be able to control the lighting design process through target values used by the objective function. As we are working in the context of global illumination, the lighting design process will have to account for multiple reflections. Our approach should be generic enough so that it does not rely on a specific global illumination

rendering algorithm. The solution should also provide flexibility in terms of nature of light sources, whereas only point lights are considered by Shacked and Lichinski [SL01]. In particular, we are interested in studying the influence of the size of the light sources on the final result.

Like Shacked and Lichinski [SL01], we specify a main object for the scene to evaluate the aesthetics properly. This is analogous to photography where an isolated subject is considered. The rest of the image that does not represent this main object is considered as a background. Contrary to Shacked et al., we do not consider objects on a uniform background, but we consider them as part of a complete scene, and the appearance of the background contributes to the overall aesthetics.

The objective functions are usually defined for a single purpose, such as in Shacked and Lichinski [SL01] where the goal is to make the final image easy to understand. However, we want to have a broader range of possibilities. Using our method, the user can express his aesthetics intent by defining target parameters. Thus, our objective function is completely configurable: it is made of several terms, each one taking into account a target value. These different terms are computed using the luminance values on the image. Many image descriptors have been considered for describing image aesthetics using a learned classifier [DJLW06, DOB11]. However, as we are mainly interested in light-based aesthetics, we do not consider composition or color-appearance, and we focus on luminance values. In the next section, we will introduce our approach for defining the objective function and for designing the optimization process. Our contributions to inverse-lighting consist in:

- describing different aesthetics;
- considering both object and background appearances;
- taking indirect lighting into account;
- handling area light source parameters.

3 Overview of the approach

This section details the different steps of our method (fig. 9.2). Then, in section 4.1 we will give further details on the free variables, the evaluation of the objective function and the optimization process.

Our goal is to allow the user modify the aesthetics of a scene by changing the appearance of the main object and the background. In order to make it feasible, our method takes as input (orange box in fig. 9.2):

- the 3D scene description which specifies the object, the materials' properties, the lights' position and the camera's position;
- the free parameters x whose values have to be determined automatically;
- the target values which describe the desired aesthetics. These target values are constant and have to be set before running the optimization process. We will detail them later on.

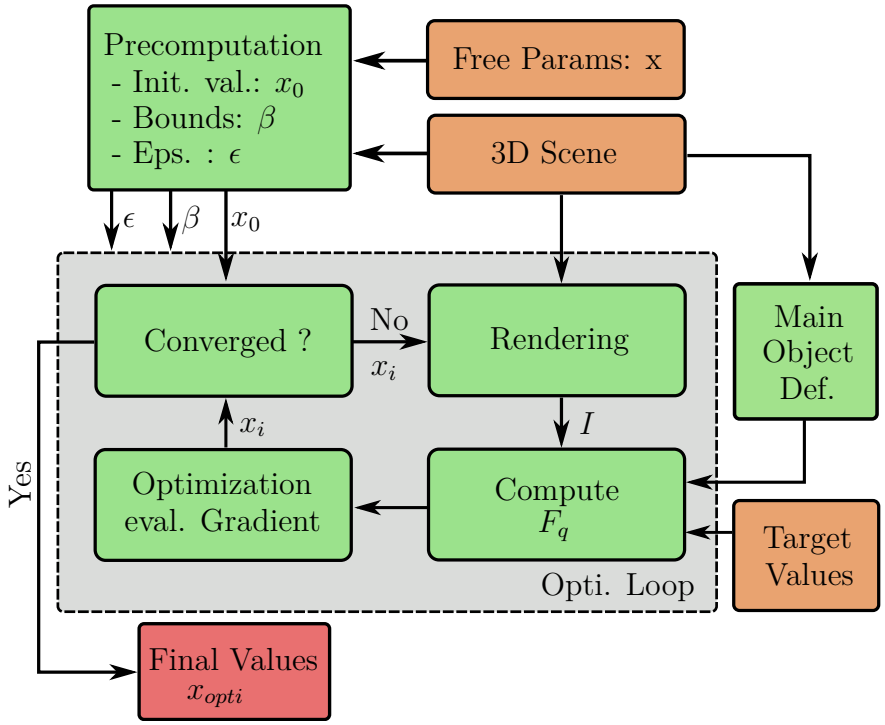


Figure 9.2 – Our framework. Green boxes represent the different processings of our algorithm. Orange boxes represent the value provided by the user. Red box is the result of our algorithm.

Before running the optimization process, we automatically find initial values x_0 in a pre-processing step. Moreover, to run the optimization, we need to specify for each free parameter its bounding values β and gradient evaluation step ϵ. Furthermore, to evaluate the aesthetics of the scene using the objective function, we have to differentiate the different zones of the image by defining a mask image ("Image Map" step in fig. 9.2). This mask image will indicates whether a pixel corresponds to the main object or to the background.

The optimization loop (gray box in fig. 9.2) is used to find a set of parameters that minimize the objective function. At each iteration, the scene is rendered using a global illumination technique. Then new values of the parameters to be optimized are computed. The choice of the rendering method does not have an influence on the overall framework, as computing the objective function only requires the luminance image. Methods such as radiosity, path tracing or photon mapping can be used, without compromising the overall framework[1]. Using the previously computed image I and the image map, the objective function f_q is evaluated. The objective function f_q is computed as a linear combination of terms:

$$\begin{aligned} f_q &= \sum_{s \in S} w_s f_s \\ S &= [meanBack, meanObj, varBack, varObj, grad, hist] \end{aligned} \tag{9.3}$$

Each term in f_q takes into account a target value. The main difference between

[1]However, less noise there is in the rendered results, more stable the optimisation is. Moreover, biased methods can leads to not optimal setup if a non-biased method is used for rendering the final image settings.

our objective function and the one used by Shacked and Lichinski [SL01] is that all the terms we use can be configured by the user. The terms of the objective function are normalized distances from a user-provided value and a value computed on a specific zone of the image:

- $f_{meanBack}$ and $f_{meanObj}$: the mean luminance of the background and the object respectively;
- $f_{varBack}$ and f_{varObj}: the luminance's variance on the background and the object respectively;
- f_{grad}: the gradient value over the principal object of the scene;
- f_{hist}: the normalized luminance histogram of the image.

Then, the optimization algorithm computes gradient values using ϵ values and decide with some criteria to continue the optimization process or not. At the end, the algorithm will return final values x_{opti}.

In our approach, in contrast to Shacked and Lichinski [SL01], the target values are not computed from the scene, but are directly provided by the user. By doing so, we allow the field of attainable aesthetics to be broader. For instance, in the method proposed by Shacked and Lichinski [SL01], the lighting aesthetics that makes shapes more difficult to understand would be penalized, whereas in our approach, if the user gives such a specification this type of aesthetics would be possible.

4 Approaching an aesthetics with function minimization

The goal of our method is to automatically design a lighting configuration that brings a target aesthetics to the scene in the context of global illumination. In our context, we want to obtain an image with a certain aesthetics, not necessarily the most understandable one, and we want the user to have control on the aesthetics of the scene using target values for image descriptors. Furthermore, in the context of global illumination, the range of attainable aesthetics is wider than when only considering direct illumination. Within that goal, as we want our approach to be as generic as possible, any rendering method can be used and any kind of light source can be considered.

In the next subsections, we will introduce the different terms of the objective function used for the minimization process (section 4.1). Then, we will list the different free variables that the algorithm can tune automatically (section 4.2) . Finally, we will describe the optimization process and the automatic pre-computation step (section 4.3).

4.1 Objective function

The objective function (section 3) is a linear combination of several terms. Each term is defined as a distance between a value (mean luminance, variance, gradient

amplitude) computed on a rendered luminance image I and a target value provided by the user. Each terms defines a distance in $[0, 1]$. The different terms of the objective function are detailed in the following subsections.

4.1.1 $\mathbf{f_{meanObj}}$ and $\mathbf{f_{meanBack}}$

These two terms control the mean pixel luminance value on the object and on the background respectively. We use two different terms for the object and for the background as the mean luminance values for the foreground and the background are characteristics of high key and low key images. In terms of aesthetics, these term control the overall brightness of the object and the background. The target values $t_{meanObj}$ and $t_{meanBack}$ are mean luminance values, defined in $[0, 1]$. The terms $f_{meanObj}$ and $f_{meanBack}$ are computed as follows:

$$f_{meanObj} = \frac{|\bar{l}(object) - t_{meanObj}|}{max(t_{meanObj}, 1 - t_{meanObj})} \tag{9.4}$$

$$f_{meanBack} = \frac{|\bar{l}(background) - t_{meanBack}|}{max(t_{meanBack}, 1 - t_{meanBack})} \tag{9.5}$$

where $\bar{l}(object)$ and $\bar{l}(background)$ are the mean luminance over object pixels and background pixels respectively, computed using a global illumination algorithm.

4.1.2 $\mathbf{f_{varObj}}$ and $\mathbf{f_{varBack}}$

These two terms measure the distance between a target value and the standard deviation of the luminance values of the main object's pixels and the background's pixels respectively. Here again, we distinguish between background and object pixels to control how smooth the background and the object appear. The expressions for the terms f_{varObj} and $f_{varBack}$ are given below:

$$f_{varObj} = \frac{|\sigma(object) - t_{varObj}|}{max(t_{varObj}, 1 - t_{varObj})} \tag{9.6}$$

$$f_{varBack} = \frac{|\sigma(background) - t_{varBack}|}{max(t_{varBack}, 1 - t_{varBack})} \tag{9.7}$$

where $\sigma(object)$ and $\sigma(background)$ are the luminance standard deviation over the main object's pixels and the background's pixels respectively. We use a different normalization from that of Shacked et al. as they use an additional factor for the target value when the objects in the scene contains a wide range of reflectances [SL01].

4.1.3 $\mathbf{f_{grad}}$

The gradient term f_{grad} is used to control the shading gradient on the object's surface. If the target value is high, then the shape of the object will appear more pronounced. A low value will make the contours more subtle. Using a pair of Sobel filters, we compute two gradient images $G_x = A \star I$ and $G_y = A^T \star I$. We obtain for

each pixel $p_{i,j}$ the gradient vector $\nabla l_{i,j} = (G_x(i,j), G_y(i,j))$. The average shading gradient norm is then computed as follows:

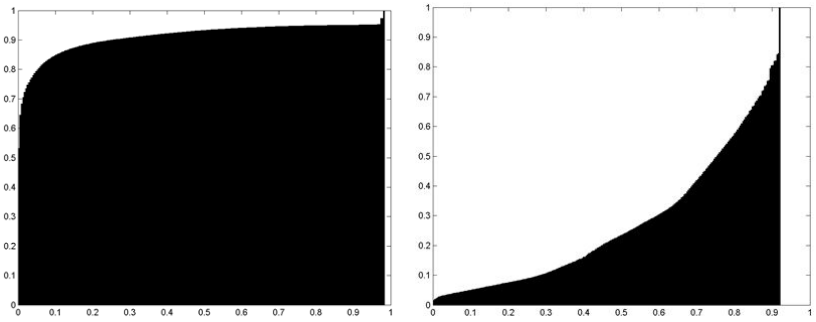
$$g(object) = \sqrt{\frac{1}{N_{obj}} \sum_{p_{i,j} \in object} |\nabla l_{i,j}|^2} \tag{9.8}$$

where *object* is the set of pixels marked as object's pixels in the image mask, and N_{obj} the number of these pixels. The final term f_{grad} is defined as:

$$f_{grad} = \frac{|g(object) - t_{grad}|}{max(t_{grad}, 1 - t_{grad})} \tag{9.9}$$

4.1.4 $\mathbf{f_{hist}}$

As explained in the work of Martin et al. [MFSG08], luminance histograms must be used with other descriptors to express an image's aesthetics. However, Sicre [Sic13] uses supervised learning on well known aesthetics (such as high key, low key, and medium key images), therefore a signature histogram can be extracted for each class. We use these signature histograms as target that will be compared with the luminance image's histogram. The signatures are represented as cumulated histograms in fig. 9.3. These signature are obtained by averaging each histogram bin over a set of low-key images (fig. 9.3a) and high-key images (fig. 9.3b) respectively.



(a) Low-key signature **(b)** High-key signature

Figure 9.3 – Signature cumulated histograms

Minimizing the objective function makes the image's histogram closer to the target histogram. The f_{mean} and f_{var} terms have an influence on the value of the term f_{hist}. However, f_{hist} describes the complete luminance distribution and does not provide any spatial information, whereas f_{mean} and f_{var} evaluate statistics for distinct areas of the image, hence these terms are more complementary than they are redundant.

This term is evaluated as the Matsusita distance between two discrete distributions [CS02], defined as follows:

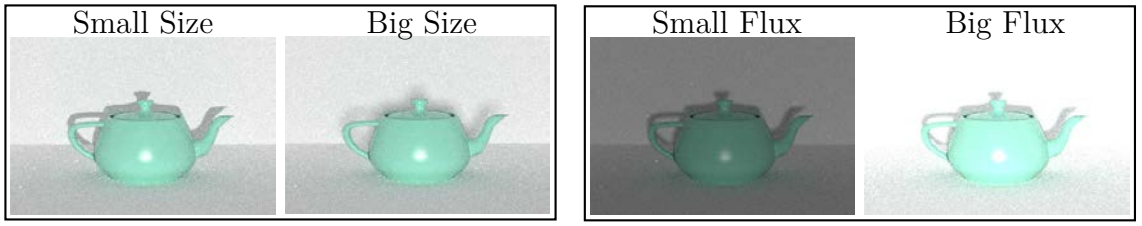
$$D(p,q) = \sqrt{\sum_{i=0}^{b-1} (\sqrt{p_i} - \sqrt{q_i})^2} \tag{9.10}$$

where b is the number of bin in the cumulated histogram, p_i and q_i is the $i-th$ cumulated value of the signature histogram and the image's histogram. For proper normalization, the final term is:

$$f_{hist} = D(p,q) \cdot \sqrt{\sum_{i=0}^{b-1} max(q_i, 1-q_i)} \tag{9.11}$$

As explained in [CS02], this $D(p,q)$ possess all the properties of a metric. $f_{hist} = 0$ implies that the target image and the rendered image have the same luminance histograms. However, this histogram metric is really simple and does not account for distribution translations. Another histogram metric is left for future work [2].

4.2 Free variables



(a) Size influence (b) Flux influence

Figure 9.4 – Influence of the light parameters on the rendering. (a) With a constant size, the flux controls the light's contribution. (b) With a constant flux, the size controls the shadows strength.

The output of our algorithm is a set of lighting parameters values that minimizes the objective function. A point light can be defined using a 3D position and a luminance value. An area light source has a size that also influences the lighting of the scene. As shown in fig. 9.4, the radius of the area light and the flux can change the aesthetics of the rendered image. However, changing its size does not have the same impact as changing the flux on the objective function's value. This could cause a problem in the optimization process as the gradient on the size variables will be low and provide few information. Instead of using the flux, we use the self-emitted luminance which introduces correlation between this value and the scale of the light source. Indeed, for an isotropic area light source, the luminous flux is:

$$\Phi = \pi \cdot A \cdot S \cdot L_e \tag{9.12}$$

where A is the original light area without scaling, S is the scaling factor (which allows to modify the size of the light source) and L_e is the self-emitted luminance of the source.

Moreover, the scene setup itself influences the space of achievable aesthetics. For example, a bright material assigned to the background makes it impossible to

[2] Indeed, a histogram metric with translation is for now an issue for the optimization algorithm. However, finding a good histogram metric and a compatible optimization algorithm is an interesting way to improve our results.

(a) Initial image with x_0 **(b)** High key target **(c)** Low key target **(d)** High key target with background modification **(e)** Low key target with background modification

Figure 9.5 – Optimization of our algorithm for the two different aestetics. In the default setup, the background is bright and the algorithm converges to a High Key image (fig. 9.9b). On the contrary, this background makes it impossible to reach a Low Key setup (fig. 9.5c). By considering the background albedo as free variable we can reach the two different aesthestics (figs. 9.5d and 9.5e).

obtain a low-key image (fig. 9.5). To overcome this problem, we propose to the user an option to change the material properties for all background objects. This is done by modulating the V channel of the HSV material's color.

4.3 Optimization

The optimization step uses the L-BFGS-B algorithm [ZBLN97]. It is a memory limited implementation of the BFGS algorithm used by Kawai et al. [KPC93] that takes into account additional constraints such as bound parameters. During the optimization process, the objective function is minimized. The algorithm uses a gradient descent technique to minimize the objective function. However, our objective function does not have an analytic expression. Partial derivatives are approximated numerically by evaluating the objective function for consecutive steps for each free variable.

However, the free variables have different orders of magnitude and need to be bounded. As such, for estimating the gradient, a proper order of magnitude of epsilon needs to be chosen. A too small value may produce no significant change in the image. A too large value of epsilon may compromise the minimization. The parameters themselves are scaled so that they have similar order of magnitude. Thus, the gradient is comparable for each free variable. The initial values x_0 for the free variables are important for the optimization: if we start from a position where only one light source contributes to the scene lighting, the optimization could only modify that light source and fall into a local minimum. Moreover, the free variables have to be bounded, because of physical properties or geometrical considerations. These questions will be addressed in the following paragraphs. The bound values are found automatically during the step called "precomputation step" in fig. 9.2.

Bounds: In the optimization process, some free variables have to be bounded. Indeed, the self-emitted luminance is considered to be strictly positive, but it does not have any upper boundary. Moreover, the bounding values for the V channel

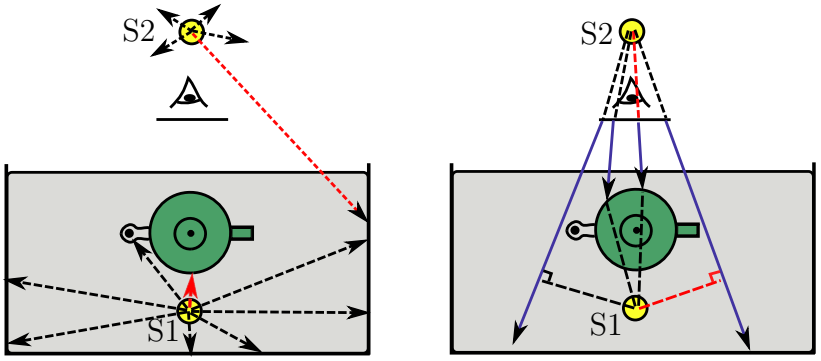
modulation of the HSV background's albedo are $[0.1, 10]$. The scale of the area light source is strictly positive. However, we have to make sure that the light source does not collide with:

- the main object's surface viewed from the camera;
- the camera frustrum;
- and optionally, with all objects of the scene.

We need to compute the minimal distance from a light source to the above mentioned of objects (camera frustrum, scene surfaces ... etc.). However, as we dispose of the scene geometry, we can either cast random rays or change the light source size to estimate this minimal distance. Then, this minimal distance is used to determine the maximum scaling factor of the light source allowed during the optimization process.

In practice, two different techniques are used to compute this minimal distance for each light source (fig. 9.6):

- Minimal Object distance: we generate a ray originating at the center of mass of a light source toward a randomly sampled point p on this light source. Then we compute the distance between p and the first intersection between the ray and the surface's scene. This process is repeated by choosing other points p. The minimum value of the resulting distances is the minimal object distance.
- Minimal Frustrum distance: we increase the light source size until its gets visible. If the light source gets visible, after a dichotomic search, we determine the highest light source size for which the light source is not visible.



(a) Object distance (b) Frustrum distance

Figure 9.6 – Example of scene with two spherical light sources $S1$ and $S2$. The two strategies to compute the light source's size bound: (a) distance to all objects (optional) and (b) distance to the camera's frustrum. The red ray shows the minimal distance for each light source.

Initial values: In order to compute the initial value for the free variables associated with the light sources (self-emitted luminance and size). However, these two types of variable have some correlation with the final image brightness (see section 4.2). So, to determine the initial values of those variables, we first initialize the

light source size by a deterministic procedure. Them, we use this size to find the self-emitted luminance by a simple optimization process[3].

The aim of this initialisation is that each light source brings a minimal contribution. In practice, we proceed as follows:

1. Scaling of the light source: we set the light source size as half the radius of the object's bounding sphere. If this size is bigger than the computed upper bounds, we reduce it to have a valid size.

2. Self-emitted luminance of the light source: we fix the light source size and change only the self-emitted luminance. We modify the self-emitted lumiance so that the mean luminance value of the image is equal to $0.5/n_{light}$ where n_{light} is the number of light sources.

Steps and scaling: Steps (used by the gradient descent for each variable) are computed as 10% of the initial values. The free variable values are scaled automatically before the optimization process, so that they have all comparable orders of magnitude.

Discussion: Because the terms of the objective function may collide, all the target values are not necessary attainable: for instance, as explained in section 4.1.4, f_{mean}, f_{var} and f_{hist} are strongly correlated. Moreover, because the optimization algorithm is based on gradient descent, it is possible that the algorithm gets stuck at a local-minimum. However, with the automatic computation of initial values and gradient steps, we try to prevent local minima and speed-up the optimization process. To compute the different statistics on pixel luminance, we use floating point luminance values ranging from 0 to 1. Floating point precision usage is important to prevent any imprecision on gradient evaluations because of image quantization.

5 Results

The method described above as been implemented in Python using the LBFGS-B [ZBLN97] optimization method, implemented using scipy [sci13], numpy and openCV for image processing [num13, ope13]. The rendering step has been performed with the Mitsuba rendering platform [Jak10]. To speed-up the optimization process, all the intermediate images have a lower resolution than the final one. This helps speeding-up the optimization. Our implementation has been applied to different scenes with different target values to obtain an optimal lighting configuration that corresponds to a desired aesthetics. This section highlights some of the results we obtained using this method.

Table 9.1 presents the two sets of target values and the weights used in the objective function. These two configuration will be used for all our scenes. The High-key configuration "HK" will target a bright background and low variance on

[3]Note that if the rendering engine can have HDR image output, the self-emitted luminance for each light sources can be straightforwardly deduced.

the object. We use a High-key histogram signature. The Low-key configuration "LK" will target a bright object and a dark background to get contrast. Moreover, the variance on the object should be high, because we want bright and dark areas on the main object. The signature is a Low-key histogram signature. For the weights of the objective function, we use the same weights for the background and the main object. We use a bigger weighting value for the histogram term f_{hist} as it describes the image globally.

function term	HK	LK	weight
$t_{meanObj}$	0.47	0.47	0.5
$t_{meanBack}$	0.78	0.04	0.5
t_{varObj}	0.24	0.63	0.4
$t_{varBack}$	0.17	0.04	0.4
t_{grad}	0.39	0.39	0.2
t_{hist}	high	low	1.0

Table 9.1 – Configurations and weights used for the target function.

fig. 9.5 shows a teapot inside an white box. There are two spherical light sources in the scene, the key-light is behind the camera and the fill-light is behind the teapot. The two light sources are aligned. We use photon mapping with final gathering for the rendering. We speed-up the rendering by using radiance caching. Table 9.2 shows the final values produced by our algorithm for the *teapot* scene. The Low-key configuration (fig. 9.5c) failed because of the bright background and diffuse inter-reflections. When considering the background albedo as a free variable, we can get a satisfying Low-key image (fig. 9.5e). This is visible in the table 9.2 where all the final values, except the variance on the object v_{varObj}, are closer to the target values (table 9.1). This problem with the variance on the object is due to the light sources emplacement where light sources are aligned with the camera. The two High-key results (figs. 9.5d and 9.9b) still have shadows cast by the object on the bottom of the box. However, as shown in table 9.2, the final values are close to the target values provided in table 9.1.

	without back.		*with back.*	
final value	HK	LK	HK	LK
$v_{meanObj}$	0.44	0.03	0.42	0.46
$v_{meanBack}$	0.83	0.03	0.78	0.10
v_{varObj}	0.14	0.01	0.13	0.13
$v_{varBack}$	0.19	0.01	0.20	0.03
v_{grad}	0.47	0.03	0.44	0.48
f_{hist}	0.102	0.110	0.092	0.188
f_q	0.240	1.066	0.266	0.555

Table 9.2 – Final values for the *teapot* scene (fig. 9.5), f_{hist} and f_q values.

Figure 9.7 shows the result of our method for two different scenes: on the left, the *Girl* scene with two planar light sources and a key-light close to the object; on the right, the *Creature* scene, with a planar light source behind the camera and two

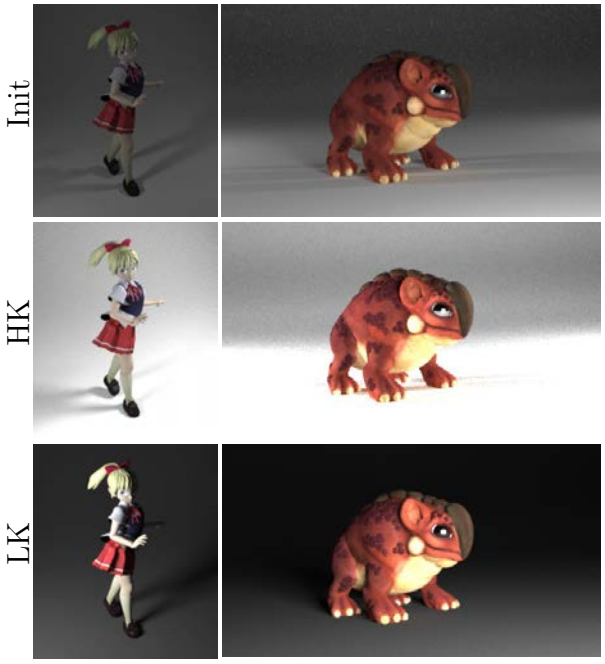


Figure 9.7 – *Girl* (left column) and *Creature* (right column) scenes. First row: initial configuration of the scene. Second row: optimization for a High-key aesthetics. Third row: optimization for a Low-key aesthetics.

spherical light sources in the background. These two scenes have been rendered using path tracing. As the background area in this scene is very large, we use two sources to light it.

As we can see in the second and third rows of fig. 9.7, we can reach two different aesthetics with the same scene. Final values for the *Girl* and the *Creature* scenes are shown in table 9.3. In the former scene with HK configuration, we obtain a bright background with low variance and a low variance on the object. In the LK configuration, we obtain a dark background with low variance and a high variance on the object, which generates contrast. For the *Creature* scene with HK configuration, we still have shadows on the background as the background is large and difficult to fill completely. Detailed plots are shown in fig. 9.9. In the LK configuration, we can see a high contrast on the main object, and a dark background.

Figure 9.8 shows the results with a more complex scene and the results are summed up in table 9.4. This scene is rendered with a bidirectional technique. This shows the flexibility of our method. Three different light sources are optimized at once. However, the glossy and uneven background surface makes it difficult to reach Low-key and High-key aesthetics. In both cases, the variance of the background has to be low, which is difficult to get with glossy reflections. In the case of the Low-key image with the background albedo considered as a free variable, as the albedo of the background is reduced, reflection from the object's glossy surfaces are reduced, therefore the background appears less glossy. Final values are presented in table 9.4. Another complex background setting is shown in fig. 9.1 with two light sources and without using some tuning of the background's albedo.

The influence of the rendering process on variance has to be taken into account. Indeed, Monte Carlo estimator provides a robust estimation of the $f_{meanObj}$ and

	Girl		*Creature*	
final value	HK	LK	HK	LK
$v_{meanObj}$	0.47	0.45	0.45	0.44
$v_{meanBack}$	0.82	0.18	0.78	0.13
v_{varObj}	0.21	0.33	0.23	0.25
$v_{varBack}$	0.13	0.06	0.21	0.11
v_{grad}	0.51	0.46	0.50	0.50
f_{hist}	0.137	0.278	0.187	0.184
f_q	0.284	0.609	0.308	0.529
timing (sec.)	569	810	718	467

Table 9.3 – Final values, f_{hist} and f_q values for the *Girl* and the *Creature* scenes (fig. 9.7).

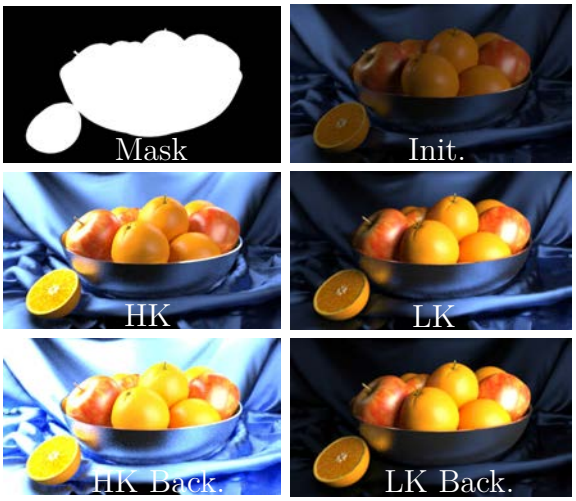


Figure 9.8 – *Fruit basket* scene. First row: the image map and the initial image. Second row: The HK and LK results without changing the background. Third row: The HK and LK with the background albedo as free variable of the objective function.

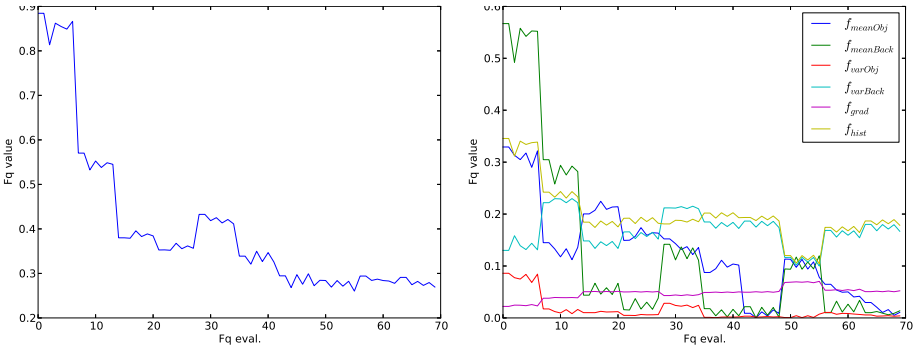
$f_{meanBack}$ values. However, the other terms of the objective function, such as variance, are more sensitive to noise due to the Monte Carlo based rendering process. To overcome this problem, we choose settings to minimize noise. An automated solution is left for future works.

6 Future improvements

Our method can be further improved to provide even more flexibility for expressing the desired image aesthetics. For example, the style is only given by a set of parameters (variance on the main object, etc.) defined by the user. In the future, these parameters could be determined by using a target image in a similar fashion to color transfer applications. Other parameters can be extracted from the input image (light direction, light color, etc.) and used to build a more faithful objective function. User study of our technique, versus manual tweaking, needs to be con-

	without back.		*with back.*	
final value	HK	LK	HK	LK
$v_{meanObj}$	0.52	0.46	0.70	0.46
$v_{meanBack}$	0.41	0.16	0.68	0.07
v_{varObj}	0.27	0.25	0.23	0.27
$v_{varBack}$	0.29	0.15	0.30	0.06
v_{grad}	0.58	0.52	0.74	0.54
f_{hist}	0.241	0.197	0.079	0.125
f_q	0.618	0.602	0.456	0.419

Table 9.4 – Final values for the *Fruit Basket* (fig. 9.8, second row), f_{hist} and f_q values with and without considering the background albedo as free variable of the objective function.



(a) f_q **(b)** objective function's terms

Figure 9.9 – The leftmost figure shows the value of the objective function for each evaluation step of the optimization process. The plot takes into account the evaluations needed for computing gradients using finite difference method. The rightmost figure shows the values of the different terms of the objective function.

ducted to properly evaluate our technique in terms of quality, easiness and speed. In general, describing and controlling the desired aesthetics needs further research.

The optimization procedure can be improved. Indeed, gradients are computed by finite differences by restarting the rendering for each evaluation. However, gradient computation can be automatically embedded in the used rendering technique, which could speed up the optimization operation (up to 5 times).

Moreover, we know that some free variables have a specific impact on the rendering as shown in fig. 9.4. A multi objective optimization algorithm could be more robust and find the optimal configuration faster.

More complex lighting configurations need to be studied. However, wrong light positions can prevent the technique from reaching a correct minimization. In our current technique, the position of the lamps are fixed. The reason for this is that the searching space for the light orientations is too large. Parametrizing lighting position could be an elegant solution to find light orientations automatically.

7 Conclusion

This chapter presented a new method for inverse rendering that accounts for target parameters given by the user to specify a desired image aesthetics. These target parameters are used to parametrize an objective function that is then minimized iteratively. After each iteration, new values for the free variable of the scene (in our case, scale and luminance of the light sources, material of the background) are set up, and the objective function is evaluated again.

We tested our approach with two classic photographic styles (high-key and low-key) and we obtained good results that convey the desired aesthetics, as specified by the target parameters. In the objective function, we use simple luminance metrics. It could be interesting to investigate the use of higher level metrics such as saliency that would bring a higher correlation with the perception of the user.

Acknowledgements

The different scenes can be downloaded at blendswap.com. *Girl* is by Sharon. *Creature* is by Kevin Hays. *Fruit basket* is by Andrew Price.

The material in this chapter is, in part, a reproduction of the material published in Vincent Leon, Adrien Gruson, Remi Cozot and Kadi Bouatouch "Automatic Aesthetics-based Lighting Design with Global Illumination", Pacific Graphics 2014 [LGCB14], Short paper track.

Conclusion 10

In this thesis, we focus on helping the artist to generate a computer generated image that matches his/her intent. To achieve this, we have presented a background on physically-based rendering and introduced Monte Carlo and Metropolis techniques.

First, we have focused on improving the rendering step. To this end, we have proposed several new algorithms such as:

- a new accelerated rendering algorithm to render participating media. This algorithm use GPU to achieve fast computation.
- extension of SPPM [HJ09] to handle participating media. In this technique, we have also used Metropolis algorithm to be robust and efficient in case of complex scenes.
- a new importance function that better distributes the relative error over the image plane. We got a better convergence rate with a perceptual metric that helps the artist to better visualize the final result. This technique does not need any pre-computation and is automatic.

Second, we have proposed several other techniques to help the artist to extract useful information or automatized redundant tasks:

- a new way to estimate the reference illuminant in 3D scenes. This technique outperforms previous techniques.
- an automatic technique to setup the lighting (light source size and flux) to achieve a desired aesthetics.

We hope that all these proposed techniques go one step further to make computer graphics generated images more friendly to the artist.

1 Future work

Rendering step Rendering step is computationally intensive and high quality images need a long time to be generated. For consumer computers, mixed CPU-GPU rendering algorithm can be a good solution to get higher efficiency. Indeed, GPU is really powerful but needs regular and coherent computation to be efficient. So, the idea will be to split the evaluation of the rendering equation into parts: the GPU will take care of the coherent computation (direct contribution, diffuse bounces, etc.) and the CPU will take care of the rest (incoherent paths, rare event sampling, etc.).

Metropolis light transport (MLT) techniques are a good candidate for complex scene rendering. However, they cannot be used for now in production due to local exploration behavior. Indeed, the Markov Chain can slowly visit the path space. A consequence, this creates wrong brightness levels in some parts of the image. To reduce this problem, Metropolis algorithm needs more stratification and fast chain mixing. A possible combination with classical Monte Carlo rendering techniques can be a solution to the problem.

Regarding our new importance function, we have only shown its application to photon mapping rendering techniques. Photon mapping is a robust rendering technique. However it has a major drawback: its bad convergence speed. Extensions to support vertex connection and merging (VCM) [GKDS12, HPJ12] with our importance function will be an important improvement in terms of quality. A combination in the same spirit as Hachisuka et al. [HKD14] needs to be studied.

Primary sample space (PSS) [KSKAC02] and path space MLT [VG97] have their own strength/weakness. PSSMLT is more robust but less computationally efficient than path space MLT. An interesting option will to mix these two MLT techniques. Moreover, PSSMLT can be more efficient using blocking random number update (to be able to reuse some parts of the path). Mutations parameters (size) need to be automatically adapted using AMCMC [AT08] or genetic algorithms [LFCD07]. Getting better local adaptability for mutation size will improve MLT performance.

In this thesis, we have focused more on image rendering. However, dedicated rendering algorithms for animation (several coherent rendered images) generation need to be deeper studied. Indeed, we have seen that the supplementary dimension needed for participating media helps find more efficient rendering techniques. In the same idea, we have planned to study how MLT algorithms can take advantage of the additional temporal dimension.

In general, efficiency and robustness are of high importance for rendering algorithms. However, we need to keep in mind that such algorithms will be used by an artist. We need to try to reduce the number of parameters needed for this step.

Tools for the artist Additional tools are developed to help the artist in his/her creation process. For example, some tools initialize some parameters automatically to get as output an image close to his intent. Then, the artist only needs to focus on the creation details.

For that, the tools need to be as fast as possible (faster if the artist sets the parameters manually). For example, the technique from the last chapter need to be optimized to be practical. This can be done by using a specific rendering method. Moreover, a graphics interface needs to be developed to make possible to the artist to refine his original intent.

More general configurations need to be supported. For example, the number of light sources, their positions and color need to be taken into account.

Résumé en Français

1 Introduction

L'utilisation de l'image de synthèse (image générée par un ordinateur) est de plus en plus répandue dans l'industrie. Par exemple, elle est utilisée dans l'industrie du divertissement pour produire du contenu comme des films ou des jeux video, ou encore, dans le cadre de la prévisualation de projets/prototypes (architectures, etc.).

L'image générée peut être d'un niveau de réalisme varié. Dans cette thèse, nous allons nous intéresser aux images photoréalistes. Ce type d'image est produit en se basant sur les lois de la physique qui régissent la propagation de la lumière.

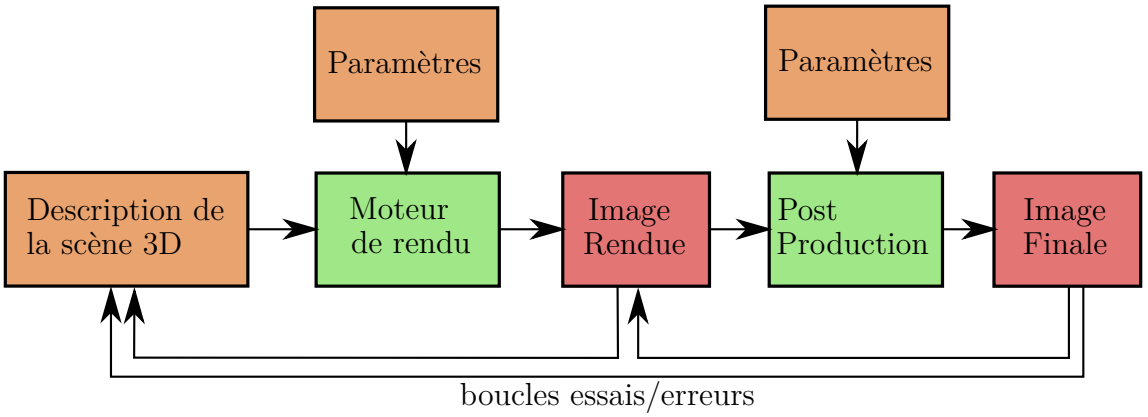


Figure 10.1 – Les différentes étapes de génération d'une image de synthèse. Plusieurs étapes sont répétées jusqu'a ce que l'artiste atteigne le style d'image visée.

Le plus souvent, l'utilisateur (par exemple, un artiste) produit une image de synthèse pour atteindre un but donné (esthétique, ambiance, informations, etc.). Pour ce faire, l'utilisateur peut jouer sur les différents paramètres d'entrée (les boîtes oranges dans fig. 10.1), tels que la scène 3D et les paramètres utilisateur, à chaque étape de génération. Plus précisement, une scène 3D contient une caméra virtuelle, un ensemble de sources de lumière et plusieurs objets interagissant avec la lumière. Chaque objet possède des paramètres qui décrivent la manière dont l'objet influence la propagation de la lumière. Par ailleurs, les différentes étapes pour atteindre l'image finale (les boîtes vertes dans fig. 10.1) dépendent d'un certain nombre de paramètres qui peuvent être plus ou moins parlants pour l'utilisateur. Pour toutes ces raisons, il est difficile pour un utilisateur d'obtenir directement le résultat voulu sans passer par plusieurs étapes d'essais/erreurs (fig. 10.1). Le but de cette thèse est d'apporter des solutions/amélorations à l'utilisateur pour faciliter le processus de création d'images.

2 Algorithmes de rendu photoréalistes

Avant de rentrer dans le vif du sujet, nous allons présentrer les algorithmes de rendu photoréalistes. Le but de ces méthodes est de trouver l'ensemble des chemins de lumière (respectant les lois de la physique) qui contribuent à l'image vue depuis la caméra. Pour ce faire, chaque chemin doit connecter une source de lumière à la caméra en rebondissant, ou non, sur différents objets de la scène 3D. A chaque rebond, la valeur et la direction du flux lumineux transporté changent en fonction du matériau de l'objet touché.

Veach [Vea97] a proposé un cadre mathématique pour exprimer ces chemins (chapitre 2). Son cadre modélise uniquement les interactions lumineuses de surface à surface. Le but de l'algorithme de rendu est d'évaluer l'équation de rendu [Kaj86] :

$$I_j = \int_{\mathbb{P}} f_j(\overline{\mathbf{x}}) \mathrm{d}\mu(\overline{\mathbf{x}}), \tag{10.1}$$

où I_j est la luminance au pixel j, $\mathbb{P}$ l'espace des chemins, $\overline{\mathbf{x}}$ un chemin de lumière de longueur k et composé des sommets $x_0, ...x_k$. La fonction de contribution f_j au pixel j peut s'exprimer comme :

$$f_j(\overline{\mathbf{x}}) = L_e(\vec{x}_0 \rightarrow \vec{x}_1) T(\overline{\mathbf{x}}) W_e^j(\vec{x}_{k-1} \rightarrow \vec{x}_k) \tag{10.2}$$

$$T(\overline{\mathbf{x}}) = G(\vec{x}_0 \leftrightarrow \vec{x}_1) \prod_{i=1}^{k-1} f_r(\vec{x}_{i-1} \rightarrow \vec{x}_i \rightarrow \vec{x}_{i+1}) G(\vec{x}_i \leftrightarrow \vec{x}_{i+1}) \tag{10.3}$$

où $L_e(\vec{x}_0 \rightarrow \vec{x}_1)$ est la luminance de la source de lumière et $W_e^j(\vec{x}_{k-1} \rightarrow \vec{x}_k)$ la fonction de filtrage au pixel j. Par ailleurs, $T(\overline{\mathbf{x}})$, l'atténuation du chemin $\overline{\mathbf{x}}$, est un produit des interactions lumineuses avec les matériaux $f_r(\vec{x}_{i-1} \rightarrow \vec{x}_i \rightarrow \vec{x}_{i+1})$ et des facteurs géométriques G.

Il existe d'autres objets non surfaciques qui interagissent avec la lumière. C'est le cas des milieux participatifs (fumée, feu, nuage, etc.), où chaque point du volume peut émettre, absorber ou diffuser la lumière. Prendre en compte ce genre de phénomènes lumineux est difficile car très coûteux. En effet, l'intégration ne se fait plus uniquement sur certains points de la scène (surfaces) mais en tout point de la scène (volume). Cependant, il est possible d'étendre le cadre mathématique de Veach pour pouvoir prendre toutes ces interactions lumineuse en compte (fin chapitre 2).

L'évaluation de l'équation de rendu est coûteuse. Pour rendre cette évaluation efficace, il existe plusieurs possibilités comme :

1. **utiliser un materiel dédié / rapide** : normalement, les algorithmes de rendu sont mis en œuvre sur CPU[1]. Cependant, la carte graphique (GPU[2]) possède une puissance de calcul qui dépasse celle des CPU. En effet, cette puissance brute est due à l'utilisation massive du calcul parallèle. Cependant, comme les CPU et les GPU ont des architectures différentes, il faut repenser les algorithmes de rendu CPU pour qu'ils puissent s'exécuter efficacement sur GPU.

[1]Central process unit : le processeur de l'ordinateur responsable des calculs généraux
[2]Graphics process unit : unité de calcul dédiée à l'affichage et aux calculs 2D/3D.

2. **utiliser un outil/modèle mathématique plus évolué** : pour une scène 3D quelquonque, il est difficile de trouver l'ensemble des chemins contributifs. Cependant, en utilisant des modèles ou des propriétés mathèmatiques, il est possible d'améliorer l'efficacité et la robustesse des algorithmes de rendu. Par exemple, au cours du processus de rendu, l'algorithme peut extraire de l'information sur la scène et l'utiliser pour concentrer le travail dans les zones intéressantes de la scène 3D.

Plus particulièrement, les estimateurs de Monte Carlo sont souvent utilisés dans les algorithmes de rendu. Ce type d'algorithme est basé sur un processus stochastique qui crée des chemins de façon aléatoire. Cependant, pour avoir un temps de calcul raisonnable, ce type d'algorithme utilise un échantillonnage par importance. Cette technique consiste à répartir les échantillions dans les zones intéressantes à calculer.

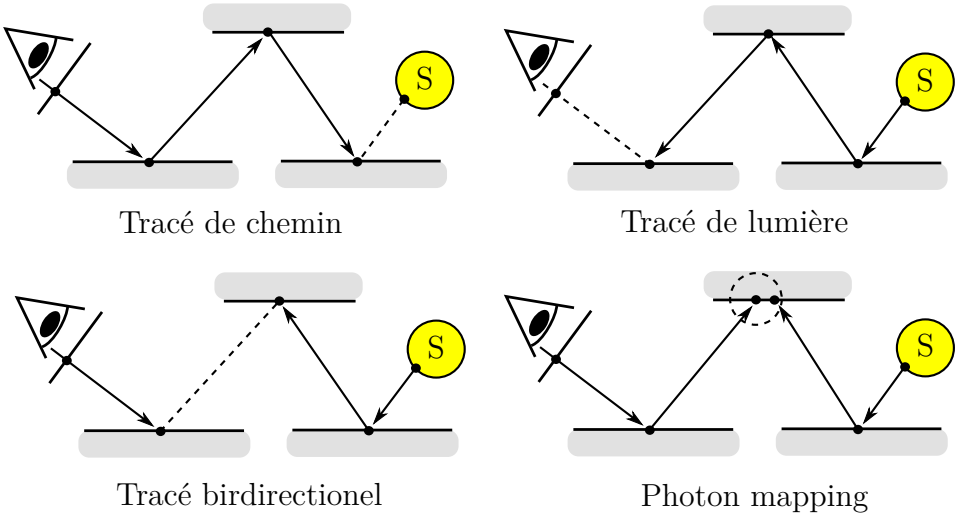


Figure 10.2 – Les différentes façons de construire un chemin de lumière.

Dans la pratique, cela se traduit par différentes façons de construire des chemins (fig. 10.2, abordé dans le chapitre 3 de la thèse). Par exemple, cela pourrait consister à créer un chemin partant de la caméra jusqu'à ce qu'il touche une source de lumière. Le but est de trouver une façon de construire les chemins pour faciliter l'exploration du processus aléatoire.

Cependant, se limiter strictement aux lois de la physique peut poser problème. En effet, certains phénomènes lumieux induisent des contraintes dans la construction d'un chemin (surface spéculaire, lampe directionnelle, etc.). Ces contraintes rendent difficile l'échantillionnage de l'espace de chemins de façon efficace. Une approache courante, appelée photon mapping, consiste à relaxer les contraintes physiques pour agrandir virtuellement l'ensemble des chemins contributifs. Dans la pratique, cela se traduit par la possiblité de connecter deux chemins différents s'ils ont un sommet proche spatialement. Cela permet de rendre les algorthimes de rendu plus robustes au détriment d'un biais[3].

[3]Le biais est dû au fait que l'on considère des chemins de lumière valides alors qu'ils ne le sont pas. Cela se traduit par une luminance différente dans certaines parties de l'image.

3 Motivations

Les motivations de ce travail de thèse ont pour objectif de simplifier le travail de l'artiste dans son processus de création. Pour cela, nous nous sommes concentrés sur 2 points principaux :

1. **développer des techniques de rendu rapides ou/et robustes** pour des scènes 3D générales. Il est important que la technique soit assez rapide pour donner un aperçu à l'artiste afin qu'il puisse rectifier sa scène 3D au plus vite. Cependant, il est important que cette visualisation rapide/intermédiaire soit fidèle au rendu final[4].

2. **développer des techniques pour extraire des informations utiles ou automatiser certaines tâches redondantes**. Le but étant de réduire le plus possible les boucles essais/erreurs en simplifiant la recherche des paramètres corrects pour que l'utilisateur atteigne son but.

4 Sommaire des contributions

4.1 Méthode de rendu

Chapitre 5 : Calcul des interactions dans des milieux participatifs sur GPU. Les interactions lumineuses dans un milieu participatif sont complexes et consomatrices en temps de calcul. Calculer ces phénomènes lumineux, de façon interactive, reste un défi. Les méthodes de type DOM permettent de calculer une solution indépendante du point de vue. Ces techniques sont coûteuses en mémoire et en temps de calcul car elles discrétisent l'espace des directions et le volume. Fattal [Fat09] a proposé une méthode de ce type permettant de réduire le temps de calcul et l'espace mémoire. Cependant, sa solution n'est adaptée qu'à une mise en œuvre sur CPU.

Dans ce chapitre, nous proposons une implémentation GPU de l'algorithme de Fattal. En réorganisant l'algorithme original, il est possible de le rendre massivement parallèle tout en évitant des synchronisations entre les différents calculs. Cependant, ce type d'algorithme reste gourmand en mémoire et peut poser un problème sur GPU où la mémoire est limitée. Pour résoudre ce problème, nous proposons une méthode de streaming des informations qui nous permette de ne plus être limité par la mémoire du GPU.

Chapitre 6 : Rendu progressif basé photon mapping prenant en compte les milieux participatifs. Le problème avec la technique précédente est qu'elle ne gère pas les interactions surface-volume. Pour lever cette limitation, une solution moins rapide mais plus générale consiste à utiliser un estimateur de Monte Carlo. Hachisuka et al. [HJ09] ont proposé une méthode de rendu progressive basée sur le

[4] On entend par rendu final un processus de rendu de plusieurs minutes/heures pour générer une image de haute qualité.

photon mapping (SPPM). Cependant, leur technique ne permet pas de prendre en compte les milieux participatifs. Zwicker et al. [KZ11] ont proposé une solution à ce problème en généralisant la méthode SPPM. Pour cela, ils effectuent de multiples rendus bruités et moyennent les valeurs calculées pour produire un rendu de haute qualité.

Dans le même temps, Hachisuka et al. [HJ11] ont proposé une amélioration de SPPM (VSPPM) en utilisant l'algorithme de Metropolis-Hasting [MRR+53, Has70] (chapitre 4). Cet algorithme permet de répartir automatiquement les échantillons dans les zones intéressantes de l'espace des chemins. Cependant, cette amélioration ne permet toujours pas de prendre en compte les milieux participatifs.

Dans ce chapitre, nous proposons d'étendre VSPPM aux milieux participatifs. Cette nouvelle méthode permet de faire des rendus de façon robuste et efficace d'une scène quelquonque. Par ailleurs, nous réutilisons l'algorithme de Metropolis pour pouvoir guider les photons vers les zones intéressantes. Pour cela, nous proposons une structure de données qui permettent de stocker les chemins venant de la caméra sous forme de faisceaux.

Chapitre 7 : Distribution de l'erreur relative avec les méthodes de Metropolis. Les méthodes basées sur Metropolis-Hasting permettent d'échantillionner de façon efficace l'espace des chemins dans une scène complexe. Pour ce faire, ces méthodes utilisent une fonction d'importance qui permette de spécifier l'importance des zones à échantillionner. Le choix de cette fonction est relativement libre et plusieurs ont été proposées dans le cadre du rendu photoréaliste [VG97, HH10].

Le problème actuel de ces méthodes est la mauvaise distribution d'erreur sur le plan image. En effet, dans le cadre d'un estimateur de Monte Carlo classique, le même nombre d'échantillons est généré pour chaque pixel. Cependant, ce n'est pas le cas des algorithmes de Metropolis car le processus échantillonne tout le plan image en une seule fois. Il est donc difficile de contrôler la répartition des échantillons sur le plan image et donc de répartir l'erreur.

Dans ce chapitre, nous proposons d'utiliser une fonction d'importance qui aura pour rôle de distribuer l'erreur relative sur le plan image. Nous montrons l'efficacité de l'utilisation de cette fonction d'importance dans le cadre du photon mapping progressif et la comparons à differents travaux. Par ailleurs, nous proposons deux fonctions d'importance concrètes : une formulée dans le plan image et l'autre dans l'espace 3D. De plus, pour une meilleure robustesse, nous proposons l'utilisation du MIS et du Replica exchange pour permettre à l'algorithme d'être efficace même pour des scènes simples. Enfin, nous utilisons un processus adaptatif pour paramétrer automatiquement la technique de rendu. Le but étant de réduire le plus possible le nombre de paramètres utilisateur.

4.2 Outils d'aide à l'artiste

Chapitre 8 : Calcul de l'illuminant de référence dans le cadre de la synthèse d'image. L'illuminant de référence est la couleur moyenne reçue par un objet diffus blanc. Cet illuminant peut venir des sources de lumière comme des interactions lumineuses avec des objets colorés. Cet illuminant aura un grand impact

sur la perception du résultat final.

Par exemple, la connaissance de cet illuminant va permettre de convertir l'image générée par le moteur de rendu dans un espace de couleur neutre (ce processus est appelé aussi balance des blancs). Cet espace neutre va nous permettre d'effectuer des transformations de couleurs. Par ailleurs, à la fin de ces transformations, il est possible de revenir dans l'illuminant de référence. Par exemple, l'illuminant de référence est utilisé par certains algorithmes de transfert de style [NKB14].

Il existe deux catégories d'algorithme pour estimer l'illuminant de référence. La première est consiste à l'estimer uniquement dans l'espace image [TTP08] (par exemple pour la photographie). La deuxième l'estime dans l'espace 3D, ce qui est possible uniquement si l'on a accès à cette dernière. C'est le cas de la technique de Wilkie et Weidlich [WW09] qui permet de surpasser les techniques 2D.

Dans ce chapitre, nous proposons une nouvelle méthode d'estimation de l'illuminant de reférence dans une scène 3D. Dans notre technique, nous proposons d'utiliser l'illumination incidente à l'observateur comme illuminant de référence. Pour ce faire, nous utilisons une boule blanche virtuelle placée à la position de la caméra qui représente l'œil de l'observateur et nous calculons la luminance incidente. Pour cela, nous utilisons un simple estimateur de Monte Carlo.

Pour évaluer notre technique, nous nous comparons avec la méthode de Wilkie et Weidlich [WW09] dans le cadre de la balance des blancs. Nous montrons que notre technique permet d'avoir de meilleurs résultats. Par ailleurs, notre technique est relativement stable temporellement pour permettre son utilisation dans le cas d'un déplacementt dans la scène.

Chapitre 9 : Détermination automatique de l'éclairage d'une scène 3D. L'éclairage d'une scène 3D est un élément important dans l'esthétique de l'image finale. Cependant, il est difficile pour un artiste de trouver les bons paramètres d'éclairage. En effet, ces paramètres sont multiples comme :

- le nombre de sources de lumière et leur type (surfacique, directionelle, etc.) ;
- chaque source peut aussi avoir ses propres paramètres (taille, flux, etc.).

Shacked et Lichinski [SL01] ont proposé une méthode automatique de placement de lampes ponctuelles pour un objet 3D. Leur but étant d'optimiser l'éclairage pour rendre plus compréhensible les formes de l'objet. Pour cela, ils ont proposé d'optimiser une fonction objectif. Cette fonction est ensuite utilisée pour déterminer automatiquement les paramètres de l'éclairage.

Dans ce chapitre, nous proposons une méthode pour optimiser l'éclairage pour atteindre une certaine esthétique. Pour ce faire, nous définisons une nouvelle fonction objectif qui va modéliser cette esthétique. Avec cette fonction objectif, notre technique va optimiser certains paramètres des sources de lumière (taille et flux). Cependant, contrairement à Shacked, nous ne changeons pas la position des lampes.

De plus, nous montrons des résultats pour plusieurs scènes 3D et pour différentes esthétiques. Nous utilisons un algorithme de rendu photoréaliste pour permettre de prendre en compte l'ensemble des phénomènes lumineux.

Bibliography

[AT08] Christophe Andrieu and Johannes Thoms. A tutorial on adaptive mcmc. *Statistics and Computing*, 18(4):343–373, 2008. 63, 103, 119, 158

[BHPB+12] Daniele Bernabei, Ajit Hakke-Patil, Francesco Banterle, Marco Di Benedetto, Fabio Ganovelli, Sumanta Pattanaik, and Roberto Scopigno. A parallel architecture for interactively rendering scattering and refraction effects. *IEEE Computer Graphics and Applications*, 32:34–43, 2012. 72

[BIOP13] Guillaume Bouchard, Jean-Claude Iehl, Victor Ostromoukhov, and Pierre Poulin. Improving robustness of monte-carlo global illumination with directional regularization. In *SIGGRAPH Asia 2013 Technical Briefs*, page 22. ACM, 2013. 45

[BRDC12] Thomas Bashford-Rogers, Kurt Debattista, and Alan Chalmers. A significance cache for accelerating global illumination. In *Computer Graphics Forum*, volume 31, pages 1837–1851. Wiley Online Library, 2012. 103

[Bre87] E. J. Brenenam. Corresponding chromaticities for different states of adaptation to complex visual fields. *Journal of the Optical Society of America A*, 4:1115–1129, 1987. 126

[Buc80] G.. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310, 1980. 127

[BW92] D.H. Brainard and B.A. Wandell. asymmetric color matching : how color appearance depends on the illuminant. *Journal of the Optical Society of America A*, 9:1433–1448, 1992. 126

[CdAS12] Francesc Castro, Esteve del Acebo, and Mateu Sbert. Energy-saving light positioning using heuristic search. *Eng. Appl. Artif. Intell.*, 25(3):566–582, April 2012. 141

[CFB02] V. Cardei, B. Funt, and K. Barnard. Estimating the scene illumination chromaticity using a neural network. *Journal of the Optical Society of America*, 19(12), 2002. 127

[Chr03] Per H. Christensen. Adjoints and importance in rendering: An overview. *Visualization and Computer Graphics, IEEE Transactions on*, 9(3):329–340, 2003. 44

[CIE98] CIE. *The CIE 1997 Interim Colour Appearance Model*. John Wiley and sons, Ltd, 1998. 126

[CPCP+05] E. Cerezo, F. Perez-Cazorla, X. Pueyo, F. Seron, and F. Sillion. A survey on participating media rendering techniques. *the Visual Computer*, 2005. 72

[CRG+13] Charly Collin, Mickaël Ribardière, Adrien Gruson, Rémi Cozot, Sumanta Pattanaik, and Kadi Bouatouch. Visibility-driven progressive volume photon tracing. *The Visual Computer*, 29(9):849–859, 2013. 96

[CS02] Sung-Hyuk Cha and Sargur N. Srihari. On measuring the distance between histograms. *Pattern Recognition*, 35(6):1355 – 1370, 2002. 147, 148

[CSFN11] Fabiano Cassol, Paulo Smith Schneider, Francis H.R. FranÃğa, and AntÃťnio J. Silva Neto. Multi-objective optimization as a new approach to illumination design of interior spaces. *Building and Environment*, 46(2):331–338, 2011. 141

[CTE05] David Cline, Justin Talbot, and Parris Egbert. Energy redistribution path tracing. *ACM Transactions on Graphics (TOG)*, 24(3):1186–1195, 2005. 61

[CW95] E. J. Chichilnisky and B. A. Wandell. Photoreceptor sensitivity changes explain color appearance shifts induced by large uniform backgrounds in dichoptic matching. *Vision Research*, 53:239–254, 1995. 126

[CWY11] Jiating Chen, Bin Wang, and Jun-Hai Yong. Improved stochastic progressive photon mapping with metropolis sampling. In *Computer Graphics Forum*, volume 30, pages 1205–1213. Wiley Online Library, 2011. 7, 62, 63, 101, 103, 104

[DJLW06] Ritendra Datta, Dhiraj Joshi, Jia Li, and James Ze Wang. Studying aesthetics in photographic images using a computational approach. In *ECCV (3)*, pages 288–301, 2006. 143

[DOB11] S. Dhar, V. Ordonez, and T.L. Berg. High level describable attributes for predicting aesthetics and interestingness. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1657–1664, 2011. 143

[ENSD12] Thomas Engelhardt, Jan Novák, Thorsten-W. Schmidt, and Carsten Dachsbacher. Approximate bias compensation for rendering scenes with heterogeneous participating media. *Computer Graphics Forum (Proceedings of Pacific Graphics 2012)*, 31(7):2145–2154, 2012. 72

[ERDS14a] Oskar Elek, Tobias Ritschel, Carsten Dachsbacher, and Hans-Peter Seidel. Interactive light scattering with principal-ordinate propagation. In *Proceedings of Graphics Interface*, Montreal/Quebec/Canada, 2014. 84

[ERDS14b] Oskar Elek, Tobias Ritschel, Carsten Dachsbacher, and Hans-Peter Seidel. Principal-ordinates propagation for real-time rendering of participating media. *Computers & Graphics*, 45, 2014. 84

[Fai91a] M. D. Fairchild. Formulation and testing of an incomplete-chromatic-adaptation model. *Color Research Application*, 16:243–250, 1991. 126

[Fai91b] M. D. Fairchild. A model of incomplete chromatic adaptation. In *the 22nd Session of the CIE*, pages 33–34. CIE, 1991. 126

[Fai05] Mark D. Fairchild. *Color Appearance Model, Second edition.* John Wiley and sons, Ltd, 2005. 125, 126

[Fat09] Raanan Fattal. Participating media illumination using light propagation maps. *ACM Trans. Graph.*, 28(1):1–11, 2009. 69, 71, 72, 73, 162

[FB12] Eduardo Fernández and Gonzalo Besuievsky. Technical section: Inverse lighting design for interior buildings integrating natural and artificial sources. *Comput. Graph.*, 36(8):1096–1108, December 2012. 141

[FB14] Eduardo Fernández and Gonzalo Besuievsky. Efficient inverse lighting: A statistical approach. *Automation in Construction*, 37(1):48–57, 2014. 141

[FCL+05] Shaohua Fan, Stephen Chenney, Yu-chi Lai, et al. Metropolis photon sampling with optional user guidance. *Rendering Techniques*, 5:127–138, 2005. 61, 101, 103

[FHT06] G. D. Finlayson, S. D. Hordley, and I. Tastl. Gamut constrained illuminant estimation. *Int. J. Comput. Vision*, 67(1):93–109, 2006. 127

[FT04] Graham D. Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color Imaging Conference*, pages 37–41, 2004. 128

[GGW10] Arjan Gijsenij, Theo Gevers, and Joost Weijer. Generalized gamut mapping using image derivative structures for color constancy. *Int. J. Comput. Vision*, 86(2-3):127–139, 2010. 127, 128

[GKDS12] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192:1–192:10, November 2012. 50, 101, 103, 118, 158

[GPC+12] Adrien Gruson, Ajit Hakke Patil, Rémi Cozot, Kadi Bouatouch, and Sumanta N Pattanaik. Light propagation maps on parallel graphics architectures. In *EGPGV*, pages 81–88, 2012. 83

[GRB+08] P.V. Gehler, C. Rother, A. Blake, T. Minka, and T. Sharp. Bayesian color constancy revisited. pages 1–8, June 2008. 127

[GRC13] Adrien Gruson, Mickaël Ribardière, and Rémi Cozot. Eye-centered color adaptation in global illumination. In *Computer Graphics Forum*, volume 32, pages 111–120. Wiley Online Library, 2013. 137

[GRWS04] Robert Geist, Karl Rasche, James Westall, and Robert J. Schalkoff. Lattice-boltzmann lighting. In Alexander Keller and Henrik Wann Jensen, editors, *Proceedings of the 15th Eurographics Workshop on Rendering Techniques, NorkÃűping, Sweden, June 21-23, 2004*, pages 355–362. Eurographics Association, 2004. 72

[Gut91] SL. Guth. Model for color vision and light adaptation. *Journal of the Optical Society of America A*, 8:976–993, 1991. 126

[Has70] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. 19, 102, 163

[HBS04] Vlastimil Havran, Jiří Bittner, and Hans-Peter Seidel. Ray maps for global illumination. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, pages 77–, New York, NY, USA, 2004. ACM. 91

[HH10] Jared Hoberock and John C Hart. Arbitrary importance functions for metropolis light transport. In *Computer Graphics Forum*, volume 29, pages 1993–2003. Wiley Online Library, 2010. 62, 101, 102, 104, 111, 163

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. In *ACM Transactions on Graphics (TOG)*, volume 28, page 141. ACM, 2009. 47, 85, 86, 101, 103, 104, 105, 108, 111, 157, 162

[HJ11] Toshiya Hachisuka and Henrik Wann Jensen. Robust adaptive photon tracing using photon path visibility. *ACM Trans. Graph.*, 30(5):114:1–114:11, October 2011. 8, 62, 64, 85, 87, 89, 92, 95, 101, 103, 104, 112, 113, 114, 117, 119, 163

[HKD14] Toshiya Hachisuka, Anton S Kaplanyan, and Carsten Dachsbacher. Multiplexed metropolis light transport. *ACM Transactions on Graphics (TOG)*, 33(4):100, 2014. 60, 102, 158

[HKRs+06] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel. *Real-time Volume Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2006. 27

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Trans. Graph.*, 27:130:1–130:8, December 2008. 47, 85, 86

[Hor06] S. D. Hordley. Scene illuminant estimation: past, present, and future. *Color Research and Application*, 31(4):303–314, 2006. 127

[HPB07] Miloš Hašan, Fabio Pellacini, and Kavita Bala. Matrix row-column sampling for the many-light problem. In *ACM Transactions on Graphics (TOG)*, volume 26, page 26. ACM, 2007. 51

[HPBC+13] Ajit Hakke-Patil, Daniele Bernabei, Chaly Collins, Ke Chen, Sumanta Pattanaik, and Fabio Ganovelli. Parallel mdom for light transport in participating media. In *Spring Conference on Computer Graphics*, pages 131–138. ACM, 2013. 83

[HPJ12] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Trans. Graph.*, 31(6):191:1–191:10, November 2012. 50, 101, 103, 118, 158

[Ish78] Akira Ishimaru. *Wave propagation and scattering in random media*. Academic Press, New York, 1978. 72

[Jak10] Wenzel Jakob. Mitsuba renderer, 2010. http://www.mitsuba-renderer.org. 94, 113, 131, 151

[JC98] Henrik Wann Jensen and Per H. Christensen. Efficient simulation of light transport in scences with participating media using photon maps. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 311–320, New York, NY, USA, 1998. ACM. 85, 86, 88

[Jen95] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Rendering TechniquesâĂŹ 95*, pages 326–335. Springer, 1995. 44

[Jen01] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001. 44, 46, 85, 88

[JM12] Wenzel Jakob and Steve Marschner. Manifold exploration: a markov chain monte carlo technique for rendering scenes with difficult specular transport. *ACM Transactions on Graphics (TOG)*, 31(4):58, 2012. 57, 58, 103

[JMLH01] Henrik Wann Jensen, Stephen R Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 511–518. ACM, 2001. 24

[JNT+11] Wojciech Jarosz, Derek Nowrouzezahrai, Robert Thomas, Peter-Pike Sloan, and Matthias Zwicker. Progressive photon beams. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2011)*, 30(6), December 2011. 71, 85, 87, 90, 96

[JZJ08a] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. The beam radiance estimate for volumetric photon mapping. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2):557–566, April 2008. 71, 86, 88, 89

[JZJ08b] Wojciech Jarosz, Matthias Zwicker, and Henrik Wann Jensen. Irradiance gradients in the presence of participating media and occlusions. In *Computer Graphics Forum*, volume 27, pages 1087–1096. Wiley Online Library, 2008. 51

[Kaj86] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986. 25, 160

[KD13a] Anton S Kaplanyan and Carsten Dachsbacher. Adaptive progressive photon mapping. *ACM Transactions on Graphics (TOG)*, 32(2):16, 2013. 49

[KD13b] Anton S Kaplanyan and Carsten Dachsbacher. Path space regularization for holistic and robust light transport. In *Computer Graphics Forum*, volume 32, pages 63–72. Wiley Online Library, 2013. 45, 64, 103

[Kel97] Alexander Keller. Instant radiosity. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 51, 87

[KGH+14] Jaroslav KrivâĂŹanek, Iliyan Georgiev, Toshiya Hachisuka, Petr V'evoda, Martin Sik, Derek Nowrouzezahrai, and Wojciech Jarosz. Unifying points, beams, and paths in volumetric light transport simulation. *ACM Trans. Graph.*, 33(4):1âĂŞ13, August 2014. 50, 103

[KGPB05] Jaroslav Krivanek, Pascal Gautron, Sumanta Pattanaik, and Kadi Bouatouch. Radiance caching for efficient global illumination computation. *Visualization and Computer Graphics, IEEE Transactions on*, 11(5):550–561, 2005. 51

[KHD14] Anton S Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. The natural-constraint representation of the path space for efficient light transport simulation. *ACM Transactions on Graphics (TOG)*, 33(4):102, 2014. 58, 103

[KJF07] J. Kuang, G. M. Johnson, and M. D. Fairchild. icam06: A refined image appearance model for hdr image rendering. *Journal of Visual Communication*, 2007. 128

[KKK09] Shinya Kitaoka, Yoshifumi Kitamura, and Fumio Kishino. Replica exchange light transport. In *Computer Graphics Forum*, volume 28, pages 2330–2342. Wiley Online Library, 2009. 64, 65, 103, 112, 113

[KPC93] John K. Kawai, James S. Painter, and Michael F. Cohen. Radioptimization: goal based rendering. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH, pages 147–154, 1993. 140, 141, 149

[KR09] Timo Kunkel and Erik Reinhard. A neurophysiology-inspired steady-state color appearance model. *Journal of the Optical Society of America A*, 26(4):776–782, April 2009. 128

[KSKAC02] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A simple and robust mutation strategy for the metropolis light transport algorithm. In *Computer Graphics Forum*, volume 21, pages 531–540. Wiley Online Library, 2002. 59, 61, 64, 65, 104, 105, 158

[KZ11] Claude Knaus and Matthias Zwicker. Progressive photon mapping: A probabilistic approach. *ACM Trans. Graph.*, 30:25:1–25:13, May 2011. 47, 85, 87, 89, 94, 96, 163

[Lan77] E. H. Land. The retinex theory of color vision. *Scientic American*, 237(6):108–120, 1977. 127, 128

[LBC94] E. Languenou, K. Bouatouch, and M. Chelle. Global illumination in presence of participating media with general properties. *Proceedings du 5th Eurographics Workshop on Rendering*, 1994. 72, 83

[LFCD07] Yu-Chi Lai, Shao Hua Fan, Stephen Chenney, and Charcle Dyer. Photorealistic image rendering with population monte carlo energy redistribution. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 287–295. Eurographics Association, 2007. 64, 158

[LGCB14] V Léon, A Gruson, R Cozot, and K Bouatouch. Automatic aesthetics-based lighting design with global illumination. 2014. 156

[LKL+13] Jaakko Lehtinen, Tero Karras, Samuli Laine, Miika Aittala, Frédo Durand, and Timo Aila. Gradient-domain metropolis light transport. *ACM Transactions on Graphics (TOG)*, 32(4):95, 2013. 61, 102

[LLW00] Jun S Liu, Faming Liang, and Wing Hung Wong. The multiple-try method and local optimization in metropolis sampling. *Journal of the American Statistical Association*, 95(449):121–134, 2000. 61

[LM71] EDWIN H. LAND and JOHN J. McCANN. Lightness and retinex theory. *J. Opt. Soc. Am.*, 61(1):1–11, Jan 1971. 128

[LW93] Eric P Lafortune and Yves D Willems. Bi-directional path tracing. In *Proceedings of CompuGraphics*, volume 93, pages 145–153, 1993. 42

[MFH+02] Y. N. Morone, M. D. Fairchild, R. W. G. Hunt, C. Li, M. R. Lou, and T. Newman. The ciecam02 color appearance model. In *In Color Imaging Conference (2002), IS&T*, pages 23–27. Society for Imaging Science and Technology, 2002. 128, 131

[MFSG08] Miguel Martin, Roland Fleming, Olga Sorkine, and Diego Gutierrez. Understanding exposure for reverse tone mapping. In *Congreso Espanol de Informática Gráfica*, pages 189–198, 2008. 147

[MPS09] Jean-Michel Morel, Ana B. Petro, and Catalina Sbert. Fast implementation of color constancy algorithms. In *Proc. SPIE 7241, Color Imaging XIV: Displaying, Processing, Hardcopy, and Applications, 724106 (January 19, 2009)*, 2009. 128

[MPS10] Jean Michel Morel, Ana Belén Petro, and Catalina Sbert. A pde formalization of retinex theory. *Trans. Img. Proc.*, 19(11):2825–2837, November 2010. 128

[MRR+53] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953. 19, 53, 163

[NCNS03] Laszlo Neumann, Francesc Castro, Attila Neumann, and Mateu Sbert. Color appearance in multispectral radiosity. In G. Renner L. Szirmay-Kalos, editor, *Proceedings on the 2nd Hungarian Computergraphics and Geometry Conference*, pages 183–194, 2003. 125, 128, 129

[Nea96] Radford M Neal. Sampling from multimodal distributions using tempered transitions. *Statistics and computing*, 6(4):353–366, 1996. 103, 112

[NKB14] RMH Nguyen, SJ Kim, and MS Brown. Illuminant aware gamut-based color transfer. In *Computer Graphics Forum*, volume 33, pages 319–328. Wiley Online Library, 2014. 123, 164

[NNDJ12a] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Progressive virtual beam lights. *Computer Graphics Forum (Proceedings of EGSR 2012)*, 31(4), June 2012. 71, 87

[NNDJ12b] Jan Novák, Derek Nowrouzezahrai, Carsten Dachsbacher, and Wojciech Jarosz. Virtual ray lights for rendering scenes with participating media. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2012)*, 31(4), July 2012. 87, 96

[NTS81] Y. Nayatani, K. Takahama, and H. Sobagaki. Formulation of anonlinear model of chromatic adaptation. *Color Research Application*, 6:161–171, 1981. 126

[num13] Numpy 1.6.2. http://www.numpy.org/, 2013. [Online; accessed May 2013]. 151

[ON94] Michael Oren and Shree K Nayar. Generalization of lambert's reflectance model. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 239–246. ACM, 1994. 24

[ope13] Opencv 2.4.4. http://www.opencv.org/, 2013. [Online; accessed May 2013]. 151

[PBPP11] Anthony Pajot, Loic Barthe, Mathias Paulin, and Pierre Poulin. Representativity for robust and adaptive multiple importance sampling. *Visualization and Computer Graphics, IEEE Transactions on*, 17(8):1108–1121, 2011. 38

[Pes73] Peter H Peskun. Optimum monte-carlo sampling using markov chains. *Biometrika*, 60(3):607–612, 1973. 55

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2nd edition, 2010. 24, 125

[PP98] Ingmar Peter and Georg Pietrek. *Importance driven construction of photon maps*. Springer, 1998. 103

[PP03] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. *Computer Graphics Forum*, 22(4):663–687, 2003. 140

[PPS97] Frederic Pérez, Xavier Pueyo, and François X. Sillion. Global illumination techniques for the simulation of participating media. *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, 1997. 72

[PW00] Panos Y Papalambros and Douglass J Wilde. *Principles of optimal design: modeling and computation*. Cambridge university press, 2000. 141

[R+11] Jeffrey S Rosenthal et al. Optimal proposal distributions and adaptive mcmc. *Handbook of Markov Chain Monte Carlo*, pages 93–112, 2011. 64

[RB07] Alexa I. Ruppertsberg and Marina Bloj. Reflecting on a room of one reflectance. *Journal of Vision*, 7(13):–, 2007. 128

[RCB11] Mickaël Ribardière, Samuel Carré, and Kadi Bouatouch. Adaptive records for irradiance caching. In *Computer Graphics Forum*, volume 30, pages 1603–1616. Wiley Online Library, 2011. 51

[RDGK12] Tobias Ritschel, Carsten Dachsbacher, Thorsten Grosch, and Jan Kautz. The state of the art in interactive global illumination. In *Computer Graphics Forum*, volume 31, pages 160–188. Wiley Online Library, 2012. 51

[RGK+08] Tobias Ritschel, Thorsten Grosch, Min H Kim, H-P Seidel, Carsten Dachsbacher, and Jan Kautz. Imperfect shadow maps for efficient computation of indirect illumination. In *ACM Transactions on Graphics (TOG)*, volume 27, page 129. ACM, 2008. 51

[RPK+12] Erik Reinhard, Tania Pouli, Timo Kunkel, Ben Long, Anders Ballestad, and Gerwin Damberg. Calibrated image appearance reproduction. *ACM Trans. Graph.*, 31(6):201:1–201:11, November 2012. 128

[sci13] Scipy 0.12.0. http://www.scipy.org/, 2013. [Online; accessed May 2013]. 151

[SDS+93] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald Greenberg. Painting with light. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH, pages 143–146, 1993. 140, 142

[SH01] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer, 4th Revised edition*. Taylor & Francis Inc, 2001. 71, 72

[Sic13] Julien Sicre. *Classifying images by light aesthetics*. PhD thesis, IRISA Rennes Bretagne Atlantique, équipe FRVSense, 2013. 147

[SIP07] Benjamin Segovia, Jean Claude Iehl, and Bernard Péroche. Metropolis instant radiosity. In *Computer Graphics Forum*, volume 26, pages 425–434. Wiley Online Library, 2007. 61

[SJJ12] Jorge Schwarzhaupt, Henrik Wann Jensen, and Wojciech Jarosz. Practical hessian-based error control for irradiance caching. *ACM Transactions on Graphics (TOG)*, 31(6):193, 2012. 51

[SKLU+09] LÃąszlÃş Szirmay-Kalos, Gabor Liktor, TamÃąs Umenhoffer, BalÃązs TÃşth, Shree Kumar, and Glenn Lupton. Parallel solution to the radiative transport. In Kurt Debattista, Daniel Weiskopf, and JoÃčo Comba, editors, *EGPGV*, pages 95–102. Eurographics Association, 2009. 72

[SKLU+11] Laszlo Szirmay-Kalos, Gabor Liktor, Tamas Umenhoffer, Balazs Toth, Shree Kumar, and Glenn Lupton. Parallel iteration to the radiative transport in inhomogeneous media with bootstrapping. *IEEE Transactions on Visualization and Computer Graphics*, 17(2):146–158, February 2011. 72

[SL01] Ram Shacked and Dani Lischinski. Automatic lighting design using a perceptual quality metric. *Computer Graphics Forum*, 20:2001, 2001. 140, 141, 142, 143, 145, 146, 164

[Sta95] Jos Stam. Multiple scattering as a diffusion process. In *In Eurographics Rendering Workshop*, pages 41–50, 1995. 72

[Sub60] Chandrasekhar Subrahmanyan. *Radiative Transfer*. Dover Publications, 1960. 27, 71, 87

[SVLL10] Marco Salvi, Kiril Vidimče, Andrew Lauritzen, and Aaron Lefohn. Adaptive volumetric shadow maps. In *Computer Graphics Forum*, volume 29, pages 1289–1296. Wiley Online Library, 2010. 83

[SWZ96] Peter Shirley, Changyaw Wang, and Kurt Zimmerman. Monte carlo techniques for direct lighting calculations. *ACM Transactions on Graphics (TOG)*, 15(1):1–36, 1996. 38

[TCE05] Justin F. Talbot, David Cline, and Parris Egbert. Importance resampling for global illumination. In *Proceedings of the Sixteenth Eurographics Conference on Rendering Techniques*, EGSR '05, pages 139–146, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association. 54

[TS67] Kenneth E Torrance and Ephraim M Sparrow. Theory for off-specular reflection from roughened surfaces. *JOSA*, 57(9):1105–1112, 1967. 24

[TTP08] Alain Tremeau, Shoji Tominaga, and Konstantinos N. Plataniotis. Color in image and video processing: Most recent trends and future research directions. *EURASIP Journal on Image and Video Processing*, 2008, 2008. 127, 164

[VdWSV07] Joost Van de Weijer, Cordelia Schmid, and Jakob Verbeek. Using high-level visual information for color constancy. In *IEEE Conference on Computer Vision (ICCV)*, 2007. 127

[Vea97] Eric Veach. *Robust Monte Carlo methods for light transport simulation*. PhD thesis, Stanford University, 1997. 26, 28, 34, 37, 43, 53, 61, 101, 102, 104, 105, 107, 113, 160

[VG95] Eric Veach and Leonidas Guibas. Bidirectional estimators for light transport. In *Photorealistic Rendering Techniques*, pages 145–167. Springer, 1995. 42

[VG97] Eric Veach and Leonidas J Guibas. Metropolis light transport. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 65–76. ACM Press/Addison-Wesley Publishing Co., 1997. 53, 54, 56, 57, 61, 102, 158, 163

[VK70] J. Von Kries. *Chromatic adaptation*, pages 109–119. MIT Press, 1970. 126, 130

[VKŠ+14] Jiří Vorba, Ondřej Karlík, Martin Šik, Tobias Ritschel, and Jaroslav Křivánek. On-line learning of parametric mixture models for light transport simulation. *ACM Transactions on Graphics (TOG)*, 33(4):101, 2014. 8, 51, 101, 102, 103, 113, 114, 117, 118

[WABG06] Bruce Walter, Adam Arbree, Kavita Bala, and Donald P Greenberg. Multidimensional lightcuts. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 1081–1088. ACM, 2006. 51

[WEV02] Greg Ward and Elena Eydelberg-Vileshin. Picture perfect rgb rendering using spectral prefiltering and sharp color primaries. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages

117–124, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association. 128

[WFA+05] Bruce Walter, Sebastian Fernandez, Adam Arbree, Kavita Bala, Michael Donikian, and Donald P Greenberg. Lightcuts: a scalable approach to illumination. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 1098–1107. ACM, 2005. 51

[WH92] Gregory J Ward and Paul Heckbert. Irradiance gradients. In *Third Eurographics Workshop on Rendering*, volume 8598, 1992. 51

[WKB12] Bruce Walter, Pramook Khungurn, and Kavita Bala. Bidirectional lightcuts. *ACM Transactions on Graphics (TOG)*, 31(4):59, 2012. 51

[WKSD13] Christoph Weber, Anton Kaplanyan, Marc Stamminger, and Carsten Dachsbacher. Interactive direct volume rendering with many-light methods and transmittance caching. In *VMV*, pages 195–202, 2013. 83

[WMLT07] Bruce Walter, Stephen R Marschner, Hongsong Li, and Kenneth E Torrance. Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics conference on Rendering Techniques*, pages 195–206. Eurographics Association, 2007. 24

[WRC88] Gregory J Ward, Francis M Rubinstein, and Robert D Clear. A ray tracing solution for diffuse interreflection. *ACM SIGGRAPH Computer Graphics*, 22(4):85–92, 1988. 51

[WW09] A. Wilkie and A. Weidlich. A robust illumination estimate for chromatic adaptation in rendered images. In *Eurographics Symposium on Rendering 2009*, 2009. 124, 128, 129, 132, 133, 164

[WWH+10] Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo. Real-time rendering of heterogeneous translucent objects with arbitrary shapes. *Computer Graphics Forum (Proceedings of Eurographics 2010)*, 2010. 72

[YCK05] Kuk-Jin Yoon, Yoo Jin Chofi, and In-So Kweon. Dichromatic-based color constancy using dichromatic slope and dichromatic line space. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 3, pages III–960–3, 2005. 128

[ZBLN97] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, December 1997. 149, 151

[ZEN09] Joseph Zupko and Magy Seif El-Nasr. System for automated interactive lighting (sail). In *Proceedings of the 4th International Conference on Foundations of Digital Games*, FDG '09, pages 223–230, New York, NY, USA, 2009. ACM. 141, 142

[ZRL+08] K. Zhou, Z. Ren, S. Lin, H. Bao, B. Guo, and H-Y. Shum. Real-time smoke rendering using compensated ray marching. *ACM Trans. Graph.*, 27(3):36, 2008. 72

[ZSK13] Károly Zsolnai and László Szirmay-Kalos. Automatic parameter control for metropolis light transport. In *Eurographics 2013 Short Papers*, pages 53–56. Eurographics Association, 2013. 64

ANNEXE 2 (Modèle dernière page de thèse)

VU :

Le Directeur de Thèse
(Nom et Prénom)

VU :

Le Responsable de l'École Doctorale

VU pour autorisation de soutenance

Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINEAU

VU après soutenance pour autorisation de publication :

Le Président de Jury,
(Nom et Prénom)